

INFO-F-302, Cours d'Informatique
Fondamentale
Logique pour l'Informatique

Emmanuel Filiot
Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles

Année académique 2011-2012
Basé sur le cours *Logique Informatique* du Pr. J.-F. Raskin

La Logique Propositionnelle

Plan

- Syntaxe de la logique propositionnelle ;
- Sémantique de la logique propositionnelle ;
- Procédure de décision : tableaux sémantiques ;
- La déduction naturelle ;
- La résolution propositionnelle ;
- Les solveurs SAT.

Construction des formules

Le *vocabulaire du langage de la logique propositionnelle* est composé :

- d'un ensemble, fini ou dénombrable, de *propositions* notées p, q, r, \dots
Dans la suite, nous notons un sous-ensemble de propositions par les lettres P, Q, \dots ;
- de deux constantes : vrai (notée \top) et faux (notée \perp);
- d'un ensemble de *connecteurs logiques* : et (noté \wedge), ou (noté \vee), non (noté \neg), implique (noté \rightarrow), équivalent (noté \leftrightarrow);
- les parenthèses : $(,)$.

Construction des formules

Les *formules de la logique propositionnelle* respectent la règle de formation BNF suivante :

$$\phi ::= \top \mid \perp \mid p \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \rightarrow \phi_2 \mid \phi_1 \leftrightarrow \phi_2 \mid \neg\phi_1 \mid (\phi_1)$$

Où \top , \perp sont respectivement les constantes vrai et faux, $p \in P$ est une proposition et ϕ_1 , ϕ_2 sont des formules propositionnelles bien formées. L'ensemble des formules propositionnelles bien formées sera noté FormulesProp.

Quelques exemples de formules

Voici quelques exemples de formules et leur lecture intuitive :

- la formule propositionnelle " $p \rightarrow (q \wedge r)$ ", peut être lue de la façon suivante " p implique q et r ", ou peut être également lue comme "si p est vrai alors q et r doivent être vrais" ;
- la formule propositionnelle " $\neg(p \wedge q)$ ", peut-être lue de la façon suivante "il est faux que p et q soient vrais (en même temps)".

Ambiguïtés

L'utilisation de la *notation infix* s'accompagne de problèmes de lecture :

Comment lire $\phi_1 \wedge \phi_2 \vee \phi_3$? $\phi_1 \rightarrow \phi_2 \rightarrow \phi_3$?

Ambiguïtés

L'utilisation de la *notation infix* s'accompagne de problèmes de lecture :

Comment lire $\phi_1 \wedge \phi_2 \vee \phi_3$? $\phi_1 \rightarrow \phi_2 \rightarrow \phi_3$?

Pour lever les ambiguïtés, on utilise les parenthèses ou des règles de priorité entre opérateurs :

- si op_1 a une plus grande priorité (priorité \circledast) que op_2 alors

$e_1 op_1 e_2 op_2 e_3$ est équivalent à $((e_1 op_1 e_2) op_2 e_3)$

▶ Par ex : $2 \times 3 + 5 = (2 \times 3) + 5 = 11 \neq 2 \times (3 + 5) = 16$.

- si op_2 a une plus grande priorité (priorité \circledast) que op_1 alors

$e_1 op_1 e_2 op_2 e_3$ est équivalent à $(e_1 op_1 (e_2 op_2 e_3))$

▶ Par ex : $2 + 3 \times 5 = 2 + (3 \times 5) = 17$.

- si op est associatif à gauche alors

$e_1 op e_2 op e_3$ est équivalent à $((e_1 op e_2) op e_3)$

▶ Par ex : $10/2/5 = (10/2)/5 = 1 \neq 10/(2/5) = 25$.

Une fois ces règles fixées, à chaque formule correspond un et un seul arbre de lecture.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$\leftrightarrow \prec \neg \rightarrow \prec \vee \prec \wedge \prec \neg$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$

se lit

se lit

se lit

se lit

se lit

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$ se lit $p \vee (q \wedge r)$
se lit
se lit
se lit
se lit

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$ se lit $p \vee (q \wedge r)$

$p \rightarrow q \rightarrow p$ se lit

se lit

se lit

se lit

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
	se lit	
	se lit	
	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	
	se lit	
	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	$(p \vee q) \rightarrow r$
	se lit	
	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	$(p \vee q) \rightarrow r$
$\neg p \wedge q$	se lit	
	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	$(p \vee q) \rightarrow r$
$\neg p \wedge q$	se lit	$(\neg p) \wedge q$
	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \neg \prec \rightarrow \prec \vee \prec \wedge \prec \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	$(p \vee q) \rightarrow r$
$\neg p \wedge q$	se lit	$(\neg p) \wedge q$
$p \rightarrow q \wedge r \rightarrow s$	se lit	

Les parenthèses permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Règles de précedence

Ordre de précedence \prec sur les opérateurs :

$$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$$

et associativité à gauche pour $\leftrightarrow, \vee, \wedge$ et \neg droite pour \rightarrow .

