INTRODUCTION

Logique : *"science du raisonnement en lui-même, abstraction faite de la matière à laquelle il s'applique et de tout processus psychologique"* [Larousse]

Remonte à l'antiquité (Aristote, IV^e S. AC)
But : formaliser le raisonnement
$\rightarrow$ règles de déduction

Etant donné un certain nombre de prémisses, la déduction suivant les règles de logique doit garantir la vérité de la conclusion

Exemple :

1. Tous les hommes sont mortels
2. Socrate est un homme
3. Donc, Socrate est mortel

Difficultés :

1. imprécision du langage naturel

   Exemple :
   (a) Some cars rattle
   (b) My car is some car
   (c) Therefore, my car rattles
2. paradoxes

   This sentence is false

Depuis 1850 : intérêt des mathématiciens pour la formalisation du concept de démonstration mathématique

Logique moderne : logique formelle ou logique mathématique

Logique formelle : s'intéresse aux méthodes
– permettant de déterminer si un raisonnement est correct ou non
– vérifiables automatiquement
$\Rightarrow$ nécessité de formalisation du raisonnement

Comment ?

On définit un langage formel permettant de représenter les phrases à analyser.

La vérité d'une phrase dépend du monde dont elle parle.

Exemple : Il pleut.

On sépare donc la structure logique du raisonnement de l'information sur le monde au sujet duquel on veut raisonner.

– Structure logique du raisonnement : *formule* (ou ensemble de formules)
  $\rightarrow$ *syntaxe*
– Information sur le monde à propos duquel on raisonne : *interprétation*
  $\rightarrow$ *sémantique*

Exemple : S'il pleut, alors la route est mouillée.

Les raisonnements qui ont une structure logique correcte doivent être *vrais quelle que soit l'interprétation*, c'est-à-dire *valides*.

Le but de la logique est donc de permettre de distinguer les formules valides ou les déductions valides.

Calcul des propositions : langage formel composé de phrases (*propositions*) qui sont soit vraies soit fausses

Exemples :
- Un plus un égale deux
- Je donne cours de logique le mardi
- Un plus un égale deux *et* la terre tourne autour du soleil

Calcul des prédicats : langage formel plus riche que celui du calcul des propositions – permet d'exprimer des propriétés vraies pour certains individus pris dans un ensemble

Exemples :
- Je_donne_cours(x,y)
- $x < y$

Connecteurs formels, connecteurs naturels :

J'irai au théâtre *ou bien* j'irai au cinéma.

Il pleut *ou* le soleil brille.

S'il pleut, *alors* la route est mouillée.

Les marées se produisent *et* l'attraction universelle s'exerce.

Les marées se produisent *et* la neige est blanche.

Il *n*'est *pas* bête.

Si c'est pile *alors* je gagne *sinon* tu perds !

'*parce que*' n'est pas un connecteur logique !

Les marées se produisent *parce que* l'attraction universelle s'exerce.

Les marées se produisent *parce que* la neige est blanche.

La vérité des phrases utilisant '*parce que*' ne dépend pas que de la vérité des composantes.

Proposition : phrase atomique – à laquelle on peut affecter une *valeur de vérité* (*vrai* ou *faux*)

Les propositions atomiques sont combinées à l'aide de *connecteurs* (booléens) pour construire des propositions composées.

Quels connecteurs ?

On veut pouvoir obtenir la valeur de vérité d'une proposition composée à partir de la valeur de vérité de ses composantes et de la nature du connecteur (*connecteur vérifonctionnel*).

Connecteurs naturels ?

Opérateurs booléens

Opérateurs booléens

Il y a 2^{2^n} opérateurs booléens à n places ($y = op(x_1, \ldots, x_n)$), puisque la table d'un opérateur n-aire comporte 2^n lignes, chacune pouvant prendre deux valeurs.

Opérateurs unaires (4)

x	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$
T	T	T	F	F
F	T	F	T	F

x	y	$\circ_1$	$\circ_2$	$\circ_3$	$\circ_4$	$\circ_5$	$\circ_6$	$\circ_7$	$\circ_8$
T	T	T	T	T	T	T	T	T	T
T	F	T	T	T	T	F	F	F	F
F	T	T	T	F	F	T	T	F	F
F	F	T	F	T	F	T	F	T	F

Opérateurs binaires (16)

x	y	$\circ_9$	$\circ_{10}$	$\circ_{11}$	$\circ_{12}$	$\circ_{13}$	$\circ_{14}$	$\circ_{15}$	$\circ_{16}$
T	T	F	F	F	F	F	F	F	F
T	F	T	T	T	T	F	F	F	F
F	T	T	T	F	F	T	T	F	F
F	F	T	F	T	F	T	F	T	F

Opérateurs binaires usuels

op.	nom	symbole	se lit
$\circ_2$	disjonction	$\vee$	ou
$\circ_3$	conditionnel inverse	$\Leftarrow$	si
$\circ_5$	conditionnel	$\Rightarrow$, $\supset$	si . . . alors
$\circ_7$	biconditionnel	$\equiv$, $\Leftrightarrow$	si et seulement si
$\circ_8$	conjonction	$\wedge$	et
$\circ_9$		$\uparrow$	*nand*
$\circ_{10}$	ou exclusif	$\oplus$, $\veebar$	*xor*
$\circ_{15}$		$\downarrow$	*nor*

x	y	$\wedge$	$\vee$	$\equiv$	$\oplus$	$\Rightarrow$
T	T	T	T	T	F	T
T	F	F	T	F	T	F
F	T	F	T	F	T	T
F	F	F	F	T	F	T

Opérateurs n-aires

Théorème. Tout opérateur n-aire ($n > 2$) peut se réduire à une combinaison de deux opérateurs $(n-1)$-aires, d'opérateurs binaires et de négations.

$$M(p_1, \ldots, p_{n-1}, p_n) \quad \longleftrightarrow$$
$$[(p_n \Rightarrow M(p_1, \ldots, p_{n-1}, true)) \ \wedge \ (\neg p_n \Rightarrow M(p_1, \ldots, p_{n-1}, false))]$$

$$M(p_1, \ldots, p_{n-1}, p_n) \quad \longleftrightarrow$$
$$[(p_n \wedge M(p_1, \ldots, p_{n-1}, true)) \ \vee \ (\neg p_n \wedge M(p_1, \ldots, p_{n-1}, false))]$$

On peut donc éliminer tout connecteur n-aire ($n > 2$).

