Aucune application (DPLL)

Énoncer la remarque : syntaxe ≠ sémantique

916 Formules du calcul propositionnel : Représentation, Formes normales, Satisfiabilité, Applications

La logique propositionnelle est un outil utile pour modéliser des problèmes divers ou pour définir des logiques plus complexes.

I) Syntaxe

Déf 1 : On appelle ensemble de variables propositionnelles, noté V, un ensemble dénombrable.

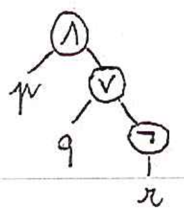
Déf 2 : On définit l'ensemble des formules propositionnelles sur V, noté F, par induction :
- $V \in F$.
- Si $f \in F$, alors $\neg f \in F$.
- Si $f, g \in F$, alors $(f \vee g) \in F$.

Prop 3 : La définition des formules propositionnelles est non-ambiguë.

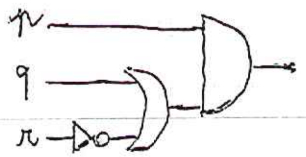
Notation 4 : On définit de nouveaux opérateurs comme abréviation des opérateurs précédents :
- $(f \wedge g) := \neg(\neg f \vee \neg g)$
- $(f \Rightarrow g) := (\neg f \vee g)$
- $(f \Leftrightarrow g) := ((f \Rightarrow g) \wedge (g \Rightarrow f))$

Notation 5 : On réduit le nombre de parenthèse en considérant $\wedge$ et $\vee$ prioritaire à gauche et $\Rightarrow$ à droite.

Autres représentations d'une formule :
- Formule classique : $p \wedge (q \vee \neg r)$
- Ecriture polonaise : $\wedge\, p \vee q \neg r$
- Arbre :



- Circuit logique :



II) Sémantique

Déf 6 : Une valuation est une application de V dans $\{0,1\}$. Cette application est alors étendue aux formule de F par induction :
- Pour tout $f \in F$, $v(\neg f) = 1$ ssi $v(f) = 0$.
- Pour tout $f, g \in F$, $v(f \vee g) = 1$ ssi $v(f) = 1$ ou $v(g) = 1$

Ex 7 : Soit $v : x \mapsto 1$ ssi $x = p$.
Ainsi $v(p \wedge \neg q) = 1$ et $v(p \Rightarrow q) = 0$.

Déf 8 : Soient f et g deux formules propositionnelles et v une valuation :
- v satisfait f ssi $v(f) = 1$.
- f est satisfiable ssi il existe une valuation qui satisfait f.
- f est valide ssi toute valuation satisfait f.
- f et g sont équivalentes, noté $f \equiv g$, ssi elles sont satisfaite par les mêmes valuations

Prop 9 : Une formule $f \in F$ est valide ssi $\neg f$ est insatisfiable.

Prop 10 : Deux formules f et g sont équivalentes

Déf 11 : Soient Γ un ensemble de formules et f une formule :
- Γ est satisfiable ssi il existe une valuation qui satisfait toute les formules de Γ.
- Γ est finiement satisfiable ssi tout sous-ensemble fini de Γ est satisfiable
- f est une conséquence de Γ ssi toute valuation qui satisfait Γ satisfait f.

Prop 12 : Un ensemble fini de la forme $\{B_i \mid 1 \leqslant i \leqslant n\}$ est satisfiable ssi la formule $\bigwedge_{i=1}^{n} B_i$ est satisfiable

Prop 13 : $\Gamma \models B$ ssi $\Gamma \cup \{\neg B\}$ est insatisfiable.

Thm 14 : <u>Théorème de compacité</u>

Soit Γ un ensemble de formules insatisfiable.

Il existe un sous-ensemble fini de Γ insatisfiable

<u>Classe de formules et leur représentation :</u>

On peut quotienter les formules par la relation $\equiv$.

On cherche alors à étudier les classes d'équivalence ainsi créées et non les formules.

Prop 15 : Soit $\mathcal{F}[x_1, \dots, x_n]$ l'ensemble des formules ne contenant que les variables $x_1, \dots, x_n$.

$\mathcal{F}[x_1, \dots, x_n]/_{\equiv}$ est en bijection avec $\{0,1\}^n \to \{0,1\}$.