Exemples

$p \vee q \wedge r$	se lit	$p \vee (q \wedge r)$
$p \rightarrow q \rightarrow p$	se lit	$p \rightarrow (q \rightarrow p)$
$p \vee q \rightarrow r$	se lit	$(p \vee q) \rightarrow r$
$\neg p \wedge q$	se lit	$(\neg p) \wedge q$
$p \rightarrow q \wedge r \rightarrow s$	se lit	$p \rightarrow ((q \wedge r) \rightarrow s)$

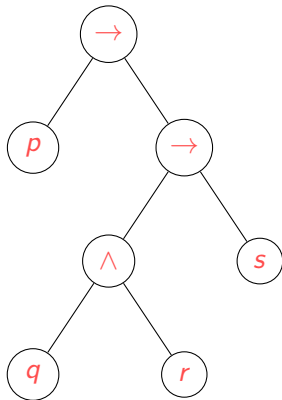
Les parenthésés permettent de contrecarrer ces règles, si elles ne conviennent pas. Elles permettent aussi de rendre une formule plus lisible, ou de ne pas devoir retenir les règles de précedence.

Arbre correspondant à une formule

La formule $p \rightarrow q \wedge r \rightarrow s$ est donc équivalente à

$$p \rightarrow ((q \wedge r) \rightarrow s)$$

et donc son arbre de lecture est :



Profondeur d'une formule

La *profondeur d'une formule* ϕ est la profondeur de son arbre de lecture associé \mathbb{A}_{ϕ} , noté A_{ϕ} .

Elle se définit de manière inductive comme suit :

- cas de base : si $\phi = p, \top, \perp$ où p est une proposition alors $Prof(\phi) = 0$;
- cas inductif : si $\phi = \phi_1 \text{ op } \phi_2$ alors $Prof(\phi) = 1 + \max(Prof(\phi_1), Prof(\phi_2))$;
si $\phi = (\phi_1)$ alors $Prof(\phi) = Prof(\phi_1)$.

Nous notons FormulesProp_i les formules propositionnelles de profondeur i .

Notons que

$$\text{FormulesProp} = \bigcup_{i \geq 0} \text{FormulesProp}_i$$

Rappel : l'induction mathématique

Il est bien connu que pour $n \geq 1$, on a

$$1 + 2 + \dots + n = \frac{n \cdot (n+1)}{2}$$

Pour démontrer que cette égalité, notée $M(n)$, est valide pour n'importe quel $n \geq 1$, on applique le principe d'*induction mathématique*.

Rappel : l'induction mathématique

SI

- Cas de base : on établit que la propriété est vraie pour $n = 1$, c'est-à-dire que l'on prouve que $M(1)$ est vraie.
- Cas inductif : en faisant l'hypothèse que la propriété est vraie au rang $n = i$ (ou pour tout rang $n \leq i$), on établit que la propriété est également vraie pour $i + 1$.

ALORS

Le principe d'induction affirme que $M(n)$ est vraie pour tout $n \geq 1$.

Rappel : l'induction mathématique

Nous utiliserons le principe d'induction mathématique pour prouver des propriétés qui sont vraies pour toutes formules de la logique propositionnelle, et ce en raisonnant par induction sur la profondeur des formules.

La sémantique

Etant donné un ensemble de propositions P , une *fonction d'interprétation* V pour P est une fonction du type $V : P \rightarrow \{\text{vrai, faux}\}$, c'est-à-dire une fonction qui assigne à chaque proposition de P la valeur vrai ou la valeur faux. L'ensemble des fonctions d'interprétation propositionnelle sera noté **FonctInterProp**.

Dans la suite, nous utiliserons parfois le terme *valuation* ou bien de fonction d'interprétation.