Exemple classique : if p then q else r

Réductions habituelles :

$$(p \Rightarrow q) \wedge (\neg p \Rightarrow r) \ ; \ \ (p \wedge q) \vee (\neg p \wedge r) \ ; \ \ (p \wedge q) \ \veebar \ (\neg p \wedge r) \, .$$

SYNTAXE DU CALCUL DES PROPOSITIONS

Soit $\mathcal{P}$: un ensemble de symboles arbitraires appelés *propositions atomiques* ou *atomes*. $\mathcal{P} = \{p, q, r, \ldots\}$

Définition. Une *formule* du calcul des propositions est une chaîne de symboles générée par la grammaire

$$\begin{array}{lll} formula & ::= & p\,, \quad \text{pour tout } p \in \mathcal{P} \\ formula & ::= & true \mid false \\ formula & ::= & \neg formula \\ formula & ::= & (formula \text{ op } formula) \\ op & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow \end{array}$$

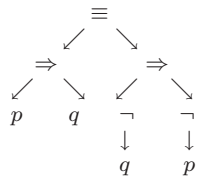
Exemple : Dérivation de la formule $(p \wedge q)$ en utilisant la grammaire :

1. *formula*
2. (*formula* op *formula*)
3. (*formula* $\wedge$ *formula*)
4. ($p \wedge$ *formula*)
5. ($p \wedge q$)

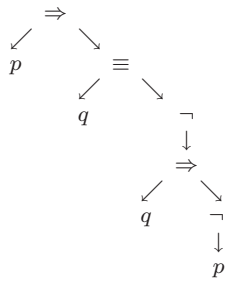
Une dérivation :

1. $formula$
2. $(formula \equiv formula)$
3. $((formula \Rightarrow formula) \equiv formula)$
4. $((p \Rightarrow formula) \equiv formula)$
5. $((p \Rightarrow q) \equiv formula)$
6. $((p \Rightarrow q) \equiv (formula \Rightarrow formula))$
7. $((p \Rightarrow q) \equiv (\neg formula \Rightarrow formula))$
8. $((p \Rightarrow q) \equiv (\neg q \Rightarrow formula))$
9. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg formula))$
10. $((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$

L'ordre des dérivations n'est pas total mais partiel ; on représente donc une dérivation par un arbre. L'arbre de gauche correspond à la formule ci-dessus, celui de droite à une formule différant par l'emplacement des parenthèses.



$((p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p))$



$(p \Rightarrow (q \equiv \neg(q \Rightarrow \neg p)))$

Règles simplificatrices.

- Omission des parenthèses extérieures :
  $p \wedge q$ au lieu de $(p \wedge q)$, et
  $q \equiv \neg(q \Rightarrow \neg q)$ au lieu de $(q \equiv \neg(q \Rightarrow \neg q))$.
- Associativité des opérateurs $\wedge$ et $\vee$:
  $p \vee q \vee r$ au lieu de $(p \vee q) \vee r$ ou $p \vee (q \vee r)$.
- Opérateurs non associatifs groupés à gauche :
  $p \Rightarrow q \Rightarrow r$ au lieu de $(p \Rightarrow q) \Rightarrow r$.
- Priorité des opérateurs :
  $a + b * c \;=\; a + (b * c) \;\neq\; (a + b) * c$,
  on convient par exemple que $p \vee q \wedge r$ équivaut à $p \vee (q \wedge r)$ et non à $(p \vee q) \wedge r$.
  La suite $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftarrow$, $\equiv$ reprend les connecteurs logiques, par ordre décroissant de priorité.

Dans la suite, seules les deux premières règles de simplification seront utilisées.

SEMANTIQUE DU CALCUL DES PROPOSITIONS

Nous donnons la sémantique pour le langage du calcul des propositions défini par :

Syntaxe : Une *formule* du calcul des propositions est une chaîne de symboles générée par la grammaire

$$\begin{array}{lll} \mathit{formula} & ::= & p\,, \quad \text{pour tout } p \in \mathcal{P} \\ \mathit{formula} & ::= & \mathit{true} \mid \mathit{false} \\ \mathit{formula} & ::= & \neg \mathit{formula} \\ \mathit{formula} & ::= & (\mathit{formula}\ \text{op}\ \mathit{formula}) \\ \mathit{op} & ::= & \vee \mid \wedge \mid \Rightarrow \mid \equiv \mid \Leftarrow \end{array}$$

La sémantique doit être *compositionnelle (dénotationnelle)* : la sémantique d'une formule est une *fonction* de la sémantique de ses composantes.

Donc la valeur de vérité d'une formule ne pourra dépendre que de la valeur de vérité des propositions qu'elle contient.

Interprétations

Soit A une formule du calcul des propositions et $\{p_1, \ldots, p_n\}$ l'ensemble des atomes apparaissant dans A.

Une *interprétation* de A est une fonction $v : \{p_1, \ldots, p_n\} \to \{T, F\}$ qui attribue une *valeur de vérité* à chaque atome ; v attribue une valeur de vérité à A suivant la définition inductive suivante.

A	$v(A_1)$	$v(A_2)$	$v(A)$
true			T
false			F
$\neg\, A_1$	T		F
$\neg\, A_1$	F		T
$A_1 \vee A_2$	F	F	F
$A_1 \vee A_2$	sinon		T
$A_1 \wedge A_2$	T	T	T
$A_1 \wedge A_2$	sinon		F
$A_1 \Rightarrow A_2$	T	F	F
$A_1 \Rightarrow A_2$	sinon		T
$A_1 \Leftarrow A_2$	F	T	F
$A_1 \Leftarrow A_2$	sinon		T
$A_1 \equiv A_2$	$v(A_1) = v(A_2)$		T
$A_1 \equiv A_2$	$v(A_1) \neq v(A_2)$		F

Exemple d'interprétation

Formule :

$$(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$$

Interprétation :

$$v(p) = F,\ v(q) = T$$

La valeur de vérité de la formule peut se déterminer comme suit :

$$\begin{array}{l} v(p \Rightarrow q) = T \\ v(\neg\, q) = F \\ v(\neg\, p) = T \\ v(\neg\, q \Rightarrow \neg\, p) = T \\ v((p \Rightarrow q) \equiv (\neg\, q \Rightarrow \neg\, p)) = T \end{array}$$

Remarque. Conceptuellement, la formule *true* et la valeur de vérité T sont des objets très différents. L'usage d'une seule notation pour les deux objets est cependant fréquent, et peu gênant en pratique.

Une interprétation détermine de manière unique la valeur de vérité d'une formule.

Ceci provient de l'unicité de l'arbre syntaxique (non-ambiguïté).

Exemple : Soit $v(p) = F$ et $v(q) = T$.

Alors, $v(p \Rightarrow (q \Rightarrow p)) = T$ et $v((p \Rightarrow q) \Rightarrow p) = F$.

Une fonction v est une *interprétation pour une formule A (un ensemble de formules U)* si v attribue une valeur de vérité au moins aux atomes intervenant dans A (dans U).

Exemple : Soit $v(p) = T$, $v(q) = F$, $v(r) = T$, $v(s) = T$ une interprétation pour l'ensemble de formules :

$$\{p \Rightarrow q,\ p,\ (p \vee s) \equiv (s \wedge q)\}$$

v attribue les valeurs de vérité :

$$v(p \Rightarrow q) = F$$
$$v(p) = T$$
$$v((p \vee s) \equiv (s \wedge q)) = F$$

- Connecteur conditionnel :

antécédent $\Rightarrow$ *conséquent*

Quand l'antécédent est vrai, le conditionnel a la valeur du conséquent.

