SAT

Informatique Théorique 2
Licence 3 informatique

Sébastien Verel
verel@univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Objectifs de la séance 06

- Connaitre la logique propostionnelle
- Connaitre le problème SAT et 3-SAT
- Savoir réduire un problème en problème SAT
- Connaitre les définitions de P, NP et NP-complet
- Connaitre le théorème de Cook et Levin.

Plan

George Boole

Mathématicien et logicien anglais (1815 - 1864)

but

Traduire des idées et des concepts en équations,
leur appliquer des lois (des transformations) et
traduire inversement l'équation en termes de concepts et d'idées.

Il crée une algèbre binaire :

- qui n'accepte que deux valeurs numériques 0 et 1 (faux, vrai),
- définie dans un ensemble E muni de deux lois de compositions interne (*et* ,*ou*)
- satisfaisant un certain nombre de propriétés (associativité, distributivité).

$\longrightarrow$ algèbre de Boole

Notations

Il existe plusieurs types de notations :

- Vrai $\longleftrightarrow$ V $\longleftrightarrow$ 1
- Faux $\longleftrightarrow$ F $\longleftrightarrow$ 0
- NON $\longleftrightarrow$ $\neg$
- ET $\longleftrightarrow$ $\wedge$
- OU $\longleftrightarrow$ $\vee$ (attention inclusif !)
- IMPLICATION $\longleftrightarrow$ $\Rightarrow$
- EQUIVALENT $\longleftrightarrow$ $\Leftrightarrow$

Notations

Il existe plusieurs types de notations :

- Vrai $\longleftrightarrow$ V $\longleftrightarrow$ 1
- Faux $\longleftrightarrow$ F $\longleftrightarrow$ 0
- NON $\longleftrightarrow$ $\neg$
- ET $\longleftrightarrow$ $\wedge$
- OU $\longleftrightarrow$ $\vee$ (attention inclusif !)
- IMPLICATION $\longleftrightarrow$ $\Rightarrow$
- EQUIVALENT $\longleftrightarrow$ $\Leftrightarrow$
- XOR $\longleftrightarrow$ $\oplus$ (ou exclusif : fromage ou dessert)

Définitions

Littéral

variable dont la valeur de vérité est soit 1 (VRAI) soit 0 (FAUX).

Définitions

Littéral

variable dont la valeur de vérité est soit 1 (VRAI) soit 0 (FAUX).

Proposition

Enoncé a qui l'on associe une valeur de vérité (0 ou 1)

Définitions

Littéral

variable dont la valeur de vérité est soit 1 (VRAI) soit 0 (FAUX).

Proposition

Enoncé a qui l'on associe une valeur de vérité (0 ou 1)

Tautologie

est une formule qui est toujours 1 quelque soit les valeurs de vérité des littéraux

Exemples

- $a \vee b$
- $(a \vee b) \oplus c$
- $a \wedge a$
- $a \vee 1$
- $b \vee \rceil b$
- $\rceil (b \vee a)$
- $(a \wedge \rceil b) \vee (b \wedge \rceil a)$

Tables de vérité

$\wedge$	0	1
0		
1		

$\oplus$	0	1
0		
1		

$\vee$	0	1
0		
1		

$\Rightarrow$	0	1
0		
1		

Zoom sur l'implication

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

$A =$ "je plonge dans la piscine"
$B =$ "je suis mouillé"

Zoom sur l'implication

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

$A =$ "je plonge dans la piscine"
$B =$ "je suis mouillé"

"SI je plonge dans la piscine
ALORS je suis mouillé"
est un théorème VRAI.

Zoom sur l'implication

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

$A =$ "je plonge dans la piscine"
$B =$ "je suis mouillé"

"SI je plonge dans la piscine
ALORS je suis mouillé"
est un théorème VRAI.

$A \Rightarrow B$

Lorsque A est VRAI, la condition est réalisée, donc B est VRAI.