La sémantique

La *valeur de vérité* d'une formule propositionnelle ϕ formée à partir (d'une des propositions de l'ensemble P , évaluée avec la fonction d'interprétation V , est notée $\llbracket \phi \rrbracket_V$. Formellement,

$\llbracket \cdot \rrbracket_V : \text{FormulesProp} \times \text{FonctInterProp} \rightarrow \{\text{vrai}, \text{faux}\}$. La fonction $\llbracket \phi \rrbracket_V$ est définie par induction sur la syntaxe de ϕ de la façon suivante :

- $\phi = \top$, dans ce cas, $\llbracket \phi \rrbracket_V = \text{vrai}$;
- $\phi = \perp$, dans ce cas, $\llbracket \phi \rrbracket_V = \text{faux}$;
- $\phi = p$, dans ce cas, $\llbracket \phi \rrbracket_V = V(p)$.
- $\phi = \neg \phi_1$, $\llbracket \phi \rrbracket_V = \begin{cases} \text{faux} & \text{si } \llbracket \phi_1 \rrbracket_V = \text{vrai} \\ \text{vrai} & \text{si } \llbracket \phi_1 \rrbracket_V = \text{faux} \end{cases}$
- $\phi = \phi_1 \vee \phi_2$,

$$\llbracket \phi \rrbracket_V = \text{vrai} \text{ ssi } \begin{array}{l} \llbracket \phi_1 \rrbracket_V = \text{vrai} \\ \text{ou } \llbracket \phi_2 \rrbracket_V = \text{vrai} \end{array}$$

- $\phi = \phi_1 \wedge \phi_2$,

$$\llbracket \phi \rrbracket_V = \text{vrai} \text{ ssi } \begin{array}{l} \llbracket \phi_1 \rrbracket_V = \text{vrai} \\ \text{et } \llbracket \phi_2 \rrbracket_V = \text{vrai} \end{array}$$

La sémantique

- $\phi = \phi_1 \rightarrow \phi_2$,

$$\llbracket \phi \rrbracket_V = \text{vrai} \text{ ssi } \llbracket \phi_1 \rrbracket_V = \text{faux} \\ \text{ou } \llbracket \phi_2 \rrbracket_V = \text{vrai}$$

- $\phi = \phi_1 \leftrightarrow \phi_2$,

$$\llbracket \phi \rrbracket_V = \text{vrai} \text{ ssi } \\ \llbracket \phi_1 \rrbracket_V = \text{faux} \text{ et } \llbracket \phi_2 \rrbracket_V = \text{faux} \\ \text{ou } \llbracket \phi_1 \rrbracket_V = \text{vrai} \text{ et } \llbracket \phi_2 \rrbracket_V = \text{vrai}$$

- $\phi = (\phi_1)$, $\llbracket \phi \rrbracket_V = \llbracket \phi_1 \rrbracket_V$.

Nous notons $V \models \phi$ si et seulement si $\llbracket \phi \rrbracket_V = \text{vrai}$.

La sémantique

L'information contenue dans la définition précédente est souvent présentée sous forme de tables, appelées *tables de vérité* :

\top	\perp
vrai	faux

p	$\neg p$	(p)
vrai	faux	vrai
faux	vrai	faux

La sémantique

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
vrai	vrai	vrai	vrai	vrai	vrai
vrai	faux	faux	vrai	faux	faux
faux	vrai	faux	vrai	vrai	faux
faux	faux	faux	faux	vrai	vrai

La sémantique

Voici deux définitions importantes :

Définition (Validité)

Une formule propositionnelle ϕ est *valide* si et seulement si pour toute fonction d'interprétation V pour les propositions de ϕ , on a $V \models \phi$.

Définition (Satisfaisabilité)

Une formule propositionnelle ϕ est *satisfaisable* si et seulement si il existe une fonction d'interprétation V pour les propositions de ϕ , telle que $V \models \phi$.

Une conséquence directe de ces deux définitions est que :

Une formule propositionnelle ϕ est valide ssi sa négation $\neg\phi$ n'est pas satisfaisable.

Théorème

Une formule propositionnelle ϕ est valide ssi sa négation $\neg\phi$ n'est pas satisfaisable.

Preuve. On applique simplement les définitions. Par définition ϕ est valide si, pour toute fonction d'interprétation V , $V \models \phi$, ce qui est équivalent à $V \not\models \neg\phi$, par définition de la sémantique de \neg et donc si ϕ est valide, il n'existe pas de fonction d'interprétation V telle que $V \models \neg\phi$ et donc $\neg\phi$ n'est pas satisfaisable. L'autre direction est laissée comme exercice.



