21/10/2014

916

Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications.

Motivation : formalisation de la logique.

I Syntaxe.

1) Définition des formules.

Comment est caractérisé le langage propositionnel ?

Déf 1 [LR, p8] :
Il est caractérisé par un alphabet $\mathcal{A}$, constitué de :
- un ensemble fini de symboles : $\mathcal{P} = \{p, q, r, \ldots\}$, appelés variables propositionnelles ;
- un ensemble de connecteurs :
  $\neg$ (non), $\wedge$ (et), $\vee$ (ou),
  $\Rightarrow$ (implication), $\Leftrightarrow$ (équivalence) ;
- les parenthèses (et).

$\neg$ est un connecteur unaire ; les autres sont des connecteurs binaires.

Déf 2 :
Un mot (ou expression) est une suite finie de symboles de $\mathcal{A}$. L'ensemble des mots est noté $\mathcal{A}^*$.

Exemples : $\neg p$, $(p \vee (q \Rightarrow r))$ et $(p(\vee \neg \vee)$ sont des mots. Seuls certains sont intéressants pour la logique.

Déf 3 :
L'ensemble des formules construit sur $\mathcal{P}$ est le plus petit ensemble $\mathcal{F}$ tel que :
- $\forall x \in \mathcal{P}$, alors $x \in \mathcal{F}$.
- Si $F \in \mathcal{F}$ alors $\neg F \in \mathcal{F}$.
- Si α est un connecteur binaire, si $F \in \mathcal{F}$ et $G \in \mathcal{F}$, alors $(F \alpha G) \in \mathcal{F}$.

Cet ensemble est bien défini ; l'ensemble des mots $\mathcal{A}^*$ respecte ces conditions, et $\mathcal{F}$ est non vide car contient $\mathcal{P}$.
On peut aussi définir cet ensemble par récurrence :

Déf 4 :
Les ensembles $\mathcal{F}_n$ sont définis par :
- $\mathcal{F}_0 = \mathcal{P}$
- $\mathcal{F}_{n+1} = \mathcal{F}_n \cup \{\neg F \,;\, F \in \mathcal{F}_n\} \cup \{(F \alpha G) \,;\, F \in \mathcal{F}_n,\ G \in \mathcal{F}_n\}$

Propo 5 :
La suite $\mathcal{F}_n$ est croissante, et on a : $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{F}_n$.

Déf 6 [CL, p20]
La hauteur d'une formule F est le plus petit $n \in \mathbb{N}$, tel que $F \in \mathcal{F}_n$. On la note $h(F)$.

2) Quelques propriétés issues de la définition inductive.

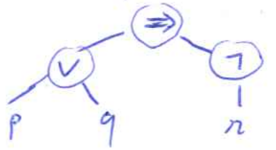
Théorème 7 (de non-ambiguïté) [CL, p27] :
Si F est une formule, alors elle s'écrit de manière unique sous la forme :
- d'une variable propositionnelle, ou ;
- $F = \neg G$, où $G \in \mathcal{F}$, ou :
- $(G \alpha H)$, où $\alpha \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$ et $G \in \mathcal{F}$ et $H \in \mathcal{F}$.

Remarque : lorsque les parenthèses sont superflues, on s'autorisera à les ôter. $(p \vee (q \vee r))$ deviendra $p \vee (q \vee r)$.

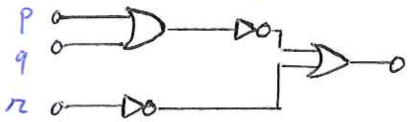
Déf 8 [LR, p11] :
Un arbre est un ensemble ordonné satisfaisant :
- il possède un plus petit élément, appelé racine de l'arbre.
- l'ensemble des minorants de chaque élément est totalement ordonné.

Exemple : $F = ((p \vee q) \Rightarrow \neg r)$

Arbre représentant F :



Circuit logique



Déf 9 [CL, p 29] :

L'ensemble des sous formules de $F \in \mathcal{F}$ peut se définir par induction : - si $F = p \in \mathcal{P}$, $sf(F) = \{p\}$.