Zoom sur l'implication

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

$A =$ "je plonge dans la piscine"
$B =$ "je suis mouillé"

"SI je plonge dans la piscine
ALORS je suis mouillé"
est un théorème VRAI.

$A \Rightarrow B$

Lorsque A est VRAI, la condition est réalisée, donc B est VRAI.

A peut aussi être FAUX et B restant VRAI.

Peut-on en déduire que le théorème devient FAUX dans ce cas ?

Zoom sur l'implication

L'implication est vraie si l'hypothèse est fausse.

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

Lorqu'un théorème $p \Rightarrow q$ est vrai,
mais que l'on a pas l'hypothèse, alors on ne peut rien en déduire sur q.

Zoom sur l'implication

L'implication est vraie si l'hypothèse est fausse.

p	q	$\Rightarrow$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

Lorqu'un théorème $p \Rightarrow q$ est vrai,
mais que l'on a pas l'hypothèse, alors on ne peut rien en déduire sur q.

Pierre Weis et Xavier Leroy

"on ne peut rien déduire d'un théorème dont l'hypothèse n'est pas vérifiée"
"un théorème reste vrai même quand il ne s'applique pas"

Tables de vérité

Un connecteur logique booléen est défini par une table de vérité et réciproquement.

Il existe 16 connecteurs logiques binaires (pourquoi ?).

Tables de vérité

Un connecteur logique booléen est défini par une table de vérité et réciproquement.

Il existe 16 connecteurs logiques binaires (pourquoi ?).

	0	1
0	?	?
1	?	?

Tables de vérité

Un connecteur logique booléen est défini par une table de vérité et réciproquement.

Il existe 16 connecteurs logiques binaires (pourquoi ?).

	0	1
0	?	?
1	?	?

D'où $2 \times 2 \times 2 \times 2 = 16$ connecteurs logiques binaires.

Logiquement équivalent

Logiquement équivalent

F est logiquement équivalent à G si et seulement si $F \Leftrightarrow G$ est une tautologie.

Logiquement équivalent

F est logiquement équivalent à G si et seulement si F et G ont la même table de vérité.

Notation

$$F \equiv G$$

Exemple

a	b	$a \wedge \neg b$	$b \wedge \neg a$	$(a \wedge \neg b) \vee (b \wedge \neg a)$
0	0			
0	1			
1	0			
1	1			

Exemple

a	b	$a \wedge \rceil b$	$b \wedge \rceil a$	$(a \wedge \rceil b) \vee (b \wedge \rceil a)$
0	0			
0	1			
1	0			
1	1			

$((a \wedge \rceil b) \vee (b \wedge \rceil a)) \equiv$

Exemple

a	b	$a \wedge \neg b$	$b \wedge \neg a$	$(a \wedge \neg b) \vee (b \wedge \neg a)$
0	0			
0	1			
1	0			
1	1			

$((a \wedge \neg b) \vee (b \wedge \neg a)) \equiv (a \oplus b)$

1. Montrer que $a \Rightarrow b \equiv (\neg a \vee b)$
2. Montrer que $a \Leftrightarrow b \equiv (\neg a \vee b) \wedge (a \vee \neg b)$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- Commutativité :
  $p \vee q \Leftrightarrow q \vee p$
  $p \wedge q \Leftrightarrow q \wedge p$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- Commutativité :
  $p \vee q \Leftrightarrow q \vee p$
  $p \wedge q \Leftrightarrow q \wedge p$
- Distributivité :
  $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- Commutativité :
  $p \vee q \Leftrightarrow q \vee p$
  $p \wedge q \Leftrightarrow q \wedge p$
- Distributivité :
  $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$
- Loi de De Morgan :
  $\rceil(p \vee q) \Leftrightarrow \rceil q \wedge \rceil p$
  $\rceil(p \wedge q) \Leftrightarrow \rceil q \vee \rceil p$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- Commutativité :
  $p \vee q \Leftrightarrow q \vee p$
  $p \wedge q \Leftrightarrow q \wedge p$
- Distributivité :
  $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$
- Loi de De Morgan :
  $\rceil(p \vee q) \Leftrightarrow \rceil q \wedge \rceil p$
  $\rceil(p \wedge q) \Leftrightarrow \rceil q \vee \rceil p$
- Contraposée : $(p \Rightarrow q) \Leftrightarrow (\rceil q \Rightarrow \rceil p)$
  $(p \Rightarrow q) \Leftrightarrow (\rceil p \vee q)$

