Logique

Christophe Fouqueré

Licence Informatique 1ère année

christophe.fouquere@univ-paris13.fr
A204, Université de Paris 13

Bibliographie (très) partielle :

- **Logique**, *[Chap. 1, 2, 3, 4]* Gochet et Gribomont, Hermès, 1990
- **Introduction à la logique**, *[Chap. 1 et 5]* David, Nour, Raffalli, Dunod Informatiques, 2001

I : Introduction
1°) Logique et raisonnement

La logique :

- au sens classique : science des lois de la pensée
- au sens moderne : logique formelle ou logique symbolique ou logique mathématique

Dans ce cours, nous nous intéresserons à la *logique mathématique*.

La Logique a pour objet de dégager les règles du **raisonnement correct** : les règles qui permettent de tirer des conclusions certaines à partir d'hypothèses données.

Exemple (R1)

Si Albert est chez lui, alors la lumière est allumée.
Or la lumière n'est pas allumée.
Donc Albert n'est pas chez lui.

Du point de vue logique, le raisonnement (R1) a rigoureusement la même valeur que le raisonnement (R2) :

Exemple (R1)

Si Albert est chez lui, alors la lumière est allumée.
Or la lumière n'est pas allumée.
Donc Albert n'est pas chez lui.

Exemple (R2)

Si Bernard est un martien, alors il a des antennes vertes.
Or Bernard n'a pas d'antennes vertes.
Donc il n'est pas un martien.

➡ Analyser les raisonnements du point de vue de leur **structure formelle**.

A-t-on le même contenu objectif dans les cas suivants ?

Exemples

- Nul courage n'est une vertu // Nulle vertu n'est un courage
- Chaque crime est un délit // Chaque délit est un crime
- Chacune de nos agences qui n'a pas de service internet a un service Relation Publique // Chacune de nos agences qui n'a pas de service Relation Publique a un service internet
- Chacune de nos agences qui a un distributeur de billets a un service informatique // Chacune de nos agences qui a un service informatique a un distributeur de billets

➡ Nécessité de savoir analyser et comprendre des énoncés (spécification, programmation, . . .).

I : Introduction
2°) Concepts et cadres

Concepts clés :

- *proposition* ou *énoncé*
- *assertion*
- *vérité*
- *preuve*

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

➨ La logique est l'étude des **connexions** entre **propositions déclaratives**.

I : Introduction
2°) Concepts et cadres

Concepts clés :

- **proposition** ou **énoncé**
  - interrogative : *quelle est la somme des 10 premiers entiers ?*
  - impérative : *mettre 96 dans la cellule-mémoire 32087.*
  - **déclarative** : $\sqrt{2}$ *n'est pas rationnel.*
- *assertion*
- *vérité*
- *preuve*

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

➡ La logique est l'étude des **connexions** entre **propositions déclaratives**

I : Introduction
2°) Concepts et cadres

Concepts clés :

- *proposition* ou *énoncé*
- **assertion**
  - proposition déclarative présentée comme vraie
- *vérité*
- *preuve*

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

➡ La logique est l'étude des **connexions** entre **propositions déclaratives**.

I : Introduction
2°) Concepts et cadres

Concepts clés :

- *proposition* ou *énoncé*
- *assertion*
- **vérité**
  - d'une assertion, ou abusivement d'une proposition
  - *1+1=0* est une assertion *vraie* en calcul sur 0 et 1
  - *1+1=0* est une assertion *fausse* en calcul sur les entiers naturels
  - *il est vrai qu'il va pleuvoir demain* est une assertion vraie ? fausse ?
- *preuve*

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

I : Introduction
2°) Concepts et cadres

Concepts clés :

- *proposition* ou *énoncé*
- *assertion*
- *vérité*
- **preuve**
  - démonstration d'une assertion,
  - suite des assertions justifiant (mécaniquement) un résultat

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

➡ La logique est l'étude des **connexions** entre **propositions déclaratives**.

I : Introduction
2°) Concepts et cadres

Concepts clés :

- *proposition* ou *énoncé*
- *assertion*
- *vérité*
- *preuve*

Cadres clés :

- *morphologie* : définition du langage des propositions,
- *sémantique* : définition des valeurs de vérités,
- *syntaxe* : définition des structures de preuves.

➡ La logique est l'étude des **connexions** entre **propositions déclaratives**.

- Avant Platon et Aristote : idée de concevoir une *démonstration* comme une succession de pas tels que le passage d'un maillon au suivant ne laisse place à aucun doute.
- Dès le 5ème siècle av.JC, les philosophes cherchent à dégager les principes généraux servant de base aux raisonnements :
  - *Parménide* énonce le principe du tiers exclu,
  - *Zénon d'Elée* énonce le principe du raisonnement par l'absurde.

(voir démonstrations de l'irrationnalité de $\sqrt{2}$ et du fait qu'il existe deux nombres irrationnels x et y tels que x^y soit rationnel)

- Avec Aristote et les Stoïciens :
  - utilisation de variables : “Un homme est mortel” devient le schéma formel de la proposition : “A est B” (voir logique des prédicats, logique des propositions)
  - étude des syllogismes, raisonnement logique à deux propositions (également appelées prémisses) conduisant à une conclusion : Les deux prémisses (dites " majeure" et "mineure ") sont des propositions données et supposées vraies permettant de vérifier la véracité formelle de la conclusion.

Exemple

Tous les hommes sont mortels, or les Grecs sont des hommes, donc les Grecs sont mortels

Un langage logique : une formule est de la forme

- $R(t_1, \ldots, t_n)$ où R est une constante prédicative, t_i est soit une variable, soit une constante
- $F \vee G$: F ou G
- $F \wedge G$: F et G
- $F \Rightarrow G$: F implique G
- $\neg F$: non F
- $\forall x F$: pour tout x, F
- $\exists x F$: il existe x tel que F

où F et G sont des formules, x est une variable
Constante prédicative : verbe d'une phrase ou attribut.

Mise en formule :

- Pierre est une personne : $Personne(Pierre)$
- Marie aime Pierre : $Aime(Marie, Pierre)$
- Marie aime une personne :
  $\exists x(Personne(x) \wedge Aime(Marie, x))$
- Marie aime une personne mais elle n'aime pas Pierre :
  $\exists x(Personne(x) \wedge Aime(Marie, x)) \wedge \neg Aime(Marie, Pierre)$
- Toutes les personnes que Marie aime, Anne les aime aussi :
  $\forall x((Personne(x) \wedge Aime(Marie, x)) \Rightarrow Aime(Anne, x))$

III : Logique propositionnelle

- Leibniz (XVIIème, formalisme, $\forall$, $\exists$), Boole (XIXème, utilisation de l'algèbre)
- Hilbert (XIX-XXème) : axiomatisation des mathématiques
- Début du XXème siècle, "crise" des mathématiques (voir le paradoxe de Russell), puis deux démarches :
  - projet logistique : théorie des types,
  - projet intuitionniste et calcul fonctionnel : informatique.

Logique = raisonnement "correct" =

Démonstration mathématique
- déduction (modus ponens)
- traitement par cas
- induction (récurrence)

- contraposée
- contradiction (absurde)

Programme informatique
- appel de fonction
- case
- boucle, tant-que, appel récursif

Le **calcul propositionnel** s'intéresse aux énoncés du langage mathématique.

1. Tout carré est un rectangle.
2. La médiatrice d'un segment est la droite qui lui est orthogonale et qui passe par le milieu de ce segment.
3. $\sqrt{2}$ n'est pas un nombre rationnel.
4. Il existe deux nombres irrationnels x et y tels que x^y est rationnel.

Le calcul propositionnel ne s'intéresse pas au contenu des propositions. Il s'intéresse seulement à la manière dont les propositions sont articulées par des *connecteurs logiques* :

- négation,
- conjonction,
- implication
- ...

➡ On s'intéresse à la manière dont les propositions se construisent/décomposent, à partir/en fonction de propositions plus élémentaires.

IV : Les formules propositionnelles

1°) Définition

Dans un premier temps il s'agit de définir précisément nos objets d'étude : **les propositions** ou **formules propositionnelles**.
Nous allons manipuler

- des symboles de **variables propositionnelles** ou **formules atomiques** ou **atomes** A, B, C, ...
- des symboles de combinateurs ou opérations logiques appelés **connecteurs** :
  - un connecteur unaire pour la négation, noté $\neg$ et lu "non",
  - un connecteur binaire pour la conjonction, noté $\wedge$ et lu "et",
  - un connecteur binaire pour la disjonction, noté $\vee$ et lu "ou",
  - un connecteur binaire pour l'implication, noté $\Rightarrow$ et lu "implique"
  - un connecteur binaire pour l'équivalence, noté $\Leftrightarrow$ et lu "si et seulement si".

Plus précisément :

Définition

On appelle **alphabet** un ensemble fini ou dénombrable.
Soit $\mathcal{A}$ un alphabet, un mot m sur $\mathcal{A}$ est une suite finie d'éléments de $\mathcal{A}$.
L'ensemble des mots de $\mathcal{A}$ est noté $\mathcal{A}^*$. Le mot vide est noté ϵ.