- si $F = \neg G, G \in \mathcal{F}$: $sf(F) = \{F\} \cup sf(G)$.

- si $F = (G \alpha H)$, $sf(F) = \{F\} \cup sf(G) \cup sf(H)$.

Déf 10 (substitution) [LR, p 17] :

La substitution de G à p dans F, notée $F(G/_p)$, est définie par induction sur F :

- si $F = p$, alors $F(G/_p) = G$; si $F = q \neq p$, $F(G/_p) = q$.
- si $F = \neg H$, alors $F(G/_p) = \neg H(G/_p)$.
- si $F = F_1 \alpha F_2$, alors $F(G/_p) = F_1(G/_p) \alpha F_2(G/_p)$.

Remarque : On ne fait qu'une substitution à la fois, il est possible de définir plusieurs substitutions simultanées

Exemple : $F = ((p \vee q) \Rightarrow r)$ et $G = (r \vee q)$.

Alors $F(G/_p) = (((r \vee q) \vee q) \Rightarrow r)$.

II Sémantique.

Permet d'interpréter les formules précédemment définies.

1) Valuations.

Déf 11 [LR, p 13] :

Une valuation $v : \mathcal{P} \to \{0,1\}$ associe à chaque variable une valeur. Elle possède un unique prolongement à $\mathcal{F}$, noté v, par les règles usuelles : $v(\neg F) = \neg v(F)$, $v(F \alpha G) = v(F) \alpha v(G)$.

Soit v une valuation. A partir des valeurs associées aux variables, on peut construire les tables de vérité pour identifier la valuation d'une formule F.

Exemple : $F = p \alpha q$ (cf annexe).

On identifie plus facilement le point de vue "circuit électrique" : une variable ayant une valuation de 1 correspond à une entrée positive, le courant passe dans cette entrée.

2) Notions fondamentales et propriétés immédiates.

Déf 12 [LR, p 15]

Une formule F est satisfaite par la valuation v si $v(F) = 1$. F est dite satisfaisable.

Une tautologie est une formule qui est satisfaite par toute valuation. Une antilogie n'est satisfaite par aucune.

Deux formules F et G sont équivalentes si, pour toute valuation v, $v(F) = v(G)$.

Exemple : $(p \Rightarrow (q \Rightarrow p))$ est une tautologie.

$((\neg p \Rightarrow q) \Rightarrow (\neg p \Rightarrow \neg q))$ aussi.

$\neg\neg p$ et p sont des formules équivalentes.

- On tire des tautologies les règles de De Morgan :

$\neg(F \vee G) \Leftrightarrow (\neg F \wedge \neg G)$, $\neg(F \wedge G) \Leftrightarrow \neg F \vee \neg G$.

- On n'a pas la même signification dans le langage courant : [CL, p 34]

F = si tu as faim, il y a des épinards au frigo.

G = S'il n'y a pas d'épinard dans le frigo, tu n'as pas faim.

Déf 13 [LR, p 16]

Soit Σ un ensemble de formules, et F une formule.

- Σ est satisfaisable s'il existe v valuation telle que : $\forall G \in \Sigma, v(G) = 1$
- F est conséquence de Σ si toute valuation satisfaisant Σ satisfait F.

Exemple : F est conséquence de Σ si et seulement si $\Sigma \cup \{\neg F\}$ n'est pas satisfaisable.

Déf 14 [LR, p 17] :
Une occurence de la variable p dans la formule F est la donnée de cette variable et d'une place où elle apparaît dans F.

Prop 15 :
La valuation d'une formule F ne va dépendre que de la valuation des variables ayant une occurence dans F.

3) Théorèmes.

Problème : On veut démontrer la satisfiabilité de certaines formules ; on dénote ce problème SAT.

Th 16 (de Cook) [WOL, p 185] : SAT est NP-complet. **DVT**

Déf 17 [LR, p44] :
- Un ensemble de formules T est *finiment satisfaisable* si tout sous-ensemble fini de T est satisfaisable.
- T, finiment satisfaible, est maximal si pour toute formule F, $F \in T$ ou $\neg F \in T$.