Propriétés algébriques

- Associativité :
  $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
  $p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
- Commutativité :
  $p \vee q \Leftrightarrow q \vee p$
  $p \wedge q \Leftrightarrow q \wedge p$
- Distributivité :
  $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
  $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$
- Loi de De Morgan :
  $\rceil(p \vee q) \Leftrightarrow \rceil q \wedge \rceil p$
  $\rceil(p \wedge q) \Leftrightarrow \rceil q \vee \rceil p$
- Contraposée : $(p \Rightarrow q) \Leftrightarrow (\rceil q \Rightarrow \rceil p)$
  $(p \Rightarrow q) \Leftrightarrow (\rceil p \vee q)$
- Preuve par l'absurde : $\rceil\rceil p \Rightarrow p$

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Toutes les cellules infectées par un virus ne le reconnaissent pas

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Toutes les cellules infectées par un virus ne le reconnaissent pas

$$\forall x \exists y, (C(x) \wedge V(y) \wedge I(x, y)) \Rightarrow \rceil R(x, y)$$

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Toutes les cellules infectées par un virus ne le reconnaissent pas

$$\forall x \exists y, (C(x) \wedge V(y) \wedge I(x, y)) \Rightarrow \rceil R(x, y)$$

Il n'existe pas de virus qui peuvent infecter toutes les cellules

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Toutes les cellules infectées par un virus ne le reconnaissent pas

$$\forall x \exists y, \ (C(x) \wedge V(y) \wedge I(x, y)) \Rightarrow \rceil R(x, y)$$

Il n'existe pas de virus qui peuvent infecter toutes les cellules

$$\nexists y, \ V(y) \Rightarrow (\forall x \ C(x) \wedge I(x, y))$$

Une application en biologie

$C(x)$ est VRAIE lorsque x est une cellule
$V(x)$ est VRAIE lorsque x est un virus

$I(x, y)$ est VRAIE lorsque x est infecté par y.
$R(x, y)$ est VRAIE lorsque x a reconnu y

Toutes les cellules infectées par un virus ne le reconnaissent pas

$$\forall x \exists y,\ (C(x) \wedge V(y) \wedge I(x, y)) \Rightarrow \neg R(x, y)$$

Il n'existe pas de virus qui peuvent infecter toutes les cellules

$$\nexists y,\ V(y) \Rightarrow (\forall x\ C(x) \wedge I(x, y))$$

$\longrightarrow$ technique de "model checking" : vérification de la formule logique par parcours astucieux des états possibles

Forme normale

Idée

Mettre les expression sous une forme "commune", facile à comparer, et facile à utiliser.

Vocabulaire, définition

- littéral : toute formule de la forme x ou $\rceil x$
- forme conjonctive : toute formule de la forme $\varphi_1 \wedge \ldots \wedge \varphi_n$
- forme disjonctive : toute formule de la forme $\varphi_1 \vee \ldots \vee \varphi_n$

Forme normale conjonctive

Définition : Forme Normale Conjonctive (CNF en anglais)

Une formule est sous forme normale conjonctive lorsqu'elle est sous la forme d'une conjonction de disjonctions de littéraux :

$$(\ell_{1,1} \vee \ldots \vee \ell_{1,k_1}) \wedge \ldots \wedge (\ell_{m,1} \vee \ldots \vee \ell_{m,k_m})$$

ce qui s'écrit sans les pointillés par :

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} \ell_{i,j}$$

Transformation

Mise sous forme CNF

En utilisant :

- la définition de $\Rightarrow$ et $\Leftrightarrow$,
- les lois de de Morgan, la double négation,
- la distributivité de $\wedge$ par rapport au $\vee$,

on peut toujours mettre toute formule en CNF.