Définition

Soit un ensemble P fini ou dénombrable de variables propositionnelles, $P = \{A, B, \ldots, X, Y, Z \ldots\}$.
On considère l'alphabet $\mathcal{A} = P \cup \{\Rightarrow, \vee, \wedge, \neg, \Leftrightarrow\} \cup \{(\ ,\ )\}$.
L'ensemble $\mathcal{F}$ des **formules propositionnelles** est le plus petit ensemble de mots sur $\mathcal{A}$ tels que :

- $\mathcal{F}$ contient P,
- si $\mathcal{F}$ contient F alors $\mathcal{F}$ contient $(\neg F)$,
- si $\mathcal{F}$ contient F et G alors $\mathcal{F}$ contient $(F \vee G)$, $(F \wedge G)$, $(F \Rightarrow G)$ et $(F \Leftrightarrow G)$.

Plus simplement : si F et G sont des formules ($\in \mathcal{F}$) alors $(\neg F)$, $(F \vee G)$, $(F \wedge G)$, $(F \Rightarrow G)$, $(F \Leftrightarrow G)$ sont des formules.

Exemples (Mots qui sont des formules)

- $(A \wedge (B \vee C))$
- $(A \vee (\neg(B \vee (\neg A))))$
- $(A \Rightarrow (B \Rightarrow (A \Leftrightarrow (\neg B))))$

Exemples (Mots qui ne sont pas des formules)

- $(A \wedge (B \vee$
- $(A \vee \neg B \wedge C)$

On peut donner de l'ensemble $\mathcal{F}$ des formules propositionnelles une description plus explicite.
On définit par récurrence une suite $(\mathcal{F}_n)_{n\in\mathbb{N}}$ d'ensembles de mots sur $\mathcal{A}$:

- $\mathcal{F}_0 = P$
- $\mathcal{F}_{n+1} = \mathcal{F}_n \cup \{\neg F; F \in \mathcal{F}_n\} \cup \{(F\alpha G); F, G \in \mathcal{F}_n, \alpha \in \{\vee; \wedge; \Rightarrow; \Leftrightarrow\}\}$.

Théorème

$\mathcal{F} = \bigcup_{n\in\mathbb{N}} \mathcal{F}_n$.

Démonstration.

Il est clair que $\mathcal{F} \subset \bigcup_{n\in\mathbb{N}} \mathcal{F}_n$. En effet, $\bigcup_{n\in\mathbb{N}} \mathcal{F}_n$ contient P, et est stable par les opérations $\neg$, $\wedge$, etc . . .et $\mathcal{F}$ est le plus petit ensemble vérifiant cela.
Réciproquement, on montre par récurrence que pour tout $n \in \mathbb{N}$, $\mathcal{F}_n \subset \mathcal{F}$. □

IV : Les formules propositionnelles

2°) Suppression de parenthèses

Une formule contient par définition un assez grand nombre de parenthèses qui parfois semblent inutiles. Aussi pour ne pas alourdir l'écriture, on convient de se donner la possibilité de supprimer certaines parenthèses.

Il faut que les suppressions faites n'apportent pas d'ambiguïtés, c'est à dire que l'**on doit toujours pouvoir reconstituer la "vraie" écriture** (avec toutes les parenthèses).

Conventions de suppression des parenthèses

1. Suppression des parenthèses encadrant la formule entière :
   - $A \wedge B$ à la place de $(A \wedge B)$
2. Suppression des parenthèses séparant des connecteurs successifs identiques groupés par la droite :
   - $A \wedge B \wedge C$ à la place de $A \wedge (B \wedge C)$,
   - $A \Rightarrow B \Rightarrow C$ à la place de $A \Rightarrow (B \Rightarrow C)$.
3. Suppression des parenthèses séparant des connecteurs successifs identiques groupés par la gauche (sauf avec $\Rightarrow$) :
   - $A \wedge B \wedge C$ à la place de $(A \wedge B) \wedge C$,
   - Mais pas $A \Rightarrow B \Rightarrow C$ à la place de $(A \Rightarrow B) \Rightarrow C$! !

IV : Les formules propositionnelles

3°) Longueur et hauteur d'une formule

Définition

Soit $F \in \mathcal{F}$ une formule propositionnelle, la **longueur** *de F notée longueur(F) est le nombre de symboles différents des parenthèses apparaissant dans l'écriture de F.*

Exemples

- $longueur(A \wedge (B \vee C)) = 5$
- $longueur(A \vee (\neg(B \vee (\neg A)))) = 7$
- $longueur(A \Rightarrow (B \Rightarrow (A \Leftrightarrow (\neg B)))) = 8$

Définition

Soit $F \in \mathcal{F}$ une formule propositionnelle. La **hauteur** *de F, notée hauteur(F), est le plus petit entier n tel que $F \in \mathcal{F}_n$.*

Exemples

- $hauteur(A \wedge (B \vee C)) = 2$
- $hauteur(A \vee \neg(B \vee \neg A)) = 4$
- $hauteur(A \Rightarrow (B \Rightarrow (A \Leftrightarrow \neg B))) = 4$

IV : Les formules propositionnelles

4°) Démonstration par récurrence, démonstration par induction

On se pose le problème suivant : de quelle façon peut-on démontrer une propriété sur toutes les formules ?

On peut

- soit faire une démonstration par **récurrence** sur la hauteur des formules,
- soit faire une démonstration par **induction** en revenant à la définition initiale des formules et montrer que la propriété considérée est vraie pour les formules atomiques et qu'elle se préserve par les opérations de négation et par les connecteurs binaires.

Théorème

$h(F) \leq l(F)$.

Est-on sûr que le principe précédent donne une démonstration correcte ?

Lemme

Soit $\mathcal{P}$ une propriété sur les mots de $\mathcal{A}$ telle que :
- pour tout mot $M \in P$, on a que $\mathcal{P}(M)$ est satisfaite ;
- pour tout couple de mots de $\mathcal{A}$, M et N, si $\mathcal{P}(M)$ et $\mathcal{P}(N)$ sont satisfaites alors, $\mathcal{P}(\neg M)$ et $\mathcal{P}(M \alpha N)$ le sont aussi.
Alors pour toute formule propositionnelle $F \in \mathcal{F}$, $\mathcal{P}(F)$ est satisfaite.

Démonstration.

L'ensemble $E_{\mathcal{P}}$ est un ensemble de mots sur $\mathcal{A}$ qui contient P et est stable . . ., donc il contient $\mathcal{F}$.
On a aussi le même résultat pour les propriétés qui sont définies directement sur les formules plutôt que sur les mots. □

Lemme

Soit $\mathcal{P}$ une propriété sur les formules prop. de $\mathcal{F}$ telle que :
- pour tout mot $M \in P$, on a que $\mathcal{P}(M)$ est satisfaite ;
- pour tout couple de mots de $\mathcal{A}$, M et N, si $\mathcal{P}(M)$ et $\mathcal{P}(N)$ sont satisfaites alors, $\mathcal{P}(\neg M)$ et $\mathcal{P}(M \alpha N)$ le sont aussi.
Alors pour toute formule propositionnelle $F \in \mathcal{F}$, $\mathcal{P}(F)$ est satisfaite.

Démonstration.

On considère la propriété $\mathcal{Q}$ définie sur les mots de $\mathcal{A}$ par $\mathcal{Q}(M)$ est satisfaite si $M \in \mathcal{F}$ et $\mathcal{P}(M)$ et on applique le lemme précédent. □

De la même façon on peut faire des **définitions par induction** sur l'ensemble des formules.

Exemple

Définition du sous-arbre de décomposition F^* associé à une formule F :

- si F est une variable propositionnelle, F^* est l'arbre réduit à la racine F ;
- si $F = \neg G$ alors F^* est l'arbre de racine $\neg$ et de fils G^* ;
- si $F = G \alpha H$ (où α est un connecteur binaire), alors F^* est l'arbre de racine α avec comme fils gauche G^* et comme fils droit H^*.

Une sous-formule de F est une formule G dont l'arbre de décomposition est un sous-arbre de celui de F.

IV : Les formules propositionnelles

5°) Substitution

Définition

Sout F une formule propositionnelle, on note $F[A_1, \ldots, A_n]$ la formule F quand $A_1, \ldots, A_n$ sont des variables propositionnelles distinctes.

Exemples

- $F[A, B, C] = A \wedge (B \vee C)$
- $G[A] = A \vee \neg(B \vee \neg A)$

Définition

Si $G_1, \ldots, G_n$ sont des formules propositionnelles, on note $F[G_1/A_1, \ldots G_n/A_n]$ la formule obtenue en remplaçant chaque occurrence de A_i par G_i.

On peut définir cette opération par induction sur F, ce qui assure qu'on obtient bien ainsi une formule. Attention la substitution est simultanée ...

