916 — Formules du calcul propositionnel : représentations, formes normales, satisfiabilité. Applications

1/3

Motivations – Formaliser les raisonnements
– Création de circuits logiques

I) Syntaxe : Ce que l'on s'autorise à écrire

1) Définition

Def 1 : P non vide, fini ou dénombrable.
L'ensemble des formules propositionnelles sur P est $\mathcal{F}$, défini par induction sur $P \cup \{\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, (,)\}$:
(B) : $P \subset \mathcal{F}$ [coin non] [flèche implique "...]
(I) ⊛ si $x \in \mathcal{F}$, $\neg x \in \mathcal{F}$
⊛ si x et $y \in \mathcal{F}$, alors $(x \wedge y)$, $(x \vee y)$, $(x \Rightarrow y)$, $(x \Leftrightarrow y)$ sont dans $\mathcal{F}$

Ex 2 : $P = \{a, b, c, d\}$. $(a \Rightarrow (b \Leftrightarrow a))$; $\neg\neg\neg\neg d$

Thm 3 : (Lecture unique). Soit $F \in \mathcal{F}$. Alors, un seul des trois cas suivants se présente :
1) $F \in P$ 2) il existe une unique $G \in \mathcal{F}$ tq $F = \neg G$
3) il existe des uniques $G, H \in \mathcal{F}$, $\alpha \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$, tq $F = (G \alpha H)$

DEV 1

2) Représentations

Thm de lecture unique ⇒ non-ambiguïté des formules,
→ Permet plusieurs représentation des formules.

⊛ Représentation par arbre
Voir annexe

⊛ Représentation préfixe :
Correspond à un parcours préfixe de l'arbre
→ $\Rightarrow \wedge \vee \neg P_0 P_1 P_2 \wedge \neg\neg P_0 \neg P_1$
Gain de place, de mémoire

⊛ Description ascendante
$\mathcal{F}_0 = P$; pour $n \geq 0$: $\mathcal{F}_{n+1} = \mathcal{F}_n \cup \{\neg F, F \in \mathcal{F}_n\} \cup \{F \alpha G \mid F, G \in \mathcal{F}_n\}$
Alors $\mathcal{F} = \bigcup_{\mathbb{N}} \mathcal{F}_n$

Def 4 : $h(F) = \min\{n \mid F \in \mathcal{F}_n\}$ est la hauteur de F.
C'est aussi la hauteur de l'arbre associé à F

3) Substitutions

Permet la création simple de formules.

Def 5 : $F, G_1, \dots, G_k \in \mathcal{F}$ et $A_1, \dots, A_k \in P$.
On définit par induction $F[G_1/A_1, \dots, G_k/A_k]$:
(B) si $F \in P$, $F[G_1/A_1, \dots, G_k/A_k] = \begin{cases} G_k & \text{si } F = A_k \\ F & \text{sinon} \end{cases}$
(I) si $F = \neg G$, $F[G_1/A_1 \dots G_k/A_k] = \neg G[G_1/A_1 \dots G_k/A_k]$
si $F = (G \alpha H)$, $F[G_1/A_1 \dots G_k/A_k] = (G[G_1/A_1 \dots G_k/A_k] \alpha H[G_1/A_1 \dots G_k/A_k])$

⚠ Ce sont des substitutions simultanées.
$F = (A \Rightarrow B)$; $(F[B/A])[A/B] = B \Rightarrow B$
$F[B/A, A/B] = B \Rightarrow A$.

II) Sémantique : Donner du sens aux formules

1) Valuations

On commence par interpréter les éléments de P.

Def 6 : Une valuation est une application de P dans $\{0, 1\}$
→ Cela induit une valuation sur $\mathcal{F}$, qui est la notion la plus importante.

+ atome taille d'une formule

La valeur d'une formule ne dépend que de la valuation de ses atomes.