Pour comparer 2 formules, il suffit alors d'écrire les clauses par ordre alphabétique par exemple.

Mettre sous forme CNF la formule suivante :

$$\rceil(a \Rightarrow b) \vee \rceil(c \vee \rceil(d \Rightarrow a))$$

Evaluation d'une formule

Définition : valuation d'une formule

Etant donné une formule φ dont les variables sont $X = (x_1, \ldots, x_n)$, une valuation de ν est une fonction des variables propositionnelle X dans $\{0, 1\}$, c'est-à-dire qu'une valuation associe à chaque variable de la formule une valeur booléenne.

Définition : évaluation d'une formule

Etant donné une formule φ et une valuation ν, une évaluation de la formule φ avec ν consiste à substituer les valeurs booléennes aux variables propositionnelles.
Notation $[\varphi, \nu]$.

Exemple, $\varphi \equiv (x_1 \vee x_2) \wedge (x_2 \vee x_3)$ et $\nu = (1, 0, 0)$, alors $[\varphi, \nu] =$

Evaluation d'une formule

Définition : valuation d'une formule

Etant donné une formule φ dont les variables sont $X = (x_1, \ldots, x_n)$, une valuation de ν est une fonction des variables propositionnelle X dans $\{0, 1\}$, c'est-à-dire qu'une valuation associe à chaque variable de la formule une valeur booléenne.

Définition : évaluation d'une formule

Etant donné une formule φ et une valuation ν, une évaluation de la formule φ avec ν consiste à substituer les valeurs booléennes aux variables propositionnelles.
Notation $[\varphi, \nu]$.

Exemple, $\varphi \equiv (x_1 \vee x_2) \wedge (x_2 \vee x_3)$ et $\nu = (1, 0, 0)$, alors $[\varphi, \nu] = 0$.

Problème de satisfiabilité

Définition : problème de satisfiabilité d'une formule

Etant donné une formule φ, existe-t-il une valuation ν qui satisfait la formule φ c'est-à-dire $[\varphi, \nu] = 1$?

Exemple,
$\varphi \equiv (x_1 \vee x_2) \wedge (x_2 \vee x_3)$

Problème SAT

Définition : problème SAT

Le problème SAT est le problème qui consiste à résoudre un problème de satisfiabilité d'une formule sous forme normale conjonctive :

$$(\ell_{1,1} \vee \ldots \vee \ell_{1,k_1}) \wedge \ldots \wedge (\ell_{m,1} \vee \ldots \vee \ell_{m,k_m})$$

Les formules $\ell_{i,1} \vee \ldots \vee \ell_{1,k_i}$ sont appelées des clauses.

Autrement dit, résoudre le problème SAT consiste à trouver une valuation telle que chaque clause de la formule est vraie.

Instance

Une instance du problème SAT est une formule particulière.
Ex. $\varphi \equiv (x_1 \vee x_2) \wedge (x_2 \vee x_3)$

$|\varphi|$ est la taille du problème, le nombre m de clauses.
n est le nombre de variables de la formule.

Résolution de SAT ?

Est-ce un problème "simple" ?
Quelle est la classe de complexité du meilleur algorithme de résolution prenant entrée n'importe quelle formule ?

Exercice

Quelle est le nombre de valuations candidates ?
Pouvez-vous définir un algorithme "brute force" de résolution ?
Quelle est sa complexité ?

$$\mathrm{NP} = \mathrm{P} \,?$$

NP = P ?

L'un des plus grand problème ouvert de l'informatique.
L'un des 7 problèmes "Clay Millennium Problems", 10^6 dollards
Turing, Nash, Gödel, Karp, etc. ont étudié ce problème.

Back to SAT

Problème 3-SAT

Problème 3-SAT est un problème SAT où chaque clause est composée de 3 littéraux.
Ex : $(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_4 \vee x_2 \vee \neg x_3)$

Une instance de 3-SAT est une instance de SAT.