Exemples

- $F[(X \vee Y)/A] = (X \vee Y) \wedge (B \vee C)$ quand $F = A \wedge (B \vee C)$
- $G[(A \wedge B)/A] = (A \wedge B) \vee \neg(B \vee \neg(A \wedge B))$ quand $G = A \vee \neg(B \vee \neg A)$
- $G[(A \wedge A)/A] = (A \wedge A) \vee \neg(B \vee \neg(A \wedge A))$ quand $G = A \vee \neg(B \vee \neg A)$

V : Sémantique : Tables de vérité

1°) Valeur de vérité

Le calcul propositionnel s'intéresse aux propositions qui appartiennent au langage mathématique, ainsi parmi les propositions ci-dessous la 1, la 6 et la 7.

1. Tout carré est un rectangle.
2. Le livre "l'éthique protestante et l'esprit du capitalisme" a été écrit par E. Durkheim.
3. Marseille est une des grandes villes françaises.
4. La révolution française est la prise de pouvoir par la bourgeoisie.
5. J'ai 17 ans et j'ai le droit de vote.
6. $\sqrt{2}$ n'est pas un nombre rationnel.
7. Il existe deux nombres irrationnels x et y tels que x^y est rationnel.

De plus, cette partie de la logique mathématique ne s'intéresse pas au contenu des propositions mais seulement à la manière dont elles sont articulées (à l'aide des opérations de négation, conjonction, implication ...).

L'étude du calcul propositionnel est une première étape pour aborder l'étude complète des énoncés mathématiques. Cette première étape est celle qui s'intéresse à la manière dont les propositions se construisent/décomposent, à partir/en fonction de propositions de bases.

Pour donner une valeur de vérité à des énoncés mathématiques, il faut disposer de **valeurs de vérité**.

Nous nous contentons ici d'un ensemble de valeurs de vérité à deux valeurs **vrai** et **faux** $\Omega = \{0, 1\}$, on notera 0 pour faux et 1 pour vrai.

Définition

La **sémantique** *du calcul propositionnel est l'étude du calcul des valeurs de vérité des propositions à partir des propositions de bases qui les composent.*

V : Sémantique : Tables de vérité

2°) Connecteurs logiques et valeurs de vérité

Il y a deux fonctions de Ω dans Ω qui correspondent aux deux façons d'obtenir une proposition à partir d'une proposition de base :

- l'identité qui correspond au fait qu'une proposition de base est une proposition,
- la fonction qui envoie 1 sur 0 et 0 sur 1 qui correspond au fait que si on a une proposition, sa négation est encore une proposition dont la valeur de vérité est changée.

Il y a 16 fonctions de $\Omega \times \Omega$ dans Ω, nous n'en retenons que quatre :
– opération de conjonction,
– opération de disjonction,
– opération d'implication
– opération d'équivalence.

Ceci est suffisant dans la mesure où toutes les autres fonctions de Ω^2 dans Ω sont des combinaisons de ces quatre fonctions.

Nous retenons donc :
- 1 fonction de Ω dans Ω, correspondant à la négation, que l'on notera $\neg^*$ et définie par $\neg^*(0) = 1$ et $\neg^*(1) = 0$.
- 4 fonctions de $\Omega \times \Omega$ dans Ω, associées aux opérations de conjonction, disjonction, implication et équivalence :

- 1 fonction correspondant à la conjonction, notée $\wedge^*$:
$$\wedge^*(x, y) = 1 \text{ ssi } (x = 1 \text{ et } y = 1)$$
- 1 fonction correspondant à la conjonction, notée $\vee^*$:
$$\vee^*(x, y) = 1 \text{ ssi } (x = 1 \text{ ou } y = 1)$$
- 1 fonction correspondant à l'implication, notée $\Rightarrow^*$:
$$\Rightarrow^* (x, y) = 1 \text{ ssi } (x = 0 \text{ ou } (x = 1 \text{ et } y = 1))$$
- 1 fonction correspondant à l'équivalence, notée $\Leftrightarrow^*$:
$$\Leftrightarrow^* (x, y) = 1 \text{ ssi } x = y$$

Il est important de ne pas confondre "vérité d'une proposition implicative" et "vérité de sa conclusion". *Si x est un multiple de 4 alors x est pair* peut s'appliquer à 24 mais aussi à 17.

Dans le langage courant on exprime de plusieur façons une proposition implicative :

- Si on a P alors on a Q
- Si P, Q
- P implique Q
- P entraîne Q
- $P \Rightarrow Q$
- Q est une condition nécessaire pour avoir P
- Pour avoir P il est nécessaire que Q
- P est une condition suffisante pour Q
- Pour avoir Q il suffit d'avoir P.

V : Sémantique : Tables de vérité

3°) Distribution de valeurs de vérité et valuation

Pour calculer la valeur de vérité de la formule $(A \vee B) \Rightarrow C$ nous avons besoin de connaître la valeur de vérité des atomes apparaissant dans cette formule. C'est pour cela que nous introduisons la notion de distribution de valeurs de vérité ou valuation.

Définition

Une distribution de valeurs de vérité ou valuation est une fonction δ de P l'ensemble des variables propositionnelles dans $\Omega = \{0, 1\}$.

Exemples.

Lemme

Une valuation δ se prolonge de manière unique en une fonction $\overline{\delta}$ de $\mathcal{F}$ dans Ω satisfaisant :

i $\overline{\delta}(A) = \delta(A)$ *lorsque* $A \in P$;

ii $\overline{\delta}(\neg F) = \begin{cases} 0 & \text{si } \overline{\delta}(F) = 1, \\ 1 & \text{si } \overline{\delta}(F) = 0. \end{cases}$

iii $\overline{\delta}(F \wedge G) = 1$ *si* $\overline{\delta}(F) = 1$ *et* $\overline{\delta}(G) = 1$.

iv $\overline{\delta}(F \vee G) = 1$ *si* $\overline{\delta}(F) = 1$ *ou* $\overline{\delta}(G) = 1$.

v $\overline{\delta}(F \Rightarrow G) = 1$ *si* $\overline{\delta}(F) = 0$ *ou* $(\overline{\delta}(F) = 1$ *et* $\overline{\delta}(G) = 1)$.

vi $\overline{\delta}(F \Leftrightarrow G) = 1$ *si* $\overline{\delta}(F) = \overline{\delta}(G)$.

La démonstration se fait par induction sur F.
$\overline{\delta}(F)$ est appelée la **valeur de vérité** de F pour δ.

Soit F une formule, la définition de δ sur $\mathcal{F}$ nous donne une méthode pour calculer $\overline{\delta}(F)$:

– on commence par calculer les valeurs prises par δ sur les sous-formules de F de hauteur 1,

– on applique autant de fois que nécessaire les fonctions asssociées aux connecteurs.

Exemples.

Lemme

La valeur de vérité d'une formule ne dépend que de la distribution des valeurs de vérité, c'est à dire des atomes figurant dans cette formule.

Ce lemme permet de représenter par un tableau la valeur de vérité d'une formule F en fonction de toutes les distributions de vérité possibles : **la table de vérité d'une formule**.
Exemple : les tables de vérité des connecteurs ; etc...

L'application $\overline{\delta}$ est définie par le tableau suivant :

F	G	$\neg F$	$F \wedge G$	$F \vee G$	$F \Rightarrow G$
0	0	1	0	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	1	0	1	1	1

Si F a n variables, alors la table de vérité de F a $2^n + 1$ lignes.
Exemple sur $(A \Rightarrow (B \vee B))$

Par abus, on notera δ au lieu de $\overline{\delta}$.

V : Sémantique : Tables de vérité

4°) Tautologie, formules logiquement équivalentes

Définition

Soit F une formule, δ une valuation :

- *δ satisfait F si $\delta(F) = 1$.*
- *F est* **satisfaisable** *si il existe une valuation δ qui satisfait F (i.e. telle que $\delta(F) = 1$).*
- *F est une* **tautologie** *si elle est satisfaite par toutes les valuations. On dit aussi universellement valide.*
- *F est une antilogie ou une contradiction si elle n'est satisfaite par aucune valuation.*

Exemples.

Définition

Soit δ une valuation et Γ un ensemble de formules :

- *δ satisfait Γ si δ satisfait toutes les formules de Γ.*
- *Γ est satisfaisable si il existe une valuation qui satisfait Γ.*
- *Γ est insatisfaisable ou inconsistante ou contradictoire s'il n'existe pas de valuation qui satisfasse simultanément toutes les formules de Γ.*

Exemples.

Définition

Soient F et G deux formules sur P.
F et G sont **logiquement équivalentes** *si pour toute valuation δ, $\delta(F) = \delta(G)$.*
On notera alors $F \equiv G$.