Thm 7: Toute valuation δ se prolonge de manière unique en une $\bar{\delta} : \mathcal{F} \to \{0,1\}$ telle que :

* $\bar{\delta}|_P = \delta$
* pour $F \in \mathcal{F}$, $\bar{\delta}(\neg F) = 1$ ssi $\bar{\delta}(F) = 0$
* pour F et $G \in \mathcal{F}$, $\bar{\delta}(F \wedge G)$ est défini par les tables de vérités :

$\bar{\delta}(F)$	$\bar{\delta}(G)$	$\bar{\delta}(F \wedge G)$	$\bar{\delta}(F \vee G)$	$\bar{\delta}(F \Rightarrow G)$	$\bar{\delta}(F \Leftrightarrow G)$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

2) Satisfaction

→ On s'intéresse à la possibilité qu'une formule soit "vraie".

Def 8: On dit que δ satisfait F lorsque $\bar{\delta}(F) = 1$.

Def 9: Soit $\mathcal{A} \subset \mathcal{F}$.
- $\mathcal{A}$ est satisfait par δ si tous ses éléments le sont
- $\mathcal{A}$ est satisfiable (consistant) si il existe δ qui satisfait $\mathcal{A}$
- $\mathcal{A}$ est contradictoire sinon

Def 10: Une tautologie est une formule satisfaite par toute valuation. ex : $((A \Rightarrow A) \Leftrightarrow (B \Rightarrow B))$
notation : $\models A$
si $\neg A$ est une tautologie, A est une antilogie ($\not\models A$)

Def 11: On dit que B est une conséquence sémantique de Δ si toute valuation satisfaisant Δ satisfait B. (noté $\Delta \models B$)
~~C'est à dire si $\models (A \Rightarrow B)$~~. Δ ens de formules
si $A \models B$ et $B \models A$, on parle d'équivalence sémantique

Exemples : $(A \Rightarrow B) \wedge (B \Rightarrow C) \models (A \Rightarrow C)$
$(A \wedge (\neg A \vee B))$ et $\neg(A \Rightarrow \neg B)$ sont équivalentes

Propriétés 12:
- $\mathcal{A} \models G$ ssi $\mathcal{A} \cup \{\neg G\}$ contradictoire → complétude
- $\mathcal{A} \cup \{G\} \models H$ ssi $\mathcal{A} \models (G \Rightarrow H)$
- si $\mathcal{B} \subset \mathcal{A}$ satisfiable, alors $\mathcal{B}$ satisfiable.

3) Compacité du calcul propositionnel

On va voir ici les limites du calcul propositionnel) hdf

Def 13: Un ensemble de formules $\mathcal{A}$ est finiment satisfiable si tout ensemble fini inclus dans $\mathcal{A}$ est satisfiable

Thm 14: (de compacité)
Pour tout $\mathcal{A} \subset \mathcal{F}$, on a
$\mathcal{A}$ satisfiable ssi $\mathcal{A}$ finiment satisfiable.

Les implications sur le calcul propositionnel apparaissent alors dans le corollaire :

Cor 15: pour $\mathcal{A} \subset \mathcal{F}$, $F \in \mathcal{F}$.
$\mathcal{A} \models F$ ssi $A \models F$ avec A finie $\subset \mathcal{A}$.

III) Des formes normales pour la pratique

1) Formes normales conjonctives et disjonctives

Def 16: $F \in \mathcal{F}$ est sous forme normale disjonctive si

$$F = \bigvee_{1 \le i \le m} (\varepsilon_{i,1} B_{i,1} \wedge \dots \wedge \varepsilon_{i,k_i} B_{i,k_i}) \quad \text{avec } m, k_1, \dots, k_m \in \mathbb{N}^* \quad (\text{FND})$$
$$B_{i,1} \dots B_{i,k} \in P \qquad \varepsilon_{i,1} \dots \varepsilon_{i,k_i} \in \{\emptyset, \neg\}$$

) marche définir littéral

elle est sous forme normale conjonctive si

$$F = \bigwedge_{1 \le i \le m} (\varepsilon_{i,1} B_{i,1} \vee \dots \vee \varepsilon_{i,k_i} B_{i,k_i}) \quad (\text{FNC})$$

Exemples : pour $P_0 \in P$, P_0 est sous FNC et FND.
$\neg P_0 \vee P_1 \vee P_2$ est sous FNC et FND
$(P_0 \wedge \neg P_1) \vee (P_2 \wedge P_1 \wedge P_0 \wedge \neg P_2)$ est sous FND

Tout l'intérêt de ces formes normales est assuré par:

Thm 17: Toute formule de $\mathcal{F}$ est sémantiquement équivalente à au moins une formule sous FNC, et au moins une formule sous FND.