Th 18 (de compacité) [CL, p61] **DVT**
Un ensemble de formules T est satisfaisable si et seulement si T est finiment satisfaisable.

III Disjonctions, conjonctions et déduction.

1) Formes normales disjonctives et conjonctives.

Déf 19 :
Un *système complet de connecteurs* est un sous-ensemble de $\{\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow\}$ qui permet, à chaque formule, de trouver une formule équivalente ayant uniquement les connecteurs du sous-ensemble.

Exemple : Avec $\{\neg, \wedge\}$, on a : $p \vee q \equiv \neg(\neg p \wedge \neg q)$
et $p \Rightarrow q \equiv \neg p \vee q$.

Déf 20 [LR, p21]
Une *forme normale disjonctive* est une formule F de la forme :
- ou bien $F_1 \vee \ldots \vee F_k$, où $\forall i$, $F_i = G_1 \wedge \ldots \wedge G_{l(i)}$ où chaque G_j est une variable ou sa négation.
- ou bien un F_i précédemment défini.

Les formes normales conjonctives sont, elles, des conjonctions de disjonctions de variables propositionnelles ou de leur négation.

Remarque : $\{\vee, \neg, \wedge\}$ est un système complet de connecteurs.

Th (de forme normale) 21 [LR, p23]
Toute formule est logiquement équivalente à au moins une formule sous FNC.

En pratique : tables de vérité ou utiliser l'élimination de connecteurs.

2) Clauses et règles de coupure.

Déf 22 [LR, p26] :
$C = (G_1 \vee G_2 \vee \ldots \vee G_l)$ est une *clause* si $\forall i$, G_i est une variable ou la négation d'une variable.
$\Delta := \{G_i \mid G_i \text{ est sans négation}\}$. $\Gamma := \{\neg G_i \mid G_i = \neg p, \text{ où } p \text{ une variable}\}$.

Exemple : $F = p \vee \neg q \vee r$.
$\Delta = \{p ; r\}$ $\qquad \Gamma = \{q\}$.

Déf 23 : Règle de coupure [LR, p31].
Soient $C_1 = (\Gamma_1, \Delta_1)$ et $(\Gamma_2, \Delta_2) = C_2$ deux clauses. Si $p \in \Gamma_2 \cap \Delta_1$, alors on peut déduire de C_1, C_2 la formule $C = (\Gamma, \Delta)$, où $\Gamma = \Gamma_1 \cup (\Gamma_2 \setminus \{p\})$, $\Delta = \Delta_2 \cup (\Delta_1 \setminus \{p\})$

Exemple d'utilisation : abstraction de contraintes propositionnelles.
On notera : $\dfrac{C_1 \quad C_2}{C}$.

Exemple : $C_1 = p \vee q \vee \neg r$
$C_2 = q \vee r$
$$\frac{C_1 \quad C_2}{p \vee q},$$

ANNEXE:

$F = p \alpha q.$

p	q	$\alpha = \wedge$	$\alpha = \vee$	$\alpha = \Rightarrow$	$\alpha = \Leftrightarrow$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Réf :

[LR] : Lassaigne - Rougemont,
Logique et fondements de l'informatique.

[CL] : Cori Lascar,
Logique mathématique, tome 1.

[WOL] : Wolper,
Introduction à la calculabilité.

Théorème de Compacité du calcul propositionnel

Soit $\mathcal{A}$ un ensemble de formules du calcul propositionnel. Alors $\mathcal{A}$ est satisfiable $\Longleftrightarrow$ $\mathcal{A}$ est finiment satisfiable

<u>preuve</u> : On démontre le résultat pour un ensemble dénombrable de variables propositionnelles. Le même résultat existe dans le cas non dénombrable,

On démontre le sens réciproque, l'implication étant évidente

- Soit $P = (\alpha_i)_{i \in \mathbb{N}}$ l'ensemble des variables propositionnelles.

On construit par récurrence une suite $(\varepsilon_n) \in \{vrai; faux\}^{\mathbb{N}}$ tel que ε_n est la valuation de α_n, pour tout $n \in \mathbb{N}$.