Lemme

Soit F, G et H des formules propositionnelles, on a :

1. $(F \vee G) \vee H \equiv F \vee (G \vee H)$ *(associativité du $\vee$)* ;
2. $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$ *(associativité du $\wedge$)* ;
3. $F \vee G \equiv G \vee H$ *(commutativité du $\vee$)* ;
4. $F \wedge G \equiv G \wedge F$ *(commutativité du $\wedge$)* ;
5. $\neg(F \wedge G) \equiv \neg F \vee \neg G$ *(négation d'une conjonction)* ;
6. $\neg(F \vee G) \equiv \neg F \wedge \neg G$ *(négation d'une disjonction)* ;
7. $\neg\neg F \equiv F$ *(double négation)* ;
8. $F \wedge F \equiv F$ *(idempotence du $\wedge$)* ;
9. $F \vee F \equiv F$ *(idempotence du $\vee$)* ;
10. $F \Rightarrow G \equiv \neg G \Rightarrow \neg F$ *(contraposée).*

Attention, ce qu'on appelle la réciproque de $F \Rightarrow G$ c'est la formule $G \Rightarrow F$.

Lemme

Les formules suivantes sont des tautologies.

1. $F \Rightarrow G \Rightarrow F$
2. $F \Rightarrow G \vee F$
3. $F \Rightarrow F \vee G$
4. $F \wedge G \Rightarrow F$
5. $F \wedge G \Rightarrow G$
6. $\neg F \Rightarrow F \Rightarrow G$
7. $F \vee \neg F$

Démonstration

Il suffit de construire les tables de vérité des formules.

V : Sémantique : Tables de vérité

5°) Déduction sémantique

Définition

Soit Γ *un ensemble de formules et* F *une formule.*
On dit que Γ *valide* F *si toute valuation qui satisfait* Γ *satisfait* F*.*

On note $\Gamma \models F$.

Remarque

Si F est une tautologie on a $\emptyset \models F$, ce que l'on notera $\models F$.

Théorème

Quels que soient les ensembles de formules Γ *et* Δ*, la formule* F*, les propriétés suivantes sont vérifiées :*

- *Si* $\Gamma \subset \Delta$ *et* Δ *est satisfaisable alors* Γ *est satisfaisable.*
- *Si* $\Gamma \subset \Delta$ *et* Γ *est insatisfaisable alors* Δ *est insatisfaisable.*
- *Si* Δ *est satisfaisable alors* Δ *est finiment satisfaisable.*
- F *est une tautologie ssi* F *est conséquence de l'ensemble vide.*

Théorème

Quels que soient les ensembles de formules Γ *et* Δ*, l'entier non nul* n *et les formules* G, H, $F_1, \ldots, F_n$*, les propriétés suivantes sont vérifiées :*

- **Règle du Modus Ponens** *Si* $\Gamma \models F \Rightarrow G$ *et* $\Gamma \models F$ *alors* $\Gamma \models G$.
- $\Delta \cup \{G\} \models H$ *ssi* $\Delta \models G \Rightarrow H$.
- $\Delta \models G \wedge H$ *ssi* $\Delta \models G$ *et* $\Delta \models H$.
- **Règle de l'absurde** $\Delta \models G$ *ssi* $\Delta \cup \{\neg G\}$ *est insatisfaisable.*
- $\{F_1, \ldots, F_n\} \models G$ *ssi* $\models (F_1 \wedge \cdots \wedge F_n) \Rightarrow G$.
- $\{F_1, \ldots, F_n\}$ *est contradictoire ssi* $\neg F_1 \vee \cdots \vee \neg F_n$ *est une antilogie.*
- *L'ensemble vide est satisfaisable.*
- *L'ensemble de toutes les formules est contradictoire.*

V : Sémantique : Tables de vérité

6°) Substitution et équivalence

Si on remplace dans une formule une sous-formule par une formule équivalente, on obtient une formule équivalente.

Proposition

Soient F, F', G, G' des formules. Soit p une variable propositionnelle.

- *Si F est une tautologie, alors $F[G/p]$ aussi.*
- *Si $F \equiv F'$, alors $F[G/p] \equiv F'[G/p]$.*
- *Si $G \equiv G'$, alors $F[G/p] \equiv F[G'/p]$.*

Preuve du résultat par induction structurelle.

V : Sémantique : Tables de vérité

7°) Système complet de connecteurs

Définition

Un **système complet de connecteurs** *S est un ensemble de connecteurs tel que toute formule du calcul propositionnel est équivalente à une formule écrite uniquement avec des variables propositionnelles et les connecteurs de S.*

Proposition

– L'ensemble $\{\neg, \wedge\}$ *est un système complet de connecteurs pour la logique propositionnelle.*
– L'ensemble $\{\neg, \vee\}$ *est un système complet de connecteurs pour la logique propositionnelle.*

Preuve du résultat par induction structurelle.

V : Sémantique : Tables de vérité

8°) Formes normales, complétude fonctionnelle

On a associé à chaque formule ayant n atomes une fonction de Ω^n dans Ω. La question se pose de savoir si, réciproquement, on peut associer une formule propositionnelle à n atomes à toute fonction de Ω^n dans Ω.
La réponse est oui, on peut trouver une formule ayant une table de vérité donnée, mais la formule trouvée n'est pas unique. On sait en construire une d'une "forme particulière" dite forme normale.

Théorème (Complétude fonctionnelle)

Supposons P un ensemble fini de variables propositionnelles. Soit V l'ensemble des valuations sur P. Toute fonction f de V dans $\{0,1\}$ est la valeur de vérité d'une formule F sur P.

Preuve par récurrence sur le nombre d'atomes.

Définition

1. *On appelle littéraux les formules utilisant au plus 2 symboles (donc des variables propositionnelles ou des négations de variable propositionnelle).*
2. *On appelle* **bloc conjonctif (BC)** *les formules n'utilisant que des littéraux et des connecteurs* $\wedge$.
3. *On appelle* **bloc disjonctif (BD)** *les formules n'utilisant que des littéraux et des connecteurs* $\vee$. *(on parle aussi de clauses)*
4. *On appelle* **forme normale conjonctive (FNC)** *une conjonction de BD (de clauses). (on parle aussi de formes clausales).*
5. *On appelle* **forme normale disjonctive** *une disjonction de BC.*

Exercice : donner des exemples de formules qui soient des *BC*, des *BD*. des *FNC*. des *FND*.

Théorème (Complétude fonctionnelle)

Chaque fonction booléenne sur $\{0,1\}^n$ *peut s'exprimer par une forme normale disjonctive.*

Démonstration

On esquisse l'algorithme permettant de construire une formule (en FND) admettant une table donnée comme table de vérité. (Rem. On représente une fonction de Ω^n dans Ω par une table avec le choix de n variables propositionnelles $A_1 \ldots, A_n$ ou A, B, C ....

Exemple :

A	B	C	$f(A, B, C)$
0	0	0	1
0	0	1	1
0	1	0	0
1	0	0	0
0	1	1	0
1	0	1	0
1	1	0	0
1	1	1	1

Etape 1 : On repère le nombre de lignes sur lesquelles il y a un 1 dans la dernière colonne (comme valeur de la formule à trouver) pour former une forme normale disjonctive avec k blocs conjonctifs.
Dans l'exemple, il y a trois lignes concernées.

Etape 2 : Formation des blocs conjonctifs.
Pour chacune des lignes repérées, pour chaque A_i :

- on écrit A_i s'il y a un 1 dans la colonne A_i de la ligne considérée,
- on écrit $\neg A_i$ sinon.

Et on prend le $\wedge$ des ces littéraux.

Dans l'exemple cela donne :

- $\neg A \wedge \neg B \wedge \neg C$ pour la première ligne ;
- $\neg A \wedge \neg B \wedge C$ pour la deuxière ligne ;
- $A \wedge B \wedge C$ pour la troisière ligne.

Etape 3 : On réunit par des $\vee$ ces blocs disjonctifs afin d'obtenir une formule en FND.

Dans l'exemple, on obtient

$$(\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C)$$

Pour terminer la preuve, il suffit de s'assurer que la table de vérité de la formule trouvée est bien celle de laquelle nous sommes partis.

Corollaire

Toute formule propositionnelle est logiquement équivalente à une FND et à une FNC.

V : Sémantique : Tables de vérité

9°) Propriétés de compacité

Soit un ensemble Σ de formules construites sur un ensemble dénombrable P de variables propositionnelles.

Théorème (Théorème de compacité (1ère version))

Σ est consistant si et seulement si toute partie finie de Σ est consistant.

Théorème (Théorème de compacité (2ème version))

Σ est inconsistant si et seulement si Σ possède une partie finie inconsistante.

Théorème (Théorème de compacité (3ème version))

Soit F une formule construite sur P, F est conséquence de Σ si et seulement si F est conséquence d'une partie finie de Σ.