Exemples:

Toute tautologie est sémantiquement équivalente à $A \vee \neg A$ (avec $A \in \mathcal{F}$).

$(A \Rightarrow B)$ est équivalente à $\neg A \vee B$

Def 18: Un ensemble système de connecteurs est dit complet s'il engendre toutes les valuations

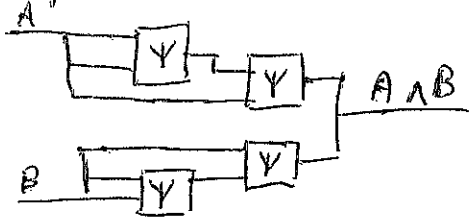
→ On vient donc de voir que $\{\neg, \vee, \wedge\}$ est complet.

Cependant, $\{\neg, \vee\}$ et $\{\neg, \wedge\}$ sont également.

Application: Le connecteur $\uparrow$ défini comme $A \uparrow B = \neg(A \vee B)$ forme un système complet.

Cela permet de réaliser des circuits logiques en utilisant un seul composant judicieusement répété.

Exemple: réalisation de $\wedge$ à partir de $\uparrow$



2) Utilisation du calcul propositionnel en théorie de la complexité

Def 19: SAT est un problème de décision prenant en entrée une formule propositionnelle et renvoyant Vrai si elle est satisfiable, Faux sinon

Ce problème est à la base de la NP-Complétude pour deux raisons:

Thm 20: (de Cook) SAT est NP-Complet

Et aussi car toutes les preuves de NP-Complétude se font par réductions successives à SAT.

Cependant, on utilise généralement une forme restreinte de SAT, aussi puissante, mais plus maniable

Def 21: 3-SAT est la restriction de SAT aux FNC où tous les k_i valent 3

Thm 22: 3-SAT est NP-Complet DEV 2

Applications:

3-Coloriage est NP-Complet

Ham-Path est NP-Complet

Clique est NP-Complet -

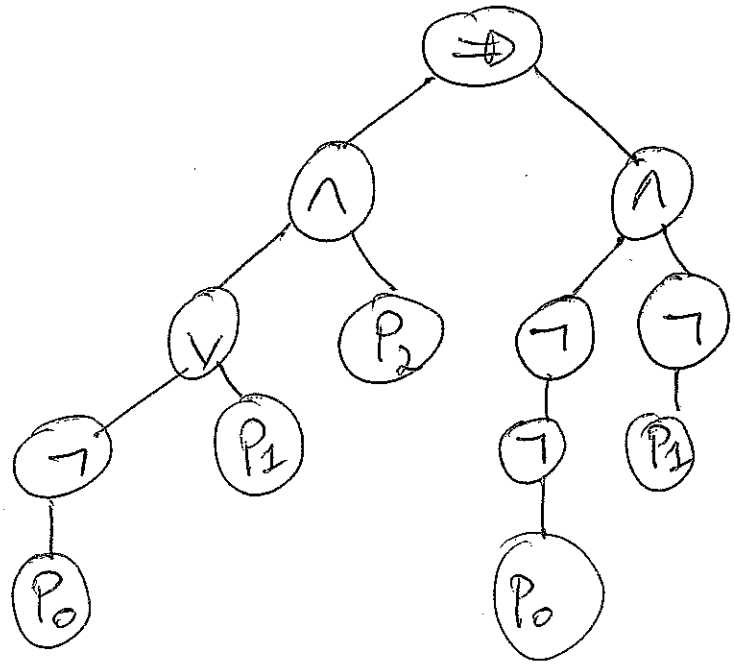
Prolog Preuve par coupure (pas indispensable mais bon)