La sémantique

- Une formule propositionnelle ψ est une *conséquence logique* d'un ensemble de formules propositionnelles $\{\phi_1, \dots, \phi_n\}$ si et seulement si on a que pour toute fonction d'interprétation V (sur les propositions de $\psi, \phi_1, \dots, \phi_n$), telle que $V \models \phi_i$ pour tout $i, 1 \leq i \leq n$, on a également $V \models \psi$; ce fait est noté $\phi_1, \dots, \phi_n \models \psi$.
- Deux formules ϕ et ψ sont *équivalentes* si et seulement si on a $\phi \models \psi$ et $\psi \models \phi$.

Tableaux sémantiques

- Algorithme pour établir la satisfaisabilité/validité de formules de la logique propositionnelle.
- Le principe est très simple : pour prouver la satisfaisabilité d'une formule de la logique propositionnelle, on cherche systématiquement un modèle pour cette formule.

Tableaux sémantiques

On a besoin de quelques nouvelles définitions :

Définition

Une formule propositionnelle ϕ est un *littéral* si et seulement si ϕ est une proposition ou la négation d'une proposition.

Définition

Deux formules ϕ et $\neg\phi$ sont des *formules complémentaires* .

Tableaux sémantiques

Considérons la formule $\phi = p \wedge (\neg q \vee \neg p)$

Essayons de construire systématiquement une fonction d'interprétation V telle que

$$\llbracket \phi \rrbracket_V = \text{vrai}$$

Par définition on a

$$\llbracket \phi \rrbracket_V = \text{vrai}$$

ssi

$$\llbracket p \rrbracket_V = \text{vrai} \quad \text{et} \quad \llbracket \neg q \vee \neg p \rrbracket_V = \text{vrai}$$

et

$$\begin{aligned} & \llbracket \neg q \vee \neg p \rrbracket_V = \text{vrai} \\ & \quad \text{ssi} \\ & \llbracket \neg q \rrbracket_V = \text{vrai} \quad \underline{\text{ou}} \quad \llbracket \neg p \rrbracket_V = \text{vrai} \end{aligned}$$

et donc,

$$\begin{aligned} & \llbracket \phi \rrbracket_V = \text{vrai} \\ & \quad \text{ssi} \\ & \llbracket p \rrbracket_V = \text{vrai} \quad \underline{\text{et}} \quad \llbracket \neg q \rrbracket_V = \text{vrai} \\ & \underline{\text{ou}} \quad \llbracket p \rrbracket_V = \text{vrai} \quad \underline{\text{et}} \quad \llbracket \neg p \rrbracket_V = \text{vrai} \end{aligned}$$

Pour ϕ , on construit la fonction d'interprétation V de la façon suivante :

- $V(p) = \text{vrai}$;
- $V(q) = \text{faux}$.

et on a bien que $V \models \phi$.

Tableaux sémantiques

- Un ensemble S de littéraux est satisfaisable ssi il ne contient pas une paire de *littéraux complémentaires*. Par exemple, $\{p, \neg q\}$ est satisfaisable alors que $\{p, \neg p\}$ ne l'est pas.
- Nous avons réduit le problème de satisfaction de ϕ à un problème de satisfaction d'ensembles de littéraux : ϕ est satisfaisable ssi $\{p, \neg q\}$ est satisfaisable ou $\{p, \neg p\}$ est satisfaisable.
- Vu que chaque formule contient un ensemble fini de propositions, et donc un ensemble fini de littéraux, il y a un nombre fini d'ensembles de littéraux.

Tableaux sémantiques

Un autre exemple : $\phi = (p \vee q) \wedge (\neg p \wedge \neg q)$.

Par définition, on a

$$\llbracket \phi \rrbracket_V = \text{vrai}$$

ssi

$$\llbracket p \vee q \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg p \wedge \neg q \rrbracket_V = \text{vrai}$$

et donc,

$$\llbracket \phi \rrbracket_V = \text{vrai}$$

ssi

$$\llbracket p \vee q \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg p \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg q \rrbracket_V = \text{vrai}$$

et donc,

$$\llbracket \phi \rrbracket_V = \text{vrai}$$

ssi

$$\begin{aligned} &\text{soit } \llbracket p \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg p \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg q \rrbracket_V = \text{vrai} \\ &\text{ou } \llbracket q \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg p \rrbracket_V = \text{vrai} \text{ et } \llbracket \neg q \rrbracket_V = \text{vrai} \end{aligned}$$

Tableaux sémantiques

et donc,

ϕ est satisfaisable
ssi
 $\{p, \neq p, \neg q\}$ est satisfaisable
ou $\{q, \neq p, \neg q\}$ est satisfaisable.