Propriétés de compacité : preuve (1)

Un des sens est trivial, autre sens :

- On définit par récurrence sur n une valuation v telle que tout sous-ensemble fini de Σ admet une valuation qui la satisfait dans laquelle $p_1, \ldots, p_n$ prennent les valeurs $v(p_1), \ldots, v(p_n)$ (cf slide suivant).
- Il en résulte que v satisfait Σ : en effet, soit F une formule de Σ. F ne dépend que d'un ensemble fini $p_{i_1}, p_{i_2}, \ldots, p_{i_k}$ de variables propositionnelles (celles qui apparaissent dans F). En considérant $n = max(i_1, i_2, \ldots, i_k)$, chacune de ces variables p_{i_j} est parmi $\{p_1, \ldots, p_n\}$.
- Nous savons alors que le sous ensemble fini $\{F\}$ réduit à la formule F est satisfait par une valuation dans laquelle $p_1, \ldots, p_n$ prennent les valeurs $v(p_1), \ldots, v(p_n)$, i.e. F est satisfaite par v.

Propriétés de compacité : preuve (2)

Lemme

Supposons qu'il existe une application v de $\{p_1, \ldots, p_n\}$ dans $\{0, 1\}$ telle que tout sous-ensemble fini de Σ admette une valuation qui la satisfait dans laquelle p_1, ..., p_n prennent les valeurs $v(p_1)$, ..., $v(p_n)$. Alors on peut étendre v à p_1, ..., p_n, p_{n+1} avec la même propriété.

Si $v(p_{n+1}) = 0$ ne convient pas, alors il existe un ensemble fini U_0 de Σ qui ne peut pas être satisfait quand p_1, ..., p_n, p_{n+1} prennent les valeurs respectives $v(p_1)$, ..., $v(p_n)$ et 0. Si U est un sous-ensemble fini quelconque de Σ, alors d'après l'hypothèse faite sur v, $U_0 \cup U$ admet une valuation qui le satisfait dans laquelle p_1, ..., p_n prennent les valeurs $v(p_1)$, ..., $v(p_n)$. Avec cette valuation, la proposition p_{n+1} prend donc la valeur 1. Autrement dit, tout sous-ensemble fini U de Σ admet une valuation qui le satisfait dans laquelle p_1, ..., p_n, p_{n+1} prennent les valeurs respectives $v(p_1)$, ..., $v(p_n)$ et 1.

VI : Démonstrations, preuves formelles

1°) Introduction

Qu'est-ce qu'une théorème ? une tautologie !
Qu'est-ce qu'une démonstration ? La preuve d'une théorème.
Qu'est-ce qu'une preuve ?

La méthode par table de vérité permet de montrer qu'une formule est une tautologie.

Cette méthode est finie dans le cas de la logique propositionnelle, ce n'est plus le cas pour d'autres logiques.

Cependant, cette méthode est particulièrement inefficace : 2^n pour n variables !

On cherche donc des méthodes plus efficaces (syntaxiques).

Il existe plusieurs systèmes de déduction valides et complets. Entre autres,

- les systèmes de Hilbert,
- la déduction naturelle,
- le calcul des séquents de Gentzen,
- la méthode par résolution,
- la méthode des tableaux.

Formellement, on écrit $\Delta \vdash F$ pour dire que la formule F se prouve à partir de l'ensemble de formules propositionnelles Δ. On note $\vdash F$ si $\emptyset \vdash F$.

En général, on s'attend à ce qu'une méthode de preuve soit toujours valide : elle ne produit que des déductions **correctes**.

Dans tous les cas se pose la question de la **complétude** : est-ce que tous les théorèmes (tautologies) sont prouvables ?

Définition

Un système de déduction est correct si $\vdash F$ *alors* $\models F$.
Un système de déduction est complet si $\models F$ *alors* $\vdash F$

VI : Démonstrations, preuves formelles
2°) Calcul des séquents

Les systèmes à la Hilbert (la règle modus ponens et beaucoup d'axiomes) présentent deux défauts majeurs :

- La recherche de preuves peut s'avérer extrêmement complexe, puisque rien ne permet de prédire ce que pourrait être les prémisses d'un modus ponens.
- Les axiomes ne permettent pas de comprendre la structure algébrique des connecteurs, donc masquent des propriétés importantes.

Les **calculs des séquents** de Gentzen (1930) pallient ces deux problèmes.

Définition

Un **séquent** *est une paire de 2 listes de formules* $(n, m \geq 0)$ *:*

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_m$$

*Le séquent se lit "*si (A_1 et …et A_n) alors (B_1 ou …ou B_m)*".*
La formule associée au séquent est

$$A_1 \wedge \cdots \wedge A_n \Rightarrow B_1 \vee \cdots \vee B_m$$

On note $\top$ quand $n = 0$ et $\bot$ quand $m = 0$.

Un calcul des séquents à la Gentzen est la donnée de

- un ensemble d'axiomes : un axiome est dans ce cas un séquent,
- un ensemble de règles : une règle est donnée par un ensemble de séquents prémisses et d'un séquent conclusion.

On a en calcul des séquents à la Gentzen peu d'axiomes et beaucoup de règles. Par exemple,

- un seul axiome (d'identité) : $A \vdash A$,
- des règles indiquant comment introduire un connecteur.

Définition

Une formule F est une **instance** *d'une formule G si F s'obtient en substituant certaines variables propositionnelles de G par des formules.*
Une **instance** *de règle est une règle sur laquelle on a appliqué la même substitution sur toutes les prémisses et la conclusion.*

La formule $((C \Rightarrow D) \Rightarrow (\neg A \Rightarrow (C \Rightarrow D)))$ est une instance de $(A \Rightarrow (B \Rightarrow A))$, en prenant $(C \Rightarrow D)$ pour A, et $\neg A$ pour B.

Définition

Une **preuve** Π *du séquent* $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *est une suite de séquents* $\mathcal{S}_1, \ldots, \mathcal{S}_p = A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *telle que pour tout* $1 \leq i \leq p$ *:*

- *soit* $\mathcal{S}_i$ *est une instance d'axiome,*
- *soit* $\mathcal{S}_i$ *est la conclusion d'une instance de règle dont les prémisses sont des séquents* $\mathcal{S}_j$ *avec* $j < i$*.*

Une formule F est **prouvable** quand il existe une preuve du séquent $\vdash F$.

Un calcul des séquents pour la logique propositionnelle, noté **LK**, est le suivant :

Groupe identité

$$\frac{}{A \vdash A}\ (\texttt{axiome}) \qquad \frac{\Gamma \vdash \Delta, A \quad A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ (\texttt{coupure})$$

Groupe structurel

$$\frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}\ (\texttt{d-échange}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ (\texttt{d-aff.}) \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ (\texttt{d-cont.})$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}\ (\texttt{g-échange}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ (\texttt{g-aff.}) \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ (\texttt{g-cont.})$$

Groupe logique

$$\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ (\texttt{d-négation}) \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\ (\texttt{g-négation})$$

$$\frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}\ (\texttt{d-implication}) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ (\texttt{g-implication})$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\ (\texttt{d-et}) \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\ (\texttt{d-ou})$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ (\texttt{g-ou}) \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ (\texttt{g-et})$$

Preuve du séquent : $A \Rightarrow B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash A \Rightarrow D$

Sans coupure :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,ax \quad \dfrac{\dfrac{\dfrac{}{B \vdash B}\,ax \quad \dfrac{}{C \vdash C}\,ax}{B, C \vdash B \wedge C}\,d-\wedge \quad \dfrac{}{D \vdash D}\,ax}{B, C, B \wedge C \Rightarrow D \vdash D}\,g-\Rightarrow}{A, B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash D}\,g-\Rightarrow}{A, A, A \Rightarrow B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash D}\,g-\Rightarrow}{A, A \Rightarrow B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash D}\,g-cont}{A \Rightarrow B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash A \Rightarrow D}\,d-\Rightarrow$$

Avec coupure :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{B \vdash B}}{A \Rightarrow B, A \vdash B} \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{C \vdash C}}{A \Rightarrow C, A \vdash C}}{A \Rightarrow B, A, A \Rightarrow C \vdash B \wedge C}}{A, A \Rightarrow B, A \Rightarrow C \vdash B \wedge C} \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{}{B \wedge C \vdash B \wedge C} \quad \dfrac{}{D \vdash D}}{B \wedge C, B \wedge C \Rightarrow D \vdash D}}{A, A \Rightarrow B \wedge C, B \wedge C \Rightarrow D \vdash D}}{A \Rightarrow B, A \Rightarrow C \vdash A \Rightarrow B \wedge C \quad A \Rightarrow B \wedge C, B \wedge C \Rightarrow D \vdash A \Rightarrow D}\,(cut)$$
$$A \Rightarrow B, A \Rightarrow C, B \wedge C \Rightarrow D \vdash A \Rightarrow D$$

Théorème de correction

Le théorème de correction se montre par récurrence sur la hauteur des preuves et en montrant que chaque règle de LK est correcte.

Définition

La hauteur $h(\Pi)$ d'une preuve Π est définie par induction :

- $h(\Pi) = 1$ si Π est un axiome.
- $h(\Pi) = h(\Pi_1) + 1$ si Π est obtenue par application d'une règle unaire à partir de la preuve Π_1 (p.e. la négation).
- $h(\Pi) = max(h(\Pi_1), h(\Pi_2)) + 1$ si Π est obtenue par application d'une règle binaire à partir des preuves Π_1 et Π_2 (p.e. le $\wedge$ droit).