<u>Cas 0_0</u> : $\forall B \subset \mathcal{A}$, B finie, il existe une valuation $\delta \in \{vrai; faux\}^P$ satisfaisant B et telle que $\delta(\alpha_0) = faux$.

on pose alors $\varepsilon_0 = faux$

<u>Cas 1_0</u> : $\exists B_0 \subset \mathcal{A}$, finie telle que pour toute valuation $\delta \in \{vrai; faux\}^P$, satisfaisant B_0, on ai $\delta(\alpha_0) = vrai$

On pose alors $\varepsilon_0 = vrai$.

<u>Lemme</u> : $\forall B \subset \mathcal{A}$, finie, il existe une valuation $\delta \in \{vrai; faux\}^P$ telle que $\delta(\alpha_0) = \varepsilon_0$.

<u>preuve</u> : . Si $\varepsilon_0 = faux$, c'est évident

. Si $\varepsilon_0 = vrai$, $B' = B \cup B_0$; il existe δ satisfaisant B' donc δ satisfait B_0, on a forcément $\delta(\alpha_0) = vrai = \varepsilon_0$. et δ satisfait B. $\square$

<u>Hypothese de Récurrence</u>

<u>R_n:</u> $\forall B \subset \mathcal{A}$, finie, il existe au moins une distribution $\delta \in \{vrai; faux\}^P$ satisfaisant B et telle que $\delta(\alpha_i) = \varepsilon_i$, $i \leq n$.

On veut montrer que $R_n \Rightarrow R_{n+1}$.

<u>Cas 0_{n+1}</u> : $\forall B \subset A$, finie, il existe une valuation $\delta \in \{vrai, faux\}^P$ telle que $\delta(\alpha_i) = \varepsilon_i$ pour $i \leq n$ (R_n)
et $\delta(\alpha_{n+1}) = faux$.

On pose alors $\varepsilon_{n+1} = faux$

<u>Cas 1_{n+1}</u> : $\exists B_{n+1} \subset A$, finie telle que pour toute valuation $\delta \in \{vrai, faux\}^P$
(δ satisfait B_{n+1} et $\delta(\alpha_i) = \varepsilon_i$ pour tout $i \leq n$)
$\Longrightarrow$ ($\delta(\alpha_{n+1}) = vrai$)

On pose alors $\varepsilon_{n+1} = vrai$

$\Longrightarrow$ $\boxed{R_{n+1}}$: $\forall B \subset A$, finie, il existe une valuation $\delta \in \{vrai, faux\}^P$ telle que $\delta(\alpha_i) = \varepsilon_i$, $i \leq n+1$

Si $\varepsilon_{n+1} = faux$, ça vient de la définition

Si $\varepsilon_{n+1} = vrai$, on pose $B' = B \cup B_{n+1}$.
alors par l'hypothèse de récurrence, il existe δ satisfaisant B' et telle que $\delta(\alpha_i) = \varepsilon_i$, $i \leq n$. δ satisfait B_{n+1} donc $\delta(\alpha_{n+1}) = \varepsilon_{n+1}$.

On pose alors δ_0 définie par $\delta_0(\alpha_i) = \varepsilon_i$, $i \in \mathbb{N}$.

δ_0 satisfait A : Soit $F \in A$, alors $\exists n \in \mathbb{N}$, $F = F(\alpha_1 \dots \alpha_n)$
Par R_n, F est satisfaite par δ.

A est satisfaite par δ.

<u>Référence</u> : Cori Lascar,
Logique mathématique p 62.

Théorème de Cook

Reference :

[WOL], p185 *****

[CAR], p189 *

Définition : Langage NP

Les langages NP sont les langages reconnus par une machine de Turing non déterministe polynomialement bornée.

Théorème de Cook.

Le problème SAT est NP-complet.

Preuve : • Le problème SAT est NP : on peut écrire un algorithme qui décide de manière non déterministe une valuation, et teste si elle satisfait une formule du calcul propositionnel en temps linéaire.