Si la terre tourne autour du soleil, *alors* un plus un égale trois.

Quand l'antécédent est faux, le conditionnel est vrai.

Si le soleil tourne autour de la terre, *alors* un plus un égale trois.

Seul connecteur qui satisfait :
- Si l'antécédent est vrai, le conditionnel a la valeur du conséquent.
- La valeur de vérité du conditionneldépend de la valeur de vérité de ses deux opérandes.
- Le connecteur n'est pas commutatif.

Calcul des propositions vs. langage naturel

Les connecteurs du langage naturel ne sont pas nécessairement vérifonctionnels.

- Connecteur *et* :

Il fait beau et je possède une voiture.

Je possède une voiture et il fait beau.

Il prit peur et tira sur l'intrus.

Il tira sur l'intrus et prit peur.

- Connecteur *ou* :

Un nombre entier est pair ou impair.

Consistance et validité

Soit A une formule propositionnelle.
- Une interprétation v de A est un *modèle* de A si $v(A) = T$.
- A est *satisfaisable* ou *consistant* si A a au moins un modèle.
- A est *valide*, ou A est une *tautologie*, si $v(A) = T$ pour toute interprétation v. Notation : $\models A$
- A est *insatisfaisable* ou *inconsistant* si A n'est pas satisfaisable, c'est-à-dire si $v(A) = F$, pour toute interprétation v.

Remarque. "(In)satisfaisabilité" est le terme propre ; "(in)consistance" est souvent préféré par euphonie.

Théorème.
Une formule A est valide si et seulement si sa négation $\neg A$ est insatisfaisable.

A valide
ssi $v(A) = T$, pour toute interprétation v
ssi $v(\neg A) = F$, pour toute interprétation v
ssi $\neg A$ est insatisfaisable. □

Procédure de décision

Définition. Soit U une classe de formules. Un algorithme est une *procédure de décision* pour U si, étant donné une formule A, il s'arrête en fournissant la réponse 'oui' si $A \in U$ et la réponse 'non' si $A \notin U$.

Logique formelle : s'intéresse aux procédures de décision pour la classe des formules valides et celle des formules satisfaisables.

Décider la satisfaisabilité revient à décider la validité :

A est-elle valide ?

Appliquer la procédure de décision pour la satisfaisabilité à $\neg A$.

Si $\neg A$ est satisfaisable, alors A n'est pas valide.
Si $\neg A$ n'est pas satisfaisable, alors A est valide.

$\hookrightarrow$ *Procédure de réfutation* : on essaye d'établir la validité de A en cherchant à réfuter $\neg A$.

Consistance des ensembles de formules

- Tout sous-ensemble d'un ensemble consistant est consistant.
- Tout sur-ensemble d'un ensemble inconsistant est inconsistant.
- Si E est consistant et si A est valide, alors $E \cup \{A\}$ est consistant.
- Si E est inconsistant et si A est valide, alors $E \setminus \{A\}$ est inconsistant.

On préserve la consistance d'un ensemble par suppression de formules (quelconques) et aussi par adjonction de formules valides.

On préserve l'inconsistance d'un ensemble par adjonction de formules (quelconques) et aussi par suppression de formules valides.

Ensemble de formules

Un *modèle* d'un ensemble E de formules est une interprétation qui rend vraies toutes les formules de E.

Un ensemble est *consistant*, ou *satisfaisable*, s'il admet au moins un modèle.

- Toute interprétation est un modèle de $\emptyset$.
- Les modèles du singleton $\{A\}$ sont les modèles de la formule A.
- Les modèles de l'ensemble fini $\{A_1, \ldots, A_n\}$ sont les modèles de la conjonction $A_1 \wedge \cdots \wedge A_n$.

!! Attention !!

La notion de modèle s'étend aux ensembles infinis,
mais il n'existe pas de formules infinies.

Remarque importante. On dit parfois qu'un ensemble est *valide* si toutes les interprétations de cet ensemble sont des modèles. Cette notion est accessoire parce qu'un ensemble est valide si et seulement si tous ses éléments sont des formules valides. Par contre, *un ensemble de formules consistantes peut être inconsistant.* Un exemple simple est $\{p, \neg p\}$.

Conséquence logique

Une formule A est *conséquence logique* d'un ensemble E si tout modèle de E est un modèle de A.

Notation : $E \models A$.

Remarque. Si E est valide, et en particulier si E est vide, A est conséquence logique de E si et seulement si A est valide. On peut donc voir la notation

$$\models A$$

comme une abréviation de

$$\emptyset \models A$$

Autrement dit, une formule est valide si et seulement si elle est conséquence logique de l'ensemble vide.

Une variante de ce théorème est :

Une formule est valide si et seulement si elle est conséquence logique de tout ensemble.

Théorème de la déduction

Soit A une formule et soit $U = \{A_1, \ldots, A_n\}$ un ensemble fini de formules. Les trois conditions suivantes sont équivalentes :
- A est une conséquence logique de U, $U \models A$;
- l'ensemble $U \cup \{\neg A\}$ est inconsistant, $U \cup \{\neg A\} \models \mathit{false}$;
- le conditionnel $(A_1 \wedge \ldots \wedge A_n) \Rightarrow A$ est valide, $\models (A_1 \wedge \ldots \wedge A_n) \Rightarrow A$.

Les deux premières conditions conservent un sens et restent équivalentes si l'ensemble U est infini. La troisième condition n'a plus de sens, puisque la notion de "formule infinie" n'a pas été définie.

Définition. La *théorie* d'un ensemble U de formules est l'ensemble des conséquences logiques de U, soit $\mathcal{T}(U) = \{A : U \models A\}$; les éléments de U sont les *axiomes* et les éléments de $\mathcal{T}(U)$ sont les *théorèmes*. Cette notion est surtout employée dans le cadre de la logique des prédicats.

Consistance et inconsistance

Rappel.

On préserve la consistance d'un ensemble par suppression de formules (quelconques) et aussi par adjonction de formules valides.

On préserve l'inconsistance d'un ensemble par adjonction de formules (quelconques) et aussi par suppression de formules valides.

Renforcement.

On préserve la consistance d'un ensemble par suppression de formules (quelconques) et aussi par adjonction de conséquences logiques.

On préserve l'inconsistance d'un ensemble par adjonction de formules (quelconques) et aussi par suppression de conséquences logiques (de ce qui n'est pas supprimé !).

Equivalence logique

Définition. Deux formules propositionnelles A_1 et A_2 sont dites *logiquement équivalentes* (noté $A_1 \leftrightarrow A_2$) si elles ont les mêmes modèles, c'est-à-dire si $v(A_1) = v(A_2)$ pour toute interprétation v.

En pratique, il est suffisant de ne considérer que les interprétations qui attribuent une valeur aux atomes des formules considérées.

Exemple : $p \vee q \leftrightarrow q \vee p$

p	q	$v(p \vee q)$	$v(q \vee p)$
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

Si A_1 et A_2 sont des formules propositionnelles, on a $A_1 \vee A_2 \leftrightarrow A_2 \vee A_1$.