Lemme

Pour toute règle de LK, si ses séquents prémisses sont des tautologies alors sa conclusion est une tautologie.

Cas de la règle $\wedge - d$: $\dfrac{\Gamma_1 \vdash \Delta_1, A \quad \Gamma_2 \vdash \Delta_2, B}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}$

Soit G_i (resp. D_i) la formule avec $\wedge$ associée à Γ_i (resp. Δ_i avec $\vee$).

Par hypothèse, les formules $G_1 \Rightarrow D_1 \vee A$ et $G_2 \Rightarrow D_2 \vee B$ sont des tautologies.

On veut montrer que $F = G_1 \wedge G_2 \Rightarrow D_1 \vee D_2 \vee (A \wedge B)$ est une tautologie.

Soit δ une dvv,
– si $\overline{\delta}(G_1 \wedge G_2) = 0$ alors $\overline{\delta}(F) = 1$,
– si $\overline{\delta}(G_1 \wedge G_2) = 1$ alors

- $\overline{\delta}(G_1) = 1$, or $\overline{\delta}(G_1 \Rightarrow D_1 \vee A) = 1$ donc $\overline{\delta}(D_1 \vee A) = 1$, donc $\overline{\delta}(D_1) = 1$ ou $\overline{\delta}(A) = 1$.
- $\overline{\delta}(G_2) = 1$, or $\overline{\delta}(G_2 \Rightarrow D_2 \vee B) = 1$ donc $\overline{\delta}(D_2 \vee B) = 1$, donc $\overline{\delta}(D_2) = 1$ ou $\overline{\delta}(B) = 1$.
- DONC $\overline{\delta}(D_1) = 1$ ou $\overline{\delta}(D_2) = 1$ ou ($\overline{\delta}(A) = 1$ et $\overline{\delta}(B) = 1$).
- DONC $\overline{\delta}(D_1 \vee D_2 \vee (A \wedge B)) = 1$.

donc $\overline{\delta}(F) = 1$.
Puisque pour toute dvv, la valuation de F est 1 alors F est une tautologie.

Théorème

Si le séquent $\Gamma \vdash \Delta$ est prouvable dans LK alors la formule associée est une tautologie.
En particulier, si $\vdash F$ est prouvable dans LK alors F est une tautologie.

Démonstration par récurrence sur la hauteur d'une preuve et en appliquant le lemme précédent.

Théorème de complétude

Théorème

Si F est une tautologie alors le séquent $\vdash F$ est prouvable dans LK.

Soient $X_1, \ldots, X_n$ les var. prop. apparaissant dans F. Soit δ une dvv, on note $\phi(F, \delta)$ la formule $\epsilon_1(X_1) \wedge \ldots \epsilon_n(X_n)$ où $\epsilon_i(X_i) = \neg X_i$ si $\delta(X_i) = 0$, $\epsilon_i(X_i) = X_i$ si $\delta(X_i) = 1$.

1. Pour toute formule G et H, pour tout connecteur $c \in \{\wedge, \vee, \Rightarrow\}$, on a :
   - $\vdash \phi(GcH, \delta) \Rightarrow \phi(G, \delta) \wedge \phi(H, \delta)$
   - $\vdash \phi(GcH, \delta) \Leftarrow \phi(G, \delta) \wedge \phi(H, \delta)$
   - $\vdash \phi(\neg G, \delta) \Rightarrow \phi(G, \delta)$
   - $\vdash \phi(\neg G, \delta) \Leftarrow \phi(G, \delta)$

   En clair : ϕ ne dépend que de δ et des var. prop., pas de la structure de la formule.
2. Pour toute formule F, la formule $\vdash \bigvee\{\phi(F, \delta)\ ;\ \delta \text{ est une dvv }\}$ est prouvable.
3. Par récurrence sur la taille de F, pour toute dvv δ,
   - si $\overline{\delta}(F) = 1$ alors $\phi(F, \delta) \vdash F$ est prouvable
   - si $\overline{\delta}(F) = 0$ alors $\phi(F, \delta) \vdash \neg F$ est prouvable
4. Si F est une tautologie alors $\bigvee\{\phi(F, \delta)\ ;\ \delta \text{ est une dvv }\} \vdash F$ est prouvable.
5. Modus Ponens entre 2 et 4

Règles dérivables

Définition

Une règle $\dfrac{\Gamma_1 \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2}{\Gamma \vdash \Delta}$ est dite **dérivable** dans LK s'il existe une preuve Π de LK auquel on adjoint $\Gamma_1 \vdash \Delta_1$ et $\Gamma_2 \vdash \Delta_2$ comme axiomes utilisables une fois et Π a comme conclusion $\Gamma \vdash \Delta$.

Puisque LK est correct, une règle dérivable est aussi correcte dans le sens où l'ajouter à LK ne change pas ce qui est prouvable.

Les règles suivantes sont dérivables dans LK :

$$\frac{\Gamma_1 \vdash \Delta, A \quad \Gamma_2 \vdash \Delta, B}{\Gamma_1, \Gamma_2 \vdash \Delta, A \wedge B} \qquad \frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \qquad \frac{B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}$$

$$\frac{\Gamma_1, A \vdash \Delta \quad \Gamma_2, B \vdash \Delta}{\Gamma_1, A \vee B, \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B} \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}$$

Démonstration par récurrence en ajoutant des contractions.

Théorème d'élimination de la coupure (Hauptsatz, Gentzen 1934)

Théorème

Tout séquent prouvable dans LK peut être prouvé sans utiliser la règle de coupure.

C'est l'un des théorèmes les plus importants du système LK : celui correspondant à l'idée qu'une démonstration d'un théorème peut se faire sans utiliser de lemmes.
Il a aussi une conséquence importante :

Théorème (Propriété de la sous-formule)

Tout séquent prouvable dans LK peut l'être en ne faisant appel qu'à des sous-formules des formules apparaissant dans le séquent prouvable.

Autre conséquence importante : cela permet de montrer que le système LK est non-contradictoire (on ne peut pas montrer $\vdash A$ et $\vdash \neg A$) :

1. si on pouvait montrer $\vdash A$ et $\vdash \neg A$, alors à partir de $\vdash A$ on pourrait montrer $\neg A \vdash$,
2. donc par coupure entre $\neg A \vdash$ et $\vdash \neg A$ on montrerait le séquent vide $\vdash$.
3. Or le séquent vide n'admet pas de sous-formule, donc n'est pas prouvable.
4. donc l'hypothèse est fausse : le système LK n'est pas contradictoire.

Preuve du théorème d'élimination des coupures de LK :

1. On montre que LK (+coupure) est équivalent à
   $$\frac{\Gamma_1 \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2^A \vdash \Delta_1^A, \Delta_2}$$
   LK-coupure+mix où la règle mix est : avec Γ_2^A et Δ_1^A obtenues en supprimant 0 ou plus occurrences de A dans Γ_2 et Δ_1.
2. La preuve est ensuite faite par récurrence sur le rang du mix, i.e. le couple (d, h) ordonné lexicographiquement avec d la taille de la formule A du mix, et h la hauteur de la preuve, en considérant les cas suivants :
   - axiome
   - commutation de règles (aff. et contr. par un mix !)
   - réductions essentielles, i.e. $\wedge/\wedge$, ...

Propriétés fondamentales

- La logique intuitionniste (i.e. au plus une conclusion) est constructive.
- Forte normalisation de la déduction naturelle pour la logique intuitionniste (Prawitz 1965) : l'ordre d'élimination des coupures n'a pas d'importance, le nombre d'étapes est fini.
- Correspondance de Curry-Howard : l'évaluation en lambda-calcul typé est la normalisation.

Logique intuitionniste

Démonstrations *constructives* : un objet mathématique qui existe est un objet mathématique que l'on sait construire.

Conséquences :

- Pas de tiers-exclu : $A \vee \neg A$ est refusé.
- Il n'y a vérité que quand il y a preuve.
- Le fait qu'une preuve prouve une proposition doit être *décidable* : il doit exister un "programme" qui construit cette preuve.
- Une proposition de la forme *si P alors Q sinon R* n'est possible que si on sait écrire un programme pour P qui renvoie vrai ou faux, et un programme qui prouve Q et un programme qui prouve R.

Le fait de vouloir 'limiter' la logique classique vient aussi de ce que certaines tautologies ne sont guère convaincantes :

Exemple

$((A \wedge B) \Rightarrow C) \;\Rightarrow\; ((A \Rightarrow C) \vee (B \Rightarrow C))$ est une tautologie.

(la démonstration utilise $\neg\neg A \Leftrightarrow A$ non démontrable en logique intuitionniste)

Prendre

pour A : *"x=0"*, pour B : *"y=0"*, pour C : *"x=0 et y=0"*.

Notez qu'alors

(i) $((A \wedge B) \Rightarrow C)$ est (intuitivement/mathématiquement/logiquement) correct.