Vu que les deux ensembles de littéraux ont des paires complémentaires, la formule ϕ n'est donc pas satisfaisable.

Tableaux sémantiques

L'information présente dans les développements précédents est plus facilement représentée sous forme d'un arbre.

Exemple : $\phi = (p \vee q) \wedge (\neg p \wedge \neg q)$ (tableau donné au cours)

Tableaux sémantiques – Remarques

- quand on applique le "pas de simplification" à un noeud où on sélectionne une conjonction, alors on obtient un seul fils ; on appelle ces règles, des α -règles ;
- quand on applique le "pas de simplification" à un noeud où on sélectionne une disjonction, alors on obtient deux fils ; on appelle ces règles, des β -règles.

Tableaux sémantiques

α	α_1	α_2
$\neg\neg\phi$	ϕ	
$\phi_1 \wedge \phi_2$	ϕ_1	ϕ_2
$\neg(\phi_1 \vee \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\neg(\phi_1 \rightarrow \phi_2)$	ϕ_1	$\neg\phi_2$
$\phi_1 \leftrightarrow \phi_2$	$\phi_1 \rightarrow \phi_2$	$\phi_2 \rightarrow \phi_1$

α -règles .

Remarque : toutes les formules α peuvent être considérées comme équivalentes à des conjonctions. Par exemple :

$\neg(\phi_1 \vee \phi_2)$ est équivalent à $\neg\phi_1 \wedge \neg\phi_2$

Tableaux sémantiques

β	β_1	β_2
$\phi_1 \vee \phi_2$	ϕ_1	ϕ_2
$\neg(\phi_1 \wedge \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\phi_1 \rightarrow \phi_2$	$\neg\phi_1$	ϕ_2
$\neg(\phi_1 \leftrightarrow \phi_2)$	$\neg(\phi_1 \rightarrow \phi_2)$	$\neg(\phi_2 \rightarrow \phi_1)$

β -règles .

Remarque : toutes les formules β peuvent être considérées comme équivalentes à des disjonctions. Par exemple :

$\neg(\phi_1 \wedge \phi_2)$ est équivalent à $\neg\phi_1 \vee \neg\phi_2$

Tableaux sémantiques

Algorithme de construction d'un *tableau sémantique*, quelques notations :

- un arbre représentant le tableau sémantique d'une formule ϕ sera noté T_ϕ ;
- l, n, m représentent des noeuds d'un arbre;
- $\text{Lab}(l)$ dénote l'étiquette du noeud l , c'est un ensemble de formules;
- $\text{Status}(l)$ est le statut du noeud l , ce statut est – pour tout noeud qui n'est pas une feuille et $\text{Status}(l) \in \{\text{ouvert}, \text{fermé}\}$ si l est une feuille;
- Q, R dénotent des ensembles de noeuds.

Tableaux sémantiques – Algorithme

- Initialisation : créer un noeud I puis $R := \{I\}$; $\text{Lab}(I) := \{\phi\}$;
- Itérer tant que $R \neq \emptyset$: choisir $I \in R$:
 - ▶ si $\text{Lab}(I)$ est un ensemble de littéraux **alors** :
 - si $\text{Lab}(I)$ contient une paire de littéraux complémentaires :
 $\text{Status}(I) := \text{fermé}$ et $R := R \setminus \{I\}$;
 - si $\text{Lab}(I)$ ne contient pas de paires de littéraux complémentaires :
 $\text{Status}(I) := \text{ouvert}$ et $R := R \setminus \{I\}$;
 - ▶ si $\text{Lab}(I)$ n'est pas un ensemble de littéraux **alors** : choisir $\psi \in \text{Lab}(I)$ qui n'est pas un littéral et :
 - si ψ est une α -formule alors créer un fils m pour I et :
 - ★ $R := R \cup \{m\} \setminus \{I\}$;
 - ★ $\text{Lab}(m) := (\text{Lab}(I) \setminus \{\psi\}) \cup \{\alpha_1, \alpha_2\}$;
 - si ψ est une β -formule alors créer deux fils, m et n , pour I et :
 - ★ $R := R \cup \{m, n\} \setminus \{I\}$;
 - ★ $\text{Lab}(m) := (\text{Lab}(I) \setminus \{\psi\}) \cup \{\beta_1\}$;
 - ★ $\text{Lab}(n) := (\text{Lab}(I) \setminus \{\psi\}) \cup \{\beta_2\}$;