Soit v une interprétation de $\{A_1, A_2\}$.
$v(A_1 \vee A_2) = T$ ssi $v(A_1) = T$ ou $v(A_2) = T$
ssi $v(A_2) = T$ ou $v(A_1) = T$
ssi $v(A_2 \vee A_1) = T$

Donc, $A_1 \vee A_2 \leftrightarrow A_2 \vee A_1$. □

Equivalence logique ≠ connecteur '≡' !

Attention à la confusion possible entre
- *le langage-objet* : la logique elle-même ($\equiv$, $p \vee q$, ...)
- *le méta-langage* : langage semi-formel dans lequel nous nous exprimons pour parler de la logique ($\leftrightarrow$, $A_1 \vee A_2$, ...) !

Mais il existe une relation entre la notion d'équivalence logique (du méta-langage) et le connecteur '≡' (du langage-objet).

Théorème (Relation entre équivalence logique et connecteur '≡') : $A_1 \leftrightarrow A_2$ ssi $(A_1 \equiv A_2)$ est valide.

$v(A_1 \equiv A_2) = T$, pour toute interprétation v
ssi $v(A_1) = v(A_2)$, pour toute interprétation v
(définition inductive de la sémantique de '≡')
ssi $A_1 \leftrightarrow A_2$ (définition de l'équivalence logique $\leftrightarrow$).

□

Quelques théorèmes

Les conditions suivantes sont équivalentes :
- $A_1 \leftrightarrow A_2$;
- $\models (A_1 \equiv A_2)$;
- $\models (A_1 \Rightarrow A_2)$ et $\models (A_2 \Rightarrow A_1)$;
- $\{A_1\} \models A_2$ et $\{A_2\} \models A_1$.

Remarque. On écrit souvent
$A \models B$ au lieu de $\{A\} \models B$, et
$E, A, B \models C$ au lieu de $E \cup \{A, B\} \models C$.

Remarque. On écrit parfois $v \models A$ au lieu de $v(A) = T$. Cela vient de ce que l'on assimile parfois l'interprétation v à l'ensemble des formules (ou simplement des littéraux) dont v est un modèle.

Théorème de l'échange

Enoncé. Soient A et B deux formules.
Soient C_A une formule admettant A comme sous-formule, et C_B la formule obtenue en remplaçant une ou plusieurs occurrence(s) de A par B dans C_A. On a

$$(A \equiv B) \models (C_A \equiv C_B)\,.$$

Exemple.
$A =_{def} p$, $B =_{def} \neg\neg p$
$C_A =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$

Trois choix possibles pour C_B :
$C_B =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$
$C_B =_{def} (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$
$C_B =_{def} (\neg\neg p \Rightarrow q) \equiv (\neg q \Rightarrow \neg\neg\neg p)$.

Comme A et B sont logiquement équivalents, C_A et C_B le sont aussi.

Remarque. On peut inclure le cas trivial dans le théorème, qui devient : "Soient A, B et C_A trois formules. Soit C_B la formule obtenue en remplaçant zéro, une ou plusieurs occurrence(s) de A par B dans C_A. On a $(A \equiv B) \models (C_A \equiv C_B)$. Naturellement, on ne peut remplacer plus d'occurrences de A qu'il n'y en a dans C_A !

Sous-formules

- A est une *sous-formule* de B si l'arbre syntaxique de A est un sous-arbre de l'arbre syntaxique de B.
- A est une *sous-formule propre* de B si A est une sous-formule de B, mais A n'est pas identique à B.

Exemples :
- $p \Rightarrow q$ est une sous-formule propre de $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$;
- $p \Rightarrow q$ est une sous-formule impropre de $p \Rightarrow q$;
- $q \equiv \neg q$ n'est pas une sous-formule de $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$.

Remarque. Une sous-formule peut avoir plusieurs occurrences dans une formule. Par exemple, la (sous-)formule $\neg p$ a deux occurrences dans la formule $\neg p \equiv \neg(p \Leftarrow \neg p)$; la (sous-)formule p a trois occurrences dans la formule $p \equiv (p \Leftarrow \neg p)$.

Le cas où $C_B = C_A$ (zéro occurrence remplacée) est évident.

Il suffit de démontrer le théorème dans le cas où C_B s'obtient par remplacement d'une seule occurrence.

Démonstration. On raisonne par *induction* sur la profondeur d de la racine du sous-arbre syntaxique A dans l'arbre C_A.

Soit v une interprétation telle que $v(A) = v(B)$.
- $d = 0$: $A = C_A$ et $B = C_B$, donc $v(C_A) = v(C_B)$.
- $d > 0$: C_A est de la forme $\neg D_A$ ou $(D_A \ op \ E_A)$;
  C_B est alors $\neg D_B$ ou $(D_B \ op \ E_B)$.
  (Dans le second cas, l'un des termes D_A, E_A comporte l'occurrence de A à remplacer.)

  Si l'occurrence de A est dans D_A, la profondeur de A est strictement inférieure à d, donc $v(D_B) = v(D_A)$, par hypothèse inductive (si l'occurrence n'est pas dans D_A, c'est trivial) ; de même, on établit $v(E_B) = v(E_A)$ (s'il y a lieu).

  Comme les valeurs de vérité des formules C_A et C_B ne dépendent que des valeurs de vérité de leurs composantes, soient $v(D_B) = v(D_A)$ et (éventuellement) $v(E_B) = v(E_A)$, on a $v(C_B) = v(C_A)$. □

$$(X \wedge X) \longleftrightarrow X \longleftrightarrow (X \vee X)$$
$$(X \wedge Y) \longleftrightarrow (Y \wedge X)$$
$$(X \vee Y) \longleftrightarrow (Y \vee X)$$
$$((X \wedge Y) \wedge Z) \longleftrightarrow (X \wedge (Y \wedge Z))$$
$$((X \vee Y) \vee Z) \longleftrightarrow (X \vee (Y \vee Z))$$

$$(X \Rightarrow X) \longleftrightarrow \mathit{true}$$
$$((X \Rightarrow Y) \wedge (Y \Rightarrow X)) \longleftrightarrow (X \equiv Y)$$
$$(((X \Rightarrow Y) \wedge (Y \Rightarrow Z)) \Rightarrow (X \Rightarrow Z)) \longleftrightarrow \mathit{true}$$
$$(X \Rightarrow Y) \longleftrightarrow ((X \wedge Y) \equiv X)$$
$$(X \Rightarrow Y) \longleftrightarrow ((X \vee Y) \equiv Y)$$

$$(X \wedge (Y \vee Z)) \longleftrightarrow ((X \wedge Y) \vee (X \wedge Z))$$
$$(X \vee (Y \wedge Z)) \longleftrightarrow ((X \vee Y) \wedge (X \vee Z))$$
$$(X \Rightarrow (Y \Rightarrow Z)) \longleftrightarrow ((X \Rightarrow Y) \Rightarrow (X \Rightarrow Z))$$

$$(X \vee \neg X) \longleftrightarrow \mathit{true}$$
$$(X \wedge \neg X) \longleftrightarrow \mathit{false}$$
$$\neg\neg X \longleftrightarrow X$$

$$\neg(X \wedge Y) \longleftrightarrow (\neg X \vee \neg Y)$$
$$\neg(X \vee Y) \longleftrightarrow (\neg X \wedge \neg Y)$$

– On utilise souvent ces lois algébriques, combinées au théorème de l'échange, pour simplifier les formules.