(ii) donc $((A \Rightarrow C) \vee (B \Rightarrow C))$ est correct (modus ponens)

(iii) donc la proposition suivante est correcte !

si (x=0) alors (x=0 et y=0)

ou

si (y=0) alors (x=0 et y=0)

Sémantique BHK (Brouwer - Heyting - Kolmogorov)

Une *preuve* doit être un *processus effectivement calculable* :
— un algorithme,
— une machine,
— un programme, . . .

- La *thèse de Church* affirme que les modes d'expression de calculs, p.e. des langages de programmation(suffisamment riches), sont tous équivalents : on peut écrire en C / PHP / Java / Caml ... le même ensemble de fonctions.
- Ces fonctions sont dites *calculables*.
- Une preuve est une manière de calculer une *fonction calculable*.

Sémantique BHK :

- la sémantique d'une formule est l' ensemble de ses preuves
- la sémantique d'une formule est l'ensemble de ses programmes

- Preuve d'une formule atomique == méthode automatisable permettant de vérifier que la formule atomique est vraie.
- Preuve de $A \wedge B$ == preuve de A et preuve de B.
- Preuve de $A \vee B$ == preuve de A
  ou bien
  Preuve de $A \vee B$ == preuve de B.
- Preuve de $\neg A$ == processus effectivement calculable qui transforme toute preuve de A en une contradiction.
- Preuve de $A \Rightarrow B$ == processus effectivement calculable qui transforme toute preuve de A en une preuve de B.

LJ : calcul des séquents intuitionniste

- Mêmes formules qu'en logique classique.
- Même forme des séquents : $\Gamma \vdash \Delta$
- MAIS : Δ ne contient au plus qu'UNE formule !

Calcul des séquents pour la logique intuitionniste **LJ** :

Groupe identité

$$\frac{}{A \vdash A}\ (\texttt{axiome}) \qquad \frac{\Gamma \vdash A \quad A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ (\texttt{coupure})$$

Groupe structurel

$$\frac{\Gamma \vdash}{\Gamma \vdash A}\ (\texttt{d-aff.})$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}\ (\texttt{g-échange}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ (\texttt{g-aff.}) \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ (\texttt{g-cont.})$$

Groupe logique

$$\frac{\Gamma, A \vdash}{\Gamma \vdash \neg A}\ (\texttt{d-négation}) \qquad \frac{\Gamma \vdash A}{\Gamma, \neg A \vdash}\ (\texttt{g-négation})$$

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\texttt{d-implication}) \quad \frac{\Gamma \vdash A \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ (\texttt{g-implication})$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\texttt{d-et}) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (\texttt{d-ou}_1) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (\texttt{d-ou}_2)$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ (\texttt{g-ou}) \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ (\texttt{g-et})$$

Propriétés

- LJ satisfait l'élimination des coupures.

- (isomorphisme de Curry-Howard) : il y a correspondance bijective entre :
  – formule de LJ et type d'un langage de programmation fonctionnelle
  – preuve d'une formule de LJ et programme dans ce langage
  – réduction de la coupure et exécution du programme

Langage fonctionnel

Un **langage fonctionnel** est un langage de programmation implantant le paradigme de calcul appelé λ-calcul.

Parmi les langages fonctionnels, on peut citer : OCaml, Haskell, ...
D'autres langages utilisent des principes fonctionnels : JavaScript, Java, Scala, ...

Avantages :

- La plupart des langages fonctionnels sont **fortement typés** : il ne peut pas y avoir d'erreurs à l'exécution.
- Il n'y a pas de différence entre donnée et fonction : il est facile de définir des fonctions très générales.

λ-calcul

Un terme (ou programme) est
– une variable x
– ou une fonction $\lambda x.t$ où x est une variable, t un terme
– ou une application $t(u)$ où t et u sont des termes

Exemples :

- x
- $x(y)$
- $x(y(z))$
- $\lambda x.x$
- $\lambda x.\lambda y.x(y)$
- $(\lambda x.z)(y)$
- $(\lambda x.\lambda y.x)(\lambda z.z)$
- $\lambda x.x(\lambda y.x(y))$
- $\lambda x.x(x)$

Par simplicité, on notera

- tu au lieu de $t(u)$
- $\lambda xy.t$ au lieu de $\lambda x.\lambda y.t$

Les variables sont notées x, y, z, ... Les termes qui ne sont pas nécessairement des variables sont notés t, u, ...

λ-calcul

Une exécution est une suite d'applications de la règle :

$$(\lambda x.t)(u) \text{ se réécrit en } t[u/x]$$

Cette réduction s'appelle une β-réduction et est notée :

$$(\lambda x.t)(u) \longrightarrow_\beta t[u/x]$$

Dans $t[u/x]$, toutes les occurrences *libres* de la variable x sont remplacées par le terme u.

λ-calcul : variables libres/liées

Une occurrence de variable x est **libre** si elle n'est pas dans un terme *sous* un λx.

Exemples (les occurrences libres de x sont en bleu, celles de y en rouge, celles de z en vert) :

- x
- $x(y)$
- $x(y(z))$
- $\lambda x.x$
- $\lambda x.\lambda y.x(y)$
- $(\lambda x.z)(y)$
- $(\lambda x.\lambda y.x)(\lambda z.z)$
- $\lambda x.x(\lambda y.x(y))$
- $\lambda x.x(x)$

Une variable qui n'est pas libre est dite **liée**.

λ-calcul : β-réduction

Exemples de β-réductions :

$$\begin{array}{lll} (\lambda x.x)t & \longrightarrow_\beta & x[t/x] = t \\ (\lambda x.\lambda y.x(y))(\lambda x.x) & \longrightarrow_\beta & (\lambda x.\lambda y.x(y))[\lambda x.x/x] = \lambda y.(\lambda x.x)(y) \\ \lambda y.(\lambda x.x)(y) & \longrightarrow_\beta & \lambda y.(x[y/x]) = \lambda y.y \end{array}$$

On peut répéter les β-réductions (on note $\longrightarrow_\beta^*$ quand il y a un nombre quelconque de β-réductions) :

$$(\lambda x.\lambda y.x(y))(\lambda x.x) \quad \longrightarrow_\beta^* \quad \lambda y.y$$

λ-calcul : β-réduction

Il existe des termes qui β-réduisent indéfiniment :

$$\begin{aligned}(\lambda x.x(x))(\lambda x.x(x)) &\longrightarrow_\beta x(x)[\lambda x.x(x)/x] = (\lambda x.x(x))(\lambda x.x(x)) \\ &\dots \\ &\longrightarrow_\beta (\lambda x.x(x))(\lambda x.x(x)) \\ &\dots\end{aligned}$$

Il existe aussi des termes qui ne peuvent plus se réduire : on dit que ces termes sont **sous forme normale**.
Un terme-programme a fini d'être exécuté quand il est sous forme normale.

λ-calcul : arithmétique

On peut représenter les entiers et les opérations sur les entiers :

- $0 \stackrel{\text{def}}{=} \lambda fy.y$
- $1 \stackrel{\text{def}}{=} \lambda fy.fy$
- $n \stackrel{\text{def}}{=} \lambda fy.f(f(...(fy)...)$ (f est appliqué n fois)
- $\texttt{succ} \stackrel{\text{def}}{=} \lambda nfz.f(nfz)$
- $\texttt{plus} \stackrel{\text{def}}{=} \lambda nmfx.nf(mfx)$
- $\texttt{mult} \stackrel{\text{def}}{=} \lambda nmf.n(mf)$
- $\texttt{exp} \stackrel{\text{def}}{=} \lambda nm.mn$

On peut montrer par récurrence sur n que les opérations sont correctes.

Remarque : tout est fonction, et tout est noté comme fonction appliqué à des arguments.
C'est pourquoi on aura le terme `plus`23 au lieu de $2 + 3$, etc.