Back to SAT

Problème 3-SAT

Problème 3-SAT est un problème SAT où chaque clause est composée de 3 littéraux.
Ex : $(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_4 \vee x_2 \vee \neg x_3)$

Une instance de 3-SAT est une instance de SAT.

Si on résoud le problème SAT avec un algorithme de complexité polynomiale, alors on résoud le problème 3-SAT avec un algorithme de complexité polynomiale.

Comparer deux problèmes

Question

Peut-on réduire n'importe quelle instance φ du problème SAT en une instance φ_3 du problème 3-SAT ?

Ici réduire est synonyme de transformer, ré-écrire de manière équivalente.

Réduction de SAT vers 3-SAT

Lorsque φ_3 est une réduction de φ alors
φ est satisfiable si et seulement si φ_3 est satisfiable.

Définition : Réduction

Une réduction d'un problème A à un problème B est une fonction (calculable) f telle que pour toute instance w de A,

w est une instance positive de A
si et seulement si
$f(w)$ est une instance positive de B.

On dit que A se réduit en B s'il existe une réduction de A à B.
On note alors $A \leq B$.

Intuitivement lorsque $A \leq B$, A est plus facile que B.
En effet, pour trouver une solution de A, on peut rechercher une solution B. Si un algorithme permet de résoudre B, alors l'algorithme et la réduction permet de résoudre A.

Réduction de SAT vers 3-SAT

Théorème

Le problème SAT se réduit au problème 3-SAT

Réduction de SAT vers 3-SAT

Théorème

Le problème SAT se réduit au problème 3-SAT

Démonstration : transformer une formule φ de SAT en une formule équivalente φ_3 de 3-SAT.
Soit $\varphi \equiv c_1 \wedge c_2 \wedge \ldots \wedge c_m$ où c_i est une clause.
Construisons $\varphi_3 \equiv \varphi_1^{'} \wedge \varphi_2^{'} \wedge \ldots \wedge \varphi_m^{'}$ telles que :

- $\varphi_i^{'}$ est une conjonction de 3-clause.
- $\varphi_i^{'}$ est construite avec les variables de c_i et d'autres variables si nécessaire
- Si une évaluation de c_i est vraie, alors $\varphi_i^{'}$ peut être vraie en donnant des valeurs aux autres variables
- Réciproquement, si une évaluation de $\varphi_i^{'}$ est vraie, alors l'évaluation de c_i vraie.

Preuve : Construction de $\varphi_i^{'}$

- 1 littéral : si $c_i = \ell_1$, alors on ajoute 2 nouvelles variables x_i et y_i et $\varphi_i^{'} = (\ell_1 \vee x_i \vee y_i) \wedge (\ell_1 \vee x_i \vee \rceil y_i) \wedge (\ell_1 \vee \rceil x_i \vee y_i) \wedge (\ell_1 \vee \rceil x_i \vee \rceil y_i)$

Si $\varphi_i^{'}$ est vraie alors les 4 disjonctions sont vraies, or l'un des couples "x_i" $\vee$ "y_i" est faux donc ℓ_1 doit être vraie. Donc $\varphi_i^{'}$ vraie ssi c_i vraie.

- 2 littéraux : si $c_i = \ell_1 \vee \ell_2$, alors on ajoute 1 nouvelle variable x_i et $\varphi_i^{'} = (\ell_1 \vee \ell_2 \vee x_i) \wedge (\ell_1 \vee \ell_2 \vee \rceil x_i)$

Si $\varphi_i^{'}$ est vraie alors les 2 disjonctions sont vraies, or l'un des x_i ou $\rceil x_i$ est faux donc $\ell_1 \vee \ell_2$ doit être vraie. Donc $\varphi_i^{'}$ vraie ssi c_i vraie.

- 3 littéraux : facile, rien à changer, $\varphi^{'} = c_i$.