Exemple :

$$\begin{array}{rcl} p \wedge (\neg p \vee q) & \leftrightarrow & (p \wedge \neg p) \vee (p \wedge q) \\ & \leftrightarrow & \mathsf{false} \vee (p \wedge q) \\ & \leftrightarrow & p \wedge q \end{array}$$

– Ces équivalences décrivent des propriétés des connecteurs.
  – associativité, commutativité de $\wedge$, $\vee$, $\equiv$
  – idempotence de $\wedge$, $\vee$
  – ...

– Tous les connecteurs peuvent s'obtenir à partir de
  – $\neg$ et $\wedge$
  – *Nand* seul
  – *Nor* seul

Systèmes minimaux de connecteurs

– $\neg$ et $\vee$:
  $(a \Rightarrow b) =_{def} (\neg a \vee b)$,
  $(a \wedge b) =_{def} \neg(\neg a \vee \neg b)$.
– $\neg$ et $\wedge$:
  $(a \Rightarrow b) =_{def} \neg(a \wedge \neg b)$,
  $(a \vee b) =_{def} \neg(\neg a \wedge \neg b)$.
– $\neg$ et $\Rightarrow$:
  $(a \vee b) =_{def} (\neg a \Rightarrow b)$,
  $(a \wedge b) =_{def} \neg(a \Rightarrow \neg b)$.
– "*nand*" seul (symbole $\uparrow$)
  $\neg a =_{def} (a \uparrow a)$, $(a \wedge b) =_{def} \neg(a \uparrow b)$.
– "*nor*" seul (symbole $|$, ou $\downarrow$)
  $\neg a =_{def} (a \mid a)$, $(a \vee b) =_{def} \neg(a \mid b)$.

Substitution uniforme (principe)

Soient A_1 et A_2 des formules et B la formule $A_1 \Rightarrow (A_1 \vee A_2)$. La formule B peut être très longue, mais on "voit" immédiatement qu'elle est valide, comme "instance" de la formule valide $p \Rightarrow (p \vee q)$.

Si C est une formule, on note $C(p/A_1, q/A_2)$ la formule obtenue en remplaçant *toutes* les occurrences des propositions p et q dans C par les formules A_1 et A_2, respectivement.

Lemme. Soient $C, A_1, \ldots, A_n$ des formules et $p_1, \ldots, p_n$ des propositions deux à deux distinctes. Si v est une interprétation telle que $v(p_i) = v(A_i)$ $(i = 1, \ldots, n)$, alors $v(C(p_1/A_1, \ldots, p_n/A_n)) = v(C)$.
Remarque. L'interprétation v est définie sur un lexique comportant toute proposition intervenant dans C ou dans l'un des A_i.
Remarque. Les formules $C(p/A_1, q/A_2)$, $C(p/A_1)(q/A_2)$ et $C(q/A_2)(p/A_1)$ peuvent être distinctes !

Substitution uniforme (exemple)

$n =_{def} 2$
$C =_{def} p_1 \vee (q \Rightarrow p_2)$
$A_1 =_{def} p_2 \wedge (p_1 \vee r)$
$A_2 =_{def} p_1 \vee q$
$C(p_1/A_1, p_2/A_2) =_{def}$
$\quad (p_2 \wedge (p_1 \vee r)) \vee (q \Rightarrow (p_1 \vee q))$
$v =_{def} \{(p_1, F), (p_2, T), (q, T), (r, F)\}$

On a $v(A_1) = v(p_1) = F$ et
$\quad v(A_2) = v(p_2) = T$;
on a aussi $v(C(p_1/A_1, p_2/A_2)) = v(C) = T$.

Remarque. Le plus souvent, on se restreint au cas où aucune proposition p_i n'intervient dans les éléments de l'ensemble $\{A_1, \ldots, A_n\}$. Dans ce cas, les formules $C(p_1/A_1, \ldots, p_n/A_n)$ et $C(p_1/A_1) \ldots (p_n/A_n)$ sont nécessairement identiques.

Théorème de substitution uniforme

Théorème. Soient $C, A_1, \ldots, A_n$ des formules et $p_1, \ldots, p_n$ des propositions deux à deux distinctes. Si C est une tautologie, alors $C(p_1/A_1, \ldots, p_n/A_n)$ est une tautologie.

Démonstration. On suppose d'abord qu'aucun p_i n'apparaît dans $\{A_1, \ldots, A_n\}$ ni, par conséquent, dans $C' =_{def} C(p_1/A_1, \ldots, p_n/A_n)$. Soit v, une interprétation quelconque de C', et w l'*extension* de v obtenue en posant $w(p_i) =_{def} v(A_i)$. Le lemme de substitution implique $w(C') = w(C)$. Par hypothèse on a $w(C) = T$ et par construction on a $w(C') = v(C')$. On a donc $v(C') = T$.

Remarque. Si les p_i intervenaient dans les A_k, la technique pourrait ne pas fonctionner. De $\models p \equiv \neg\neg p$, on ne déduit pas immédiatement que $\models (p \vee r) \equiv \neg\neg(p \vee r)$ car l'interprétation $v : v(p) = F, v(r) = T$ telle que $v(A) = v(p \vee r) = T$ n'admet pas d'extension w telle que $w(p) = T$. Le remède est simple. De $\models p \equiv \neg\neg p$ on déduit $\models q \equiv \neg\neg q$, d'où on déduit $\models (p \vee r) \equiv \neg\neg(p \vee r)$.

Si par contre les p_i interviennent dans $\{A_1, \ldots, A_n\}$, on se donne une famille de nouveaux atomes q_i. Si C est une tautologie, alors $C'' =_{def} C(p_1/q_1, \ldots, p_n/q_n)$ est une tautologie. D'autre part, C' peut s'écrire $C''(q_1/A_1, \ldots, q_n/A_n)$, où les q_i n'interviennent pas dans les A_k ; C' est donc une tautologie.

Substitution uniforme (démonstration du lemme)

On procède par induction sur la *structure* de la formule C.
On note C' la formule $C(p_1/A_1, \ldots, p_n/A_n)$.

Cas de base. C est une proposition.
Si C est l'un des p_i, on a $v(C') = v(A_i) = v(p_i) = v(C)$;
sinon on a $C' = C$, d'où $v(C') = v(C)$.