• Le problème SAT est NP-dur ; On réduit l'exécution d'une machine de Turing à la satisfiabilité d'une forme normale, et ce en temps et en espace mémoire polynomiaux.

Soit $M = \langle Q, \Sigma, T, \Delta, s, B, F \rangle$, borné par $p(n)$, n est la taille de l'entrée. Une exécution de la machine est représentée dans le tableau suivant ;

Q	P	C	R
q_0	0	c_0	w
q_f	k		

$p(n)+1$ (lignes)

$p(n)+1$ (colonnes de R)

$Q[i] \in Q$, état de l'automate au temps i

$P[i] \in [\![0, p(n)]\!]$ est la position de la tête de lecture

$C[i] \in [\![1, r]\!]$ est le choix effectué au temps i

$R[i,j] \in T$ est la j-ème lettre du ruban.

On utilise les variables propositionnelles suivantes

$q_{i,a}$: "$Q[i] = a$" $\qquad$ $c_{i,a}$: "$C[i] = a$"

$p_{i,a}$: "$P[i] = a$" $\qquad$ $r_{i,j,a}$: "$R[i,j] = a$"

On écrit la formule SAT en quatre parties $\varphi = \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3$

φ_0 : "Il y a un et un seul symbol par case"

$$\varphi_{0,R} = \bigwedge_{0 \le i,j \le p(n)} \left(\left(\bigvee_{a \in \Gamma} r_{i,j,a} \right) \wedge \bigwedge_{a \ne a' \in \Gamma} \left(\overline{r_{i,j,a}} \vee \overline{r_{i,j,a'}} \right) \right)$$

$$\varphi_{0,Q} = \bigwedge_{0 \le i \le p(n)} \left(\left(\bigvee_{a \in Q} q_{i,a} \right) \wedge \bigwedge_{a \ne a' \in Q} \left(\overline{q_{i,a}} \vee \overline{q_{i,a'}} \right) \right)$$

, $\varphi_{0,c}$, $\varphi_{0,p}$ sur le même modèle.

La taille de ces formules est inférieure à $\lambda \times \left(p(n)^2 \times |\Gamma|^2 + p(n) \times \left(|Q| + r + p(n)\right)\right)$.

φ_1 : "La configuration initiale est : état initial, mot w sur le ruban, tête de lecture en position 0"

$$q_{0,s} \wedge p_{0,0} \wedge \left(\left(\bigwedge_{0 \le j \le n-1} r_{0,j,w_{j+1}} \right) \wedge \left(\bigwedge_{n \le j \le p(n)} r_{0,j,B} \right) \right)$$

φ_2 : "On atteint un état final avant la fin, en temps $t \le p(n)$"

$$\bigvee_{\substack{0 \le i \le p(n) \\ k \in F}} q_{ik}$$

Ces deux formules sont de taille inférieure à $\mu \times p(n)$.

φ_3 : "Les passages d'une configuration à une autre suivent les transitions et le choix non déterministe effectué".

a) Aucune case sauf celle sous la tête de lecture n'est modifié.

$$\varphi_{3.1} \quad \bigwedge_{\substack{0 \le i \le p(n) \\ 0 \le j \le p(n)-1 \\ a \in \Gamma}} \left(r_{i,j,a} \wedge \overline{p_{ij}} \Rightarrow r_{i,j+1,a} \right)$$

b) La case sous la tête de lecture, l'état et la position de la tête de lecture sont modifiés dépendamment de Δ

$$\varphi_{3.2} \quad \bigwedge_{\substack{0 \le i \le p(n)-1 \\ 0 \le j \le p(n) \\ k,k' \in Q \\ \alpha,\alpha' \in \Gamma}} \ \bigwedge_{\substack{1 \le \ell \le r \\ k \xrightarrow{\alpha,\alpha',d,\ell} k' \in \Delta}} \left(\left(q_{i,k} \wedge p_{ij} \wedge r_{i,j,a} \wedge c_{i,\ell} \right) \Rightarrow \left(q_{i+1,k'} \wedge r_{i+1,j,a'} \wedge p_{(i+1)(j+d)} \right) \right)$$