Tableaux sémantiques

- Un tableau dont la construction est terminée est appelé un *tableau complet* .
- Un *tableau fermé* est un tableau complet dont toutes les feuilles sont étiquetées avec *fermé*. C'est un *tableau ouvert* sinon.

Tableaux sémantiques

Théorème

Si T_ϕ est le tableau complet de la formule ϕ , alors ϕ est non satisfaisable ssi T_ϕ est fermé.

Avant de prouver la correction de l'algorithme par rapport au théorème ci-dessus observons quelques conséquences de ce théorème :

- ϕ est satisfaisable ssi T_ϕ est ouvert (T_ϕ a au moins une feuille ouverte)
- ϕ est valide ssi $T_{\neg\phi}$ est fermé.

Tableaux sémantiques

Théorème

Si T_ϕ est le tableau complet de la formule ϕ , alors ϕ est non satisfaisable ssi T_ϕ est fermé.

Avant de prouver la correction de l'algorithme par rapport au théorème ci-dessus observons quelques conséquences de ce théorème :

- ϕ est satisfaisable ssi T_ϕ est ouvert (T_ϕ a au moins une feuille ouverte)
- ϕ est valide ssi $T_{\neg\phi}$ est fermé.

Questions

- 1 Si toutes les feuilles de T_ϕ sont ouvertes, est-ce que cela signifie que ϕ est valide ?
- 2 quelle est la taille maximale de T_ϕ (nombre de noeuds) en fonction du nombre de symboles de ϕ ?

Tableaux sémantiques

Correction de l'algorithme : qu'est-ce que cela veut dire ?

- *terminaison* : l'algorithme termine pour toutes les formules de la logique propositionnelle ;
- *adéquation* : si l'algorithme construit un tableau qui ferme pour une formule ϕ alors la formule ϕ est non satisfaisable ;
- *complétude* : pour toute formule ϕ non satisfaisable, l'algorithme construit un tableau fermé.

Tableaux sémantiques – Terminaison de l'Algorithme

On définit une fonction de “poids” qui associe à chaque ensemble de formules un nombre naturel ; on note $\text{Poids}(I)$ le poids de l'étiquette du noeuds I .

On montre que si m est un fils de I alors

$$\text{Poids}(m) < \text{Poids}(I).$$

Vu que $\text{Poids}()$ est une fonction des ensembles de formules vers les naturels alors la profondeur de tout tableau sémantique est finie et donc l'algorithme termine.

Tableaux sémantiques – Terminaison de l'Algorithme

Définition de la fonction $\text{Poids}(\Psi)$. On définit d'abord cette fonction pour une formule ψ par induction sur la structure de la formule.

- $\text{Poids}(p) = 1$.
- $\text{Poids}(\neg\psi) = 1 + \text{Poids}(\psi)$;
- $\text{Poids}(\psi_1 \wedge \psi_2) = 2 + \text{Poids}(\psi_1) + \text{Poids}(\psi_2)$;
- $\text{Poids}(\psi_1 \vee \psi_2) = 2 + \text{Poids}(\psi_1) + \text{Poids}(\psi_2)$;
- $\text{Poids}(\psi_1 \rightarrow \psi_2) = 2 + \text{Poids}(\psi_1) + \text{Poids}(\psi_2)$;
- $\text{Poids}(\psi_1 \leftrightarrow \psi_2) = 2 \times (\text{Poids}(\psi_1) + \text{Poids}(\psi_2)) + 5$

et finalement, $\text{Poids}(\Psi) = \sum_{\psi \in \Psi} \text{Poids}(\psi)$.

Tableaux sémantiques – Terminaison

Propriétés :

Propriétés

- la fonction **Poids()** attribue un poids fini et strictement positif à chaque formule ;
- le poids de l'étiquette d'un noeud \wedge est toujours strictement supérieur aux poids des étiquettes de ses fils.