Cas inductifs. Si C est la formule $\neg D$, alors C' est la formule $\neg D'$.
Par hypothèse inductive, on a $v(D') = v(D)$, d'où $v(C') = v(C)$.
De même, si C est la formule $D \vee E$, alors C' est la formule $D' \vee E'$, et un raisonnement analogue s'applique ; il en va de même pour les autres connecteurs binaires. □

Méthode des tables de vérité

Toute formule contient un nombre fini d'atomes : elle a un nombre fini d'interprétations. On peut donc déterminer la valeur de vérité de la formule pour toutes ses interprétations.

Algorithme très inefficace !
Puisque si la formule contient n atomes,
il y a 2^n interprétations, l'algorithme est exponentiel en fonction du nombre de propositions intervenant dans la formule.

Le problème de la satisfaisabilité en logique des propositions est NP-complet. On ne s'attend donc pas à trouver un algorithme polynomial mais, en pratique, il y a des algorithmes meilleurs que la méthode des tables de vérité.

Exemples.

– Table de vérité de $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$:

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	T
F	F	T	T	T

→ formule valide

– Table de vérité de $p \wedge q$:

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

→ formule satisfaisable, mais non valide

– *Table de vérité* de $(p \vee q) \wedge \neg p \wedge \neg q$:

p	q	$p \vee q$	$\neg p$	$\neg q$	$(p \vee q) \wedge \neg p \wedge \neg q$
T	T	T	F	F	F
T	F	T	F	T	F
F	T	T	T	F	F
F	F	F	T	T	F

→ formule insatisfaisable

On peut améliorer la méthode des tables de vérité en utilisant diverses simplifications.

Exemple.

Formule :

$$(p \Rightarrow q) \vee (q \Rightarrow r) \vee (r \Rightarrow p)$$

Interprétation :

$$v(p) = F,\ v(q) = T,\ v(r) = F$$

Evaluation complète :

$$\begin{gathered}(F \Rightarrow T) \vee (T \Rightarrow F) \vee (F \Rightarrow F)\\ T \vee (T \Rightarrow F) \vee (F \Rightarrow F)\\ T \vee F \vee (F \Rightarrow F)\\ T \vee F \vee T\\ T \vee T\\ T\end{gathered}$$

Simplification :

$$\begin{gathered}(F \Rightarrow q) \vee (q \Rightarrow r) \vee (r \Rightarrow p)\\ T \vee \cdots\\ T\end{gathered}$$

LES TABLEAUX SEMANTIQUES

Procédure de décision pour la satisfaisabilité (consistance) dans le calcul des propositions.

Relativement efficace
(plus que la méthode des tables de vérité).

Principe : Pour déterminer la consistance,
chercher *systématiquement* un modèle.

Tables de vérité : de l'intérieur vers l'extérieur : la valeur de vérité d'un composé est *fonction* de la valeur de vérité des composants.

Tableaux sémantiques : de l'extérieur vers l'intérieur : valeur de vérité des composants à partir de la valeur de vérité du composé, mais pas *fonction* !

Principe de la méthode

Un *littéral* est un atome ou la négation d'un atome. Si p est une proposition atomique, $\{p, \neg p\}$ est une *paire complémentaire de littéraux*. Un ensemble de littéraux est consistant si et seulement s'il est dépourvu de paire complémentaire (ce qui se détermine par simple inspection).

La méthode des tableaux sémantiques consiste à réduire la question

La formule A est-elle consistante ?

à la question beaucoup plus facile

La famille (finie) $\mathcal{A}$ d'ensembles de littéraux contient-elle un élément consistant ?

Une formule peut donner lieu à plusieurs tableaux sémantiques différents suivant l'ordre d'application des règles de construction, mais tous conduisent à la même conclusion (la bonne !) concernant la consistance de la formule.

Technique de construction

La construction des tableaux sémantiques est basée sur la partition des formules en trois catégories :
- les littéraux ;
- les formules conjonctives ;
- les formules disjonctives.

La formule $\neg(X \Rightarrow Y)$ est conjonctive car elle est équivalente à la conjonction des deux formules (plus simples) X et $\neg Y$. La formule $X \Rightarrow Y$ est disjonctive car elle est équivalente à la disjonction de $\neg X$ et Y. Le connecteur $\equiv$ est exclu ici, et $\neg\neg X$ a pour unique composant X (conjonctif ou disjonctif, au choix).

La construction est basée sur deux types de règles de décomposition de formules : les règles de *prolongation* (type α) et les règles de *ramification* (type β).

Dans le contexte des tableaux sémantiques on utilise
les règles α pour les formules conjonctives ;
les règles β pour les formules disjonctives.
Attention ! Ce point sera nuancé plus loin.

Exemples I

Considérons $A = p \wedge (\neg q \vee \neg p)$.
La formule est conjonctive, de type α ;
ses deux composants sont p et $\neg q \vee \neg p$.
Graphiquement, on a :

$$p \wedge (\neg q \vee \neg p)$$
$$\downarrow$$
$$p, \neg q \vee \neg p$$

La signification est : "v est un modèle de A si et seulement si v est un modèle de p **et** de $\neg q \vee \neg p$".

La formule $\neg q \vee \neg p$ est disjonctive, de type β ;
ses deux composants sont $\neg q$ et $\neg p$.
Graphiquement, on a :

$$p, \neg q \vee \neg p$$
$$\swarrow \qquad \searrow$$
$$p, \neg q \qquad p, \neg p$$

La signification est : "v est un modèle de p et de $\neg q \vee \neg p$ si et seulement si v est un modèle de p et de $\neg q$ **ou** de p et de $\neg p$".

Enfin, l'ensemble $\{p, \neg q\}$ est consistant (symbole $\bigcirc$, *feuille ouverte*) tandis que $\{p, \neg p\}$ ne l'est pas (symbole $\times$, *feuille fermée*). La représentation complète est

$$p \wedge (\neg q \vee \neg p)$$
$$\downarrow$$
$$p, \neg q \vee \neg p$$
$$\swarrow \qquad \searrow$$
$$p, \neg q \qquad p, \neg p$$
$$\bigcirc \qquad \times$$

Exemples II

Conclusion. L'arbre est ouvert (sa racine est consistante) si et seulement si l'une des feuille est ouverte (consistante). C'est le cas ici : A est consistante parce que $\{p, \neg q\}$ l'est ; un modèle commun est $p = T$, $q = F$.

Considérons $B = (p \vee q) \wedge (\neg p \wedge \neg q)$.
La formule est conjonctive, de type α ;
ses deux composants sont
$B_1 =_{def} p \vee q$ et $B_2 =_{def} \neg p \wedge \neg q$.

La formule B_1 est disjonctive, de type β ;
ses deux composants sont p et q.

La formule B_2 est conjonctive, de type α ;
ses deux composants sont $\neg p$ et $\neg q$.