Nous allons chercher des méthodes pour représenter ~~un~~ élément de $\mathcal{F}/_{\equiv}$:

· <u>Table de vérité :</u>

p	q	B
0	0	0
0	1	1
1	0	1
1	1	0

Une table de vérité représente toute les valuations possibles

· <u>Formes normales :</u>

Un littéral est une variable ou sa négation.

Une forme normale conjonctive est une conjonction de disjonction de littéraux (CNF)

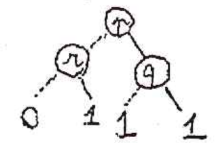
On définit de même les formes normales disjonctive (DNF).

Ex : $p \Leftrightarrow q \equiv (\neg p \vee q) \wedge (p \vee \neg q)$

$\equiv (p \wedge q) \vee (\neg p \wedge \neg q)$.

· <u>Arbre binaire de décision (ABD) :</u>

Arbre où chaque noeud contient une variable et a deux fils, et où les feuilles sont étiquettées par 0 ou 1.



· <u>Diagramme binaire de décision (BDD) :</u>

Représentation plus compacte d'un ABD.

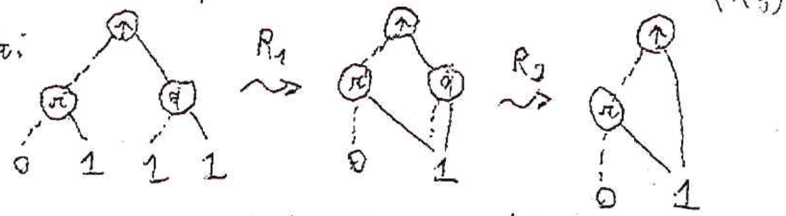
Un BDD est réduit si l'on ne peut appliquer aucune des réductions suivantes :

· Fusionner les feuilles de mêmes étiquettes. (R_1)

· Supprimer les noeuds dont les deux branches se dirigent vers le même noeud. (R_2)

· Fusionner deux noeud dont les sous-diagramme sont identique (R_3)

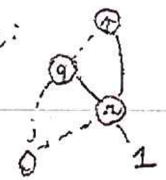
Ex :



· <u>Diagramme binaire de décision ordonné (OBDD) :</u>

On se donne un ordre sur nos variables et on considère les BDD respectant cet ordre dans tous ces chemins

Ex :



est un OBDD pour l'ordre $p > q > r$.

pour quoi faire ?

Thm 16 : Pour un ordre donné, il existe un unique OBDD représentant une formule donnée

Application : • Permet de décider la validité et la satisfiabilité en temps constant

• Représentation usant peu de mémoire.

• Décide de l'équivalence de deux formules en temps linéaire :

III] SAT

Déf 17 : Le problème SAT est le suivant :

Etant donné une formule CNF, décider la satisfiabilité de cette formule.

Théorème de Cook :

SAT est NP-complet (DEV)

Application : On peut résoudre n'importe quel problème NP grâce à un algorithme qui décide SAT.

Ex : k-coloration d'un graphe.

Application : On peut décider de la satisfiabilité de n'importe quelle formule à l'aide d'un algorithme qui décide SAT. Pour passer la forme en CNF, deux méthodes sont possibles :

• Trouver une CNF équivalente : le résultat est de taille exponentielle

• Transformation de Tseitin (DVPT).

Écrire 1 fct, pour un algo

Permet de mettre toute formule sous forme normale 3-SAT

Algorithme DPLL : → algo backtracking = essais successifs = recherche exhaustive

Algo_DPLL (Γ : ensemble de clauses) : bool =

Supprimer les clauses valide de Γ :

Si $\Gamma = \emptyset$ retourner vrai

Sinon retourner DPLL(Γ)

DPLL (Γ : ensemble de clauses non-valides) : bool =

Si $\bot \in \Gamma$ retourne faux

Si $\Gamma = \emptyset$ retourne vrai

Si Γ contient des littéraux isolés

| Enlever de Γ les clauses contenant des littéraux isolés
| DPLL(Γ)

Sinon si Γ contient des clauses unitaires

| Appliquer la résolution unitaire à Γ
| DPLL(Γ)

Sinon

| Choisir x une variable de Γ
| retourner DPLL($\Gamma[x:=0]$) ou DPLL($\Gamma[x:=1]$)

Référence :

Majoritairement : Devismes, Lafourcade
Logique et démonstration automatique

Cook : Carton

Tseitin : Stern.
Page 155.

BDD : Huth, Ryan
Logic in computer science