Représentation par arbre des formules logiques propositionnelles

Formule : $P = \{P_0, P_1, \dots\}$.

$$(((\neg P_0 \vee P_1) \wedge P_2) \Rightarrow (\neg\neg P_0 \wedge \neg P_1))$$

Arbre :



Références ⊛ Principalement Cori & Lascar. (I, II, III.1) –
Attention, parfois des notations bizarres ! (⊨ est noté ⊢*)

- ⊛ Stern
- ⊛ Arnold – Gueissarion (bonnes notations)

} Deux bouquins de "maths pour l'info" très différents.

- ⊛ Carton pour la partie III.2.

Autres possibilités :
- Tables de Karnaugh
- Déduction

Autres devs :
- Thm de Cook
- Algo de déduction par coupure
- Additionneur n-bits

Annexe

3SAT est NP-complet.

REF: Carton

NP : même algorithme polynomial non-déterministe que pour SAT.

NP-DUR : Par réduction de SAT.

On se donne φ une formule et on considère son arbre syntaxique. Ses nœuds sont étiquetés par les variables et par les opérateurs $\neg$, $\wedge$, $\vee$.

On construit alors une formule φ' ayant pour variables :

- chaque variable de φ
- Une nouvelle variable pour chaque nœuds interne de l'ar

Ensuite, suivant qu'un nœud interne est étiqueté par $\neg$, $\wedge$ ou $\vee$, on lui associe l'égalité $x_i = \neg x_j$, $x_i = x_j \wedge x_k$ ou $x_i = x_j \vee x_k$ où x_i est la variable correspondant au nœuds et x_j et x_k celles correspondantes à ses fils.

φ' est alors la conjonction de ces égalités et de la clause x_a ou x_a est la variable associée à la racine.

On remplace enfin chaque égalité par équivalence logique :

$$x = \neg y \equiv (x \vee y) \wedge (\neg x \vee \neg y)$$

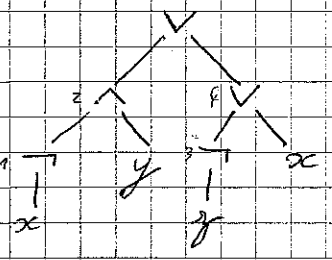
$$x = (y \wedge z) \equiv (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y) \wedge (\neg x \vee z)$$

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

Comme la taille de l'arbre est polynomialement bornée par la taille de φ, la taille de φ' l'est aussi.

Pour compléter le développement : (mmm ça ne fait pas 13 min)

- expliquer réduction de 3SAT à 3-coloration
- réduction de 3SAT à clique
- exemple de réduction d'une formule de SAT.



$x_1 = \neg x$

$x_2 = x_1 \wedge y$

$x_3 = \neg y$

$x_4 = x_3 \vee x$

$x_5 = x_2 \vee x_4$

$x_5 \wedge (x_5 = x_2 \vee x_4) \wedge (x_4 = x_3 \vee x) \wedge (x_3 = \neg y) \wedge (x_2 = x_1 \wedge y) \wedge (x_1 = \neg x)$

$x_5) \wedge (x_5 \vee \neg x_2) \wedge (x_5 \vee \neg x_4) \wedge (\neg x_5 \vee x_2 \vee x_4)$

$\wedge (x_4 \vee \neg x_3) \wedge (x_4 \vee \neg x) \wedge (\neg x_4 \vee x_3 \vee x)$

$\wedge (x_3 \vee y) \wedge (\neg x_3 \vee \neg y)$

$\wedge (x_2 \vee \neg x_1 \vee \neg y) \wedge (\neg x_2 \vee x_1) \wedge (\neg x_2 \vee y)$

$\wedge (x_1 \vee x) \wedge (\neg x_1 \vee \neg x)$

Théorème de lecture unique

Soit F une formule logique, alors
F s'écrit d'une manière unique comme :

- X_i avec X_i une variable.
- $\neg F'$ avec F' une formule logique
- $(P \alpha Q)$ avec P et Q formules logiques et $\alpha \in \{\wedge, \vee, \Rightarrow\}$

<u>preuve :</u>

• cas où F s'écrit X_i, alors la formule est de longueur 1 donc ne peut s'écrire $\neg F'$ ou $(P \alpha Q)$ et comme les variables sont distinctes F ne peut que s'écrire X_i.