Preuves laissées comme exercices.

Par conséquent, l'algorithme termine.

Tableaux sémantiques – Adéquation de l’algorithme.

Adéquation de l’algorithme :

- On doit montrer que “Si T_ϕ est fermé alors ϕ est non satisfaisable”.
- On montre une propriété plus forte : “Si un sous-arbre de racine I est fermé alors l’ensemble de formules $Lab(I)$ est non satisfaisable”. Pour établir cette propriété, on raisonne par induction sur la profondeur du sous-arbre.

Tableaux sémantiques – Adéquation de l'Algorithme

Lemme

Si un sous-arbre de racine I est fermé alors $\text{Lab}(I)$ est non satisfaisable.

Preuve : Par induction sur la profondeur du sous-arbre.

Cas de base. $\text{Prof}(I) = 0$. Alors $\text{Lab}(I)$ est un ensemble de littéraux. Si I est fermé, $\text{Lab}(I)$ contient une paire de littéraux complémentaires et donc $\text{Lab}(I)$ est non satisfaisable.

Tableaux sémantiques – Adéquation de l'Algorithme

Lemme

Si un sous-arbre de racine I est fermé alors $\text{Lab}(I)$ est non satisfaisable.

Preuve : Par induction sur la profondeur du sous-arbre.

Induction. Dans ce cas $\text{Prof}(I) > 0$, et l'algorithme a sélectionné $\psi \in \text{Lab}(I)$.

Deux cas sont possibles :

Premier Cas : ψ est une α -formule.

Traitons le cas $\psi \equiv \neg\neg\phi$. Dans ce cas, I a un fils m . On a $\text{Lab}(m) = \text{Lab}(I) \setminus \{\neg\neg\phi\} \cup \{\phi\}$. On sait que le sous-arbre de racine m est fermé (dans le cas contraire I ne pourrait pas l'être). Par **HI**, on a que $\text{Lab}(m)$ est non satisfaisable et donc pour toute valuation V , on a soit $V \not\models \text{Lab}(I) \setminus \{\neg\neg\phi\}$ ou $V \not\models \phi$. Donc pour chaque valuation V on sait que $V \not\models \text{Lab}(I) \setminus \{\neg\neg\phi\}$ ou $V \not\models \neg\neg\phi$. Par conséquent, nous avons bien que pour toute valuation V , $V \not\models \text{Lab}(I)$, et $\text{Lab}(I)$ est non satisfaisable.

Tableaux sémantiques – Adéquation de l'Algorithme

Lemme

Si un sous-arbre de racine l est fermé alors $\text{Lab}(l)$ est non satisfaisable.

Preuve : Par induction sur la profondeur du sous-arbre.

Induction. Dans ce cas $\text{Prof}(l) > 0$, et l'algorithme a sélectionné $\psi \in \text{Lab}(l)$.

Deux cas sont possibles :

Deuxième Cas : ψ est une β -formule.

Traitons le cas $\psi \equiv \phi_1 \vee \phi_2$. Dans ce cas, l a deux fils m et n . On a

$\text{Lab}(m) = \text{Lab}(l) \setminus \{\phi_1 \vee \phi_2\} \cup \{\phi_1\}$ et

$\text{Lab}(n) = \text{Lab}(l) \setminus \{\phi_1 \vee \phi_2\} \cup \{\phi_2\}$. Par **HI** on sait que $\text{Lab}(m)$ et $\text{Lab}(n)$

sont non satisfaisables. Donc toute valuation V qui satisfait

$\text{Lab}(l) \setminus \{\phi_1 \vee \phi_2\}$ ne satisfait ni ϕ_1 ni ϕ_2 , et donc ne satisfait pas $\phi_1 \vee \phi_2$.

On a donc bien que $\text{Lab}(l) \setminus \{\phi_1 \vee \phi_2\} \cup \{\phi_1 \vee \phi_2\} = \text{Lab}(l)$ est non satisfaisable.

Tableaux sémantiques – Complétude

Complétude de l'algorithme :

- On doit montrer que “Si ϕ est non satisfaisable alors T_ϕ est fermé”.
- Prouver cette propriété revient à établir la contraposée : “Si T_ϕ est ouvert alors la formule ϕ est satisfaisable”.

Pour établir cela, nous avons besoin de quelques notions supplémentaires.