λ-calcul : arithmétique

Exemples : (on note $f^n y$ pour $f(f(...(fy)...))$

$$\begin{array}{rcl}
\texttt{succ}\ 0 & = & (\lambda nfz.f(nfz))(\lambda fy.y) \\
& \longrightarrow_\beta & \lambda fz.f((\lambda fy.y)fz) \\
& \longrightarrow_\beta & \lambda fz.f((\lambda y.y)z) \\
& \longrightarrow_\beta & \lambda fz.fz \\
& = & 1
\end{array}$$

$$\begin{array}{rcl}
\texttt{succ}\ n & = & (\lambda nfz.f(nfz))(\lambda fy.f^n y) \\
& \longrightarrow_\beta & \lambda fz.f((\lambda fy.f^n y)fz) \\
& \longrightarrow_\beta & \lambda fz.f((\lambda y.f^n y)z) \\
& \longrightarrow_\beta & \lambda fz.f(f^n z) = \lambda fz.f^{n+1} z \\
& = & n+1
\end{array}$$

λ-calcul : arithmétique

Un dernier calcul

$$
\begin{array}{rcl}
\texttt{plus}\ 2\ 3 & = & (\lambda nmfx.nf(mfx))(\lambda fy.f^2y)(\lambda fy.f^3y) \\
& \longrightarrow_\beta & (\lambda mfx.(\lambda fy.f^2y)f(mfx))(\lambda fy.f^3y) \\
& \longrightarrow_\beta & (\lambda mfx.(\lambda y.f^2y)(mfx))(\lambda fy.f^3y) \\
& \longrightarrow_\beta & (\lambda mfx.f^2(mfx))(\lambda fy.f^3y) \\
& \longrightarrow_\beta & \lambda fx.f^2((\lambda fy.f^3y)fx) \\
& \longrightarrow_\beta & \lambda fx.f^2((\lambda y.f^3y)x) \\
& \longrightarrow_\beta & \lambda fx.f^2(f^3x)) = \lambda fx.f^5x \\
& = & 5
\end{array}
$$

λ-calcul : Booléens et conditionnelle

On peut représenter les booléens et la conditionnelle :

- $\texttt{Vrai} \stackrel{\text{def}}{=} \lambda xy.x$
- $\texttt{Faux} \stackrel{\text{def}}{=} \lambda xy.y$
- $\texttt{siAlorsSinon} \stackrel{\text{def}}{=} \lambda xyz.xyz$

Ainsi :

$$\begin{array}{rcl} \texttt{siAlorsSinon}\ tuv & = & (\lambda xyz.xyz)tuv \\ & \longrightarrow_\beta^* & tuv \end{array}$$

Et :

- Si t est vrai : $tuv = (\lambda xy.x)uv \longrightarrow_\beta (\lambda y.u)v \longrightarrow_\beta u$
- Si t est faux : $tuv = (\lambda xy.y)uv \longrightarrow_\beta (\lambda y.y)v \longrightarrow_\beta v$

λ-calcul : Booléens et conditionnelle

On peut représenter les opérations logiques, le test à zéro :

- $\texttt{et} \stackrel{\text{def}}{=} \lambda xy.xy\ \texttt{Faux}$
- $\texttt{ou} \stackrel{\text{def}}{=} \lambda xy.x\ \texttt{Vrai}\ y$
- $\texttt{non} \stackrel{\text{def}}{=} \lambda xyz.xzy$
- $\texttt{iszero} \stackrel{\text{def}}{=} \lambda x.x(\lambda y.\texttt{Faux})\ \texttt{Vrai}$

Ainsi :

$$\begin{array}{rcl} \texttt{non Vrai} & = & (\lambda xyz.xzy)(\lambda xy.x) \\ & \longrightarrow_\beta & \lambda yz.(\lambda xy.x)zy \\ & \longrightarrow_\beta & \lambda yz.(\lambda y.z)y \\ & \longrightarrow_\beta & \lambda yz.z = \texttt{Faux} \end{array}$$

Et encore :

$$\begin{array}{rcl} \texttt{iszero}\ n & = & (\lambda x.x(\lambda y.\texttt{Faux})\ \texttt{Vrai})n \\ & \longrightarrow_\beta & n(\lambda y.\texttt{Faux})\ \texttt{Vrai} \end{array}$$

Si $n = 0$:

$$\begin{array}{rcl} 0(\lambda y.\texttt{Faux})\ \texttt{Vrai} & = & (\lambda fy.y)(\lambda y.\texttt{Faux})\ \texttt{Vrai} \\ & \longrightarrow_\beta & (\lambda y.y)\ \texttt{Vrai} \\ & \longrightarrow_\beta & \texttt{Vrai} \end{array}$$

sinon n est de la forme $\lambda fy.ft$:

$$\begin{array}{rcl} n(\lambda y.\texttt{Faux})\ \texttt{Vrai} & = & (\lambda fy.ft)(\lambda y.\texttt{Faux})\ \texttt{Vrai} \\ & \longrightarrow_\beta & (\lambda y.(\lambda y.\texttt{Faux})t)\ \texttt{Vrai} \\ & \longrightarrow_\beta & (\lambda y.\texttt{Faux})t) \\ & \longrightarrow_\beta & \texttt{Faux} \end{array}$$

λ-calcul : Appel récursif

Une fonction récursive est en fait un point fixe.
Exemple : la fonction factorielle est définie par

*fact(n) == si n=0 alors 1 sinon n*fact(n-1)*

En λ-calcul, cela donne :

$$\texttt{fact}\ n = \texttt{siAlorsSinon}(\texttt{iszero}\ n)1(\texttt{mult}\ n(\texttt{fact}(\texttt{pred}\ n)))$$

(le terme `pred` représente la fonction prédécesseur : voir plus loin)
donc

$$\texttt{fact} = \lambda n.\texttt{siAlorsSinon}(\texttt{iszero}\ n)1(\texttt{mult}\ n(\texttt{fact}(\texttt{pred}\ n)))$$

d'où

$$\texttt{fact} = (\lambda fn.\texttt{siAlorsSinon}(\texttt{iszero}\ n)1(\texttt{mult}\ n(f(\texttt{pred}\ n))))\texttt{fact}$$

Si on note
$FACT = \lambda fn.\texttt{siAlorsSinon}(\texttt{iszero}\ n)1(\texttt{mult}\ n(f(\texttt{pred}\ n)))$, on a donc $\texttt{fact} = FACT\ \texttt{fact}$, c'est à dire que `fact` est le point fixe du terme *FACT*.

λ-calcul : opérateur de point fixe

Les termes suivants sont des **opérateurs de point fixe** :

- $Y \stackrel{\text{def}}{=} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
- $\Theta \stackrel{\text{def}}{=} (\lambda xy.y(xxy))(\lambda xy.y(xxy))$

On peut montrer que $Yt \equiv t(Yt)$ et que $\Theta t \equiv t(\Theta t)$

($\equiv$ est la plus petite relation d'équivalence telle que $u \equiv v$ si $u \longrightarrow_\beta^* v$ ou $v \longrightarrow_\beta^* u$)

Ainsi la fonction factorielle peut être définie comme :

$$\texttt{fact} = Y\ FACT$$

On peut montrer que toute fonction calculable (i.e., pour laquelle il existe un programme en C, en Java, ... qui la calcule) est représentable par un terme en λ-calcul, et vice-versa.

Autres propriétés importantes :

- Si t admet une forme normale, elle est unique.
- (Church-Rosser) si $t \longrightarrow_\beta^* u$ et $t \longrightarrow_\beta^* v$ alors il existe w tel que $u \longrightarrow_\beta^* w$ et $v \longrightarrow_\beta^* w$

λ-calcul : typage simple

On associe à un terme un type (=formule) par les règles ci-dessous où $t : A$ signifie le terme t a le type A

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i} \qquad \frac{x : A, \Gamma \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash u : A \quad \Gamma \vdash t : A \Rightarrow B}{\Gamma \vdash t(u) : B}$$

La règle droite peut être vue comme une règle de coupure. Eliminer la coupure dans la preuve c'est trouver le résultat de l'exécution du terme $t(u)$.

λ-calcul : typage simple

Exemple : Au terme $\lambda xy.x$ on peut associer le type $A \Rightarrow (B \Rightarrow A)$:

$$\dfrac{\dfrac{\dfrac{}{x : A, y : B \vdash x : A}}{x : A \vdash \lambda y.x : B \Rightarrow A}}{\vdash \lambda xy.x : A \Rightarrow (B \Rightarrow A)}$$

Aux entiers on peut associer le type $(A \Rightarrow A) \Rightarrow (A \Rightarrow A)$: (par induction sur n)

$$\dfrac{\dfrac{\dfrac{}{f : A \Rightarrow A, y : A \vdash y : A}}{f : A \Rightarrow A \vdash \lambda y.y : A \Rightarrow A}}{\vdash \lambda fy.y : (A \Rightarrow A) \Rightarrow (A \Rightarrow A)}$$

$$\dfrac{\dfrac{\dfrac{\vdots \text{ induction}}{f : A \Rightarrow A, y : A \vdash f^n y : A} \quad \dfrac{}{f : A \Rightarrow A, y : A \vdash f : A \Rightarrow A}}{f : A \Rightarrow A, y : A \vdash f^{n+1} y : A}}{\dfrac{f : A \Rightarrow A \vdash \lambda y.f^{n+1} y : A \Rightarrow A}{\vdash \lambda fy.f^{n+1} y : (A \Rightarrow A) \Rightarrow (A \Rightarrow A)}}$$

λ-calcul : typage simple

Remarque : tout terme n'est pas simplement typable !

Théorème

Tout terme simplement typable est fortement normalisable, i.e., l'ordre des réductions n'importe pas et termine toujours.

VI : Calcul de Hilbert

Définition

*Un système de preuves à la Hilbert est donné par un ensemble d'***axiomes** *et une seule* **règle**.

- *Un axiome est une formule.*
- *Une règle est un ensemble fini de formules appelées* **prémisses** *et une formule appelée* **conclusion**.

Définition

Soit Γ *un ensemble de formules, une* **preuve** Π *de la formule* F_n *sous les hypothèses* Γ *est une suite de formules* $F_1, \ldots, F