On obtient le tableau de gauche si on traite B_1 avant B_2, celui de droite sinon. La conclusion est naturellement la même dans les deux cas : toutes les feuilles de l'arbre sont fermées, le tableau est fermé, la racine B est inconsistante.

$$\begin{array}{c} (p \vee q) \wedge (\neg p \wedge \neg q) \\ \downarrow \\ p \vee q, \neg p \wedge \neg q \\ \swarrow \qquad \searrow \\ p, \neg p \wedge \neg q \qquad q, \neg p \wedge \neg q \\ \downarrow \qquad\qquad \downarrow \\ p, \neg p, \neg q \qquad q, \neg p, \neg q \\ \times \qquad\qquad \times \end{array} \qquad \begin{array}{c} (p \vee q) \wedge (\neg p \wedge \neg q) \\ \downarrow \\ p \vee q, \neg p \wedge \neg q \\ \downarrow \\ p \vee q, \neg p, \neg q \\ \swarrow \qquad \searrow \\ p, \neg p, \neg q \qquad q, \neg p, \neg q \\ \times \qquad\qquad \times \end{array}$$

Règles de décomposition

On distingue deux types de formules :
- les α-*formules* (conjonctives) donnent lieu à une *prolongation* du tableau ; $v(\alpha) = T$ si et seulement si $v(\alpha_1) = v(\alpha_2) = T$.

α	α_1	α_2
$\neg\neg A$	A	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \Rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \Leftarrow A_2)$	$\neg A_1$	A_2

- les β-*formules* (disjonctives) donnent lieu à une *ramification* ; $v(\beta) = T$ si et seulement si $v(\beta_1) = T$ ou $v(\beta_2) = T$.

β	β_1	β_2
$B_1 \vee B_2$	B_1	B_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \Rightarrow B_2$	$\neg B_1$	B_2
$B_1 \Leftarrow B_2$	B_1	$\neg B_2$

Construction d'un tableau sémantique pour une formule donnée C.

Chaque nœud sera étiqueté d'un ensemble de formules.

Initialisation : racine étiquetée $\{C\}$.

Réitérer l'étape inductive suivante : sélectionner une feuille non marquée, soit ℓ étiquetée $U(\ell)$.

- Si $U(\ell)$ est un ensemble de littéraux :
  - si $U(\ell)$ contient une paire complémentaire, alors marquer ℓ comme étant *fermée* '$\times$' ;
  - sinon, marquer ℓ comme étant *ouverte* '$\bigcirc$'.
- Si $U(\ell)$ n'est pas un ensemble de littéraux, sélectionner une formule dans $U(\ell)$:
  - si c'est une α-formule A, créer un nouveau nœud ℓ', descendant de ℓ, et étiqueter ℓ' avec
$$U(\ell') = (U(\ell) - \{A\}) \cup \{\alpha_1, \alpha_2\};$$
  - si c'est une β-formule B, créer deux nouveaux nœuds ℓ' et ℓ'', descendants de ℓ, et étiqueter ℓ' avec
$$U(\ell') = (U(\ell) - \{B\}) \cup \{\beta_1\}$$
    et étiqueter ℓ'' avec
$$U(\ell'') = (U(\ell) - \{B\}) \cup \{\beta_2\}.$$

Terminaison :
quand toutes les feuilles sont marquées '$\times$' ou '$\bigcirc$'.

Terminaison

Théorème. La construction d'un tableau sémantique se termine.

Remarque. On peut, sans restriction essentielle, supposer que le connecteur d'équivalence n'est pas employé. Cette restriction simplifie la forme de la mesure W définie dans la démonstration.

Démonstration. Soit T le tableau d'une formule A,
à une étape quelconque de sa construction.

Soit ℓ une feuille quelconque de T.

Soit $b(\ell)$ le nombre de connecteurs binaires dans $U(\ell)$
et $n(\ell)$ le nombre de négations dans $U(\ell)$.

On définit $W(\ell) = 2b(\ell) + n(\ell)$.

On peut vérifier que quelle que soit la règle utilisée, toute étape de la construction crée un nouveau nœud ℓ' ou deux nouveaux nœuds ℓ', ℓ'' tels que $W(\ell') < W(\ell)$ et $W(\ell'') < W(\ell)$.

Or, $W(\ell)$ prend ses valeurs dans **N**.

Donc, il ne peut y avoir de branche infinie dans T.

La démonstration peut être étendue au cas où "$\equiv$" apparaît dans la formule (exercice). □

Un tableau *complet* (dont la construction est achevée) est *fermé* si toutes ses feuilles sont fermées. Sinon, il est *ouvert*.

Adéquation et complétude d'une méthode de preuve

Adéquation (soundness) : toute formule dont la méthode permet de déduire qu'elle est valide est effectivement valide.

Complétude (completeness) : si une formule est valide, alors la méthode produit une preuve de sa validité.

Une méthode est adéquate si elle est correcte ;
elle est complète si elle est assez puissante.

Méthode des tableaux sémantiques

Adéquation :
Si $T(A)$ est fermé, alors A est inconsistante ;
si $T(\neg B)$ est fermé, alors B est valide.

Complétude :
Si A est inconsistante, alors $T(A)$ est fermé ;
si B est valide, alors $T(\neg B)$ est fermé.

Adéquation de la méthode des tableaux sémantiques I

Si $T(A)$ est fermé, A est inconsistante.

Démonstration. On suppose que l'arbre $T(A)$ est fermé (tous ses sous-arbres le sont aussi). On démontre par induction sur la hauteur h du nœud n dans $T(A)$ que pour tout sous-arbre de racine n de $T(A)$ l'ensemble $U(n)$ est inconsistant. La hauteur d'une feuille est 0 ; la hauteur d'un nœud α est celle de son fils, plus 1 ; la hauteur d'un nœud β est celle du plus haut de ses fils, plus 1.

- $h = 0$: n est une feuille fermée
  donc $U(n)$ contient une paire complémentaire de littéraux et $U(n)$ est inconsistant.
- $h > 0$: une règle α ou une règle β a été utilisée pour créer le(s) descendant(s) de n.

  Règle α : n : $\{\alpha\} \cup U_0$
  $$\downarrow$$
  n' : $\{\alpha_1, \alpha_2\} \cup U_0$

  On a $h(n') < h(n)$ et l'hypothèse inductive s'applique au sous-arbre (fermé) de racine n' ; l'ensemble $U(n')$ est inconsistant.
  Pour toute interprétation v,
  il y a une formule $A' \in U(n')$ telle que $v(A') = F$.

Adéquation de la méthode des tableaux sémantiques II

Trois possibilités pour A' :

1. soit $A' \in U_0 \subseteq U(n)$;
2. soit $A' = \alpha_1$: $v(\alpha_1) = F$ d'où $v(\alpha) = F$
   (cfr. règles α) ;
3. soit $A' = \alpha_2$: $v(\alpha_2) = F$ d'où $v(\alpha) = F$.

Dans les trois cas, il y a une formule dans $U(n)$ que v rend fausse ; $U(n)$ est donc inconsistant
(v étant quelconque).