Tableaux sémantiques – Complétude

Un ensemble de formules Ψ (on considère des littéraux, conjonctions ou disjonctions) a la *propriété de Hintikka* ssi les trois propriétés suivantes sont vérifiées :

- 1 pour toute proposition p , on a soit $p \notin \Psi$ ou $\neg p \notin \Psi$;
- 2 si $\phi_1 \vee \phi_2 \in \Psi$ alors on a $\phi_1 \in \Psi$ ou $\phi_2 \in \Psi$;
- 3 si $\phi_1 \wedge \phi_2 \in \Psi$ alors on a $\phi_1 \in \Psi$ et $\phi_2 \in \Psi$.

Propriété des ensembles de Hintikka :

Tout ensemble de formules Ψ qui a la propriété de Hintikka est satisfaisable.

Lemme

Tout ensemble de formules Ψ qui a la propriété de Hintikka est satisfaisable.

Preuve. On construit une valuation V de la façon suivante :

$$V(p) = \text{Vrai} \text{ si } p \in \Psi$$

$$V(p) = \text{Faux} \text{ si } \neg p \in \Psi$$

et (arbitrairement) $V(p) = \text{Vrai}$
si $p \notin \Psi$ et $\neg p \notin \Psi$.

Montrons que toute formule $\phi \in \Psi$ est satisfaite par V .

On raisonne par induction sur la structure de ϕ :

$$\text{CB : } \begin{cases} \phi = p, \llbracket \phi \rrbracket_V = V(p) = \text{Vrai} & (p \in \Psi) \\ \phi = \neg p, \llbracket \phi \rrbracket_V = \neg V(p) = \text{Vrai} & (\neg p \in \Psi) \end{cases}$$

CI :

- $\phi = \phi_1 \vee \phi_2$.

Par définition des ensembles de Hintikka on sait que soit $\phi_1 \in \Psi$, soit $\phi_2 \in \Psi$. Par **HI**, si $\phi_1 \in \Psi$ on a $\llbracket \phi_1 \rrbracket_V = \text{vrai}$, et si $\phi_2 \in \Psi$ on a $\llbracket \phi_2 \rrbracket_V = \text{vrai}$. Donc $\llbracket \phi_1 \vee \phi_2 \rrbracket_V = \text{vrai}$.

- $\phi = \phi_1 \wedge \phi_2$. On sait que $\phi_1 \in \Psi$ et $\phi_2 \in \Psi$. Par **HI** on a $\llbracket \phi_1 \rrbracket_V = \llbracket \phi_2 \rrbracket_V = \text{vrai}$ et donc $\llbracket \phi_1 \wedge \phi_2 \rrbracket_V = \text{vrai}$.

Tableaux sémantiques – Complétude

On appelle l'ensemble des noeuds que l'on parcourt depuis la racine d'un arbre jusqu'à une feuille l la *branche* de l ; on note $\text{Branche}(A, l)$ la branche qui mène de la racine de l'arbre A à la feuille l .

On a le lemme suivant :

Lemme

Soit l une feuille ouverte de T_ϕ alors

$$\bigcup_{m \in \text{Branche}(A, l)} \text{Lab}(m)$$

est un ensemble de formules qui a la propriété de Hintikka.

Preuve

. on montre que chaque propriété est vérifiée :

- 1 Pour tout p , $p \notin \Psi$ ou $\neg p \notin \Psi$. En effet I est une feuille ouverte et donc elle ne contient pas de littéraux complémentaires.
- 2 Si $\phi_1 \vee \phi_2 \in \Psi$ alors on a appliqué une β -règle et soit $\phi_1 \in \Psi$ ou $\phi_2 \in \Psi$.
- 3 Si $\phi_1 \wedge \phi_2 \in \Psi$ alors on a appliqué une α -règle et $\phi_1 \in \Psi$ ainsi que $\phi_2 \in \Psi$.

Tableaux sémantiques – Complétude

On obtient la preuve de complétude par le raisonnement suivant :
Si T_ϕ est ouvert alors il contient une feuille I qui est ouverte et donc

$$\bigcup_{m \in \text{Branche}(A, I)} \text{Lab}(m)$$

est un ensemble de formules qui a la propriété de Hintikka. Donc toutes les formules de

$$\bigcup_{m \in \text{Branche}(A, I)} \text{Lab}(m)$$

sont satisfaisables et en particulier

$$\phi \in \bigcup_{m \in \text{Branche}(A, I)} \text{Lab}(m).$$