• cas où F s'écrit $\neg F'$, alors la formule est au moins de taille 2 donc ne peut s'écrire X_i, de plus elle commence par "$\neg$" donc ne peut s'écrire $(P \alpha Q)$. Enfin si F s'écrit aussi $\neg F''$, en supprimant l'occurrence "$\neg$" on s'aperçoit que F' et F'' doivent forcement être la même formule logique.

• cas où F s'écrit $(P \alpha Q)$, alors, là encore, de par son premier symbole, elle ne peut s'écrire X_i ou $\neg F'$.
Reste à prouver que si F s'écrit aussi $(P' \alpha' Q')$ alors en fait P est P', Q est Q' et α est α'.

~~Pour cela, il faut considérer les parenthèses de F fermantes de poids 1~~
~~- Si il n'y en a pas alors F s'écrit $(X_i \alpha X_j)$ avec X_i et X_j des variables ou "$\neg$" des variables~~
RA
~~- Si il y en a deux elles sont nécessairement à la fin de P et Q et de P' et Q', on déduit donc que P est P', Q est Q' et α est α'.~~
~~- Si il n'y a qu'une on regarde si elle est à la fin de P ou de Q et on déduit alors~~
[illegible]

finition: (Poids des parenthèses)

à une formule logique F, le poids de sa première parenthèse est 1
lle est forcément ouvrante). Ensuite la $(i+1)^{ième}$ parenthèse prend pour poids
poids de la parenthèse précédente, +1 si elle est ouvrante, −1 si elle est fermante.

mple: $(_1(_2 \alpha \wedge \gamma)_1 \vee \gamma)_0$

me: (dont la démonstration est évidente par induction sur les formules logiques).

Soit F une formule logique, alors

- elle a un nombre pair de parenthèses
- elle a autant de parenthèses ouvrantes que fermantes
- Si elle a des parenthès, alors elle a une seule parenthèse ouvrante de
: 1 (la première) et une seule parenthèse fermante de poids 0 (la dernière)
- Si une parenthèse d'une formule logique P a pour poids p alors
a pour poids p+1 dans les formules $(P \alpha Q)$ et $(Q \alpha P)$
- Une formule logique a au plus 2 parenthèses fermantes de poids 1.

exactement: Si elle est de la forme X_i ou $\neg X_i$ elle en a 0 si
est de la forme $(P \alpha Q)$, si P et Q n'ont pas de parenthèse elle en a 0,
ou Q a des parenthèses alors c'est 1 et si P et Q ont des parenthèses c'est 2.

oup si F s'écrit $(P \alpha Q)$ et $(P' \alpha' Q')$

Si F n'a pas de parenthèses fermantes de poids 1 alors F est $(X_i \alpha X_j)$ avec X_i et X_j
ittéraux. RAS.

Si F a deux parenthèses fermantes de poids 1 alors elles sont respectivement à la fin de
à la fin de Q et de même dans l'écriture $(P' \alpha' Q')$ on déduit donc que α et α'
le même connecteur juste après la première parenthèse et donc α est α' puis
P' et Q est Q'.

Si F n'a qu'une parenthèse fermante de poids 1. Si elle est à la fin de P et P' alors
induction est triviale. De même si elle est à la fin de Q et Q' car à cette parenthèse
prend une unique parenthèse ouvrante de poids 2.
si elle est à la fin de P et Q' (ou de P' et Q, cas miroir) alors on déduit que
' Q' est (P - ce qui n'est pas possible.