- au moins 4 littéraux : si $c_i = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_k$, alors on ajoute $k-3$ nouvelles variables et

$\varphi_i^{'} =$

$(\ell_1 \vee \ell_2 \vee x_{i,1}) \wedge (\rceil x_{i,1} \vee x_{i,2} \vee \ell_3) \wedge (\rceil x_{i,2} \vee x_{i,3} \vee \ell_4) \wedge \ldots \wedge (\rceil x_{i,k-3} \vee \ell_{k-1} \vee \ell_k)$

Si $\varphi_i^{'}$ est vraie alors toutes les disjonctions sont vraies,
si $\ell_1 \vee \ell_2$ est vraie alors c_i est vraie, sinon $x_{i,1} = 1$ et $\rceil x_{i,1} = 0$.
si $\ell_3 = 1$, alors c_i est vraie, sinon $x_{i,2} = 1$, etc. Donc $\varphi_i^{'}$ vraie ssi c_i vraie.

Preuve : the end

Finalement, c_i est satisfiable si et seulement si φ_i' est satisfiable.

Les variables ajoutées sont spécifiques à chaque clause.

Donc :

$$\varphi \text{ est satisfiable si et seulement } \varphi_3 \text{ est satisfiable.}$$

$$\text{donc : SAT} \leq \text{3-SAT}$$

De plus, l'algorithme de réduction est de complexité polynomiale en fonction de $|\varphi|$ et du nombre de variables n.
Or, comme 3-SAT $\leq$ SAT, le problème SAT est aussi "difficile" à résoudre que le problème 3-SAT.

Algorithme recherche1( ) : bool
début
variable i : entier
  $i \leftarrow 0$
  tant que $i < n$ et $t[i] \neq x$ **faire**
    $i \leftarrow i + 1$
  fin tant que

Classe P (Polynomial-time)

Définition : classe de complexité P

La classe P est l'ensemble des problèmes de décision résolus par une machine de Turing déterministe en temps polynomial en fonction de la taille d'entrée.

Remarque : Comme tout langage de programmation se code en temps polynomial en une machine de Turing, la classe P est la classe des problèmes résolus en temps polynomial par un algorithme dans n'importe quel langage.

Exemples : recherche d'un facteur dans un mot, multiplication de matrices, calcul du chemin le plus court dans un graphe (algorithme de Dijkstra), etc.

Classe NP (Nondeterminist Polynomial-time)

Il existe 2 définitions équivalentes.

Définition 1 : Classe NP

La classe NP est l'ensemble des problèmes de décision résolus par une machine de Turing non-déterministe en temps polynomial en fonction de la taille d'entrée.

Comme pour les automates finis non-déterministes, pour une machine de Turing non-déterministe, il peut exister plusieurs transitions possibles à partir d'un même état et une même lecture.

	a	b	□
→1	b , →, 1 □ , →, 1	a , →, 1	accepté

Classe NP (Nondeterminist Polynomial-time)

Un vérifieur pour un problème A est une machine de Turing déterministe telle que pour toute instance w positive de A, il existe une réponse c tel que M accepte (w, c). c est appelé un certificat ou une solution.

Ex : Pour le problème SAT, il est possible de vérifier en temps polynomial qu'une valuation $c = (0, 1, \ldots)$ satisfasse une formule.

Définition 2 : Classe NP

La classe NP est l'ensemble des problèmes de décision A tels qu'il existe un vérifieur de A en temps polynomial.

Classe NP (Nondeterminist Polynomial-time)

Définition 2 : Classe NP

La classe NP est l'ensemble des problèmes de décision A tels qu'il existe un vérifieur de A en temps polynomial.

Les définitions 1 et 2 sont équivalentes.

idée de la preuve : Définir une MT non-déterministe à partir d'un vérifieur

Algorithme M_NonDeterminst(w) : bool
début
  Choisir une solution c
  Simuler le vérifieur M sur w avec c
  si M accepte **alors**
    retourner Accepter
  sinon
    retourner Refuser
  fin si
fin

Point d'étape

Les problèmes P se résolvent avec un algorithme qui s'exécute en temps polynomial.
Intuitivement, les problèmes NP sont compris comme des problèmes qui se résolvent, a priori, en temps exponentiel. L'une des raisons est que le nombre de solutions candidates est exponentiel.

$$NP = P ?$$

Est-il possible de réduire un problème NP en un problème P ?