Règle β :

$$\begin{array}{ccc} & n : \{\beta\} \cup U_0 & \\ \swarrow & & \searrow \\ n' : \{\beta_1\} \cup U_0 & & n'' : \{\beta_2\} \cup U_0 \end{array}$$

$h(n') < h(n)$ et $h(n'') < h(n)$; les ensembles $U(n')$ et $U(n'')$ sont tous deux inconsistants
Pour toute interprétation v,

1. soit il y a une formule $A' \in U_0 \subseteq U(n)$: $v(A') = F$
2. soit $v(\beta_1) = v(\beta_2) = F$,
   d'où $v(\beta) = F$ (cfr. règles β).

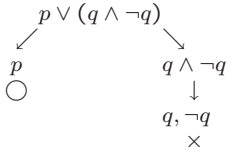
Dans les deux cas, il y a une formule dans $U(n)$ que v rend fausse ; on en déduit que $U(n)$ est inconsistant (v étant quelconque). □

Ensembles de Hintikka

Définition : Soit U un ensemble de formules. U est un *ensemble de Hintikka* si les trois conditions suivantes sont satisfaites :

1. Pour tout atome p, $p \notin U$ ou $\neg p \notin U$
2. Si $\alpha \in U$ est une α-formule,
   alors $\alpha_1 \in U$ et $\alpha_2 \in U$.
3. Si $\beta \in U$ est une β-formule,
   alors $\beta_1 \in U$ ou $\beta_2 \in U$.

Exemple :



$U = \{p, p \vee (q \wedge \neg q)\}$ est un ensemble de Hintikka.

On va démontrer que la réunion des ensembles étiquetant les nœuds d'une branche ouverte (chemin joignant la racine à une feuille ouverte) forme un ensemble de Hintikka, et qu'un ensemble de Hintikka est consistant.

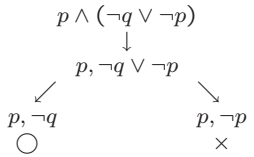
Complétude de la méthode des tableaux sémantiques

Si A est inconsistante, $T(A)$ est fermé.

Démonstration. On suppose que $T(A)$ est ouvert et
on démontre que A est consistante (contraposition).

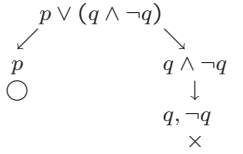
Idée : Toute feuille ouverte de $T(A)$ caractérise un modèle de A.

Exemple : $A =_{def} p \wedge (\neg q \vee \neg p)$



Interprétation $v(p) = T$, $v(q) = F$: modèle de A.

Exemple : $B = p \vee (q \wedge \neg q)$



Interprétation $v(p) = T$: modèle de A ? Oui, à condition d'assigner une valeur (arbitraire) à q.

Lemme de la branche ouverte

Soit *b* une branche ouverte d'un tableau complet T. L'ensemble $U =_{def} \bigcup_{n \in b} U(n)$ est un ensemble de Hintikka.

Démonstration. On montre que U respecte les trois conditions caractérisant les ensembles de Hintikka.

1. On note ℓ la feuille (ouverte) de *b*.
   Pour tout littéral m, ($m \in \{p, \neg p\}$)
   $m \in U$ implique $m \in U(\ell)$
   (aucune règle ne décompose les littéraux).
   Or, ℓ est un nœud ouvert, sans paire complémentaire.
   On a donc $p \notin U$ ou $\neg p \notin U$.
2. Pour toute α-formule $\alpha \in U$, l'arbre étant complet, la règle α correspondante a dû être utilisée à un certain nœud n. Par construction, $\alpha_1, \alpha_2 \in U(n') \subseteq U$.
3. Pour toute β-formule $\beta \in U$, l'arbre étant complet, la règle β correspondante a dû être utilisée à un certain nœud n. Par construction, $\beta_1 \in U(n')$ et $\beta_2 \in U(n'')$, et $U(n') \subseteq U$ ou $U(n'') \subseteq U$, d'où $\beta_1 \in U$ ou $\beta_2 \in U$. □

Lemme de Hintikka

Tout ensemble de Hintikka est consistant.

Démonstration. Soit U un ensemble de Hintikka et soit $\mathcal{P} = \{p_1, \ldots, p_m\}$ l'ensemble des atomes apparaissant dans les formules de U.

Définissons une interprétation v de U :

$$v(p) = T \quad \text{si} \quad p \in U \text{ ou } \neg p \notin U$$
$$v(p) = F \quad \text{si} \quad \neg p \in U$$

v assigne une et une seule valeur de vérité à chaque atome de $\mathcal{P}$ (car U est un ensemble de Hintikka).

Il faut démontrer que pour tout $A \in U$: $v(A) = T$.

Par induction sur la structure de A :

A est un littéral. Par définition de v on a :
- Si $A = p$, alors $v(A) = v(p) = T$.
- Si $A = \neg p$, alors $v(p) = F$, d'où $v(A) = T$.

A est une α-formule α : $\alpha_1, \alpha_2 \in U$, donc par hypothèse inductive, $v(\alpha_1) = T$ et $v(\alpha_2) = T$,
d'où $v(\alpha) = T$ par définition des règles α.

A est une β-formule β : $\beta_1 \in U$ ou $\beta_2 \in U$, donc par hypothèse inductive, $v(\beta_1) = T$ ou $v(\beta_2) = T$,
d'où $v(\beta) = T$ par définition des règles β. □

Résumé

La formule A est inconsistante si et seulement si $T(A)$ est un tableau fermé ; la formule B est valide si et seulement si $T(\neg B)$ est un tableau fermé ; la formule C est simplement consistante si et seulement si $T(C)$ et $T(\neg C)$ sont des tableaux ouverts.

La méthode des tableaux sémantiques est un algorithme de décision pour la validité (la consistance, l'inconsistance) dans le calcul des propositions.

Complétude de la méthode des tableaux sémantiques

Théorème.
Si A est inconsistante, $T(A)$ est fermé.

Démonstration. (Par contraposition.)
Si $T(A)$ ouvert, A est consistante.

Si $T(A)$ est ouvert, il existe une branche ouverte dans $T(A)$. la réunion des ensembles de formules étiquetant les nœuds de cette branche forme un ensemble de Hintikka (donc consistant) ; cet ensemble contient la formule A, qui est donc consistante. □

En pratique

Si on présuppose qu'une formule X est inconsistante, on construira d'abord $T(X)$;
si on présuppose qu'elle est valide,
on construira d'abord $T(\neg X)$.

Si le tableau $T(Y)$ est fermé, l'analyse est terminée : on sait que Y est inconsistant et que $\neg Y$ est valide.

Si le tableau $T(Z)$ est ouvert, on sait que Z est consistant et que $\neg Z$ n'est pas valide ; il faut construire $T(\neg Z)$ pour en savoir plus.

Simplifications de la construction :

- Une branche peut être fermée lorsqu'une paire complémentaire de formules (pas seulement de littéraux) apparaît.
- Les formules inchangées ne doivent pas être recopiées d'un nœud à son descendant (économie d'espace).
- Des heuristiques peuvent éventuellement écourter le tableau (par exemple, utiliser les règles α avant les règles β).