Nous allons voir une réduction possible de problème NP

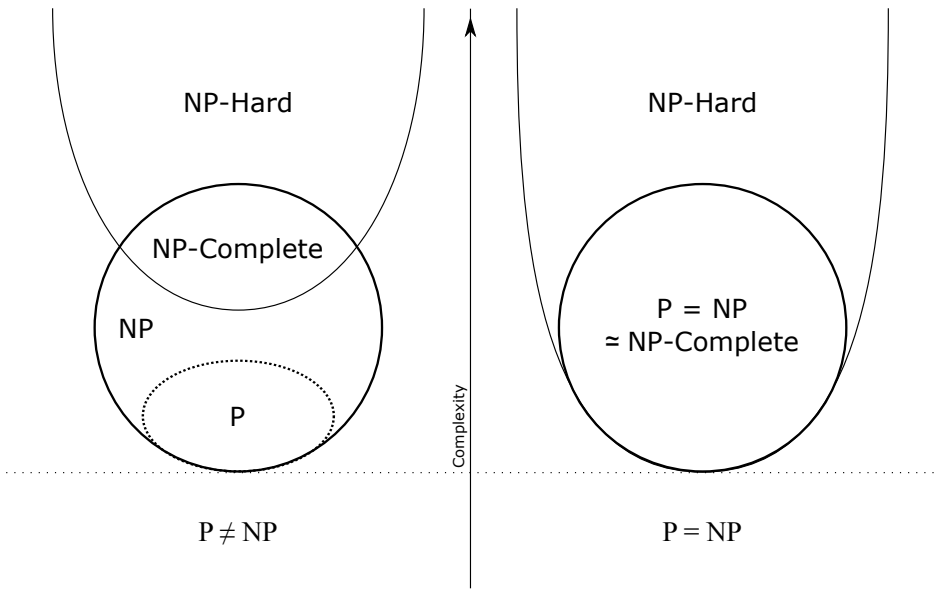
NP-difficile et NP-complet

Définition : NP-difficile

Un problème A est dit NP-difficile (NP-hard) si pour tout problème B de la classe NP, il existe une réduction en temps polynomial de B vers A

Définition : NP-complet (Cook, 1971)

Un problème A est dit NP-complet si le problème A est un NP-difficile et qu'en plus il est NP.



Existe-t-il un problème NP-complet ?

Théorème de Cook-Levin, 1971

La problème SAT est NP-complet.

Preuve : voir par exemple
`https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Cook`

Conséquence

Cela signifie que tout problème NP peut réduire en un problème SAT. Si nous pouvons résoudre en temps polynomial le problème SAT, alors il est possible de résoudre en temps polynomial n'importe quel problème NP.

Par des réductions, de nombreux problèmes ont été montré comme étant NP-complet. Voir :

D.S. Johnson M.R. Garey. Computers and Intractability : A Guide to the Theory of NP- Completeness. W. H. Freeman & Co., 1979.

Conclusion

Le problème SAT, la satisfaction d'une formule booléenne, permet de traduire et de vérifier formellement l'état d'un système.

Nombreuses applications en correction (model checking) :
machine, programme, système biologique, etc.

Même s'il existe de nombreux algorithmes de résolution efficaces (solveurs SAT), le problème SAT n'est pas simple à résoudre.

SAT est NP (si P $\neq$ NP, pas d'algo. polynomial pour le résoudre)
SAT est NP-complet : "prototype" du problème NP qui permet résoudre tous les autres.

Perspectives

$$NP = P$$

est une question ouverte,
à au moins, 1 million de dollars...

La résolution du problème SAT est d'importance, et de nombreuses recherches sont menées.

L'un des espoirs dans les machines quantiques et de l'informatique quantique est de pouvoir résoudre cette classe de problèmes NP (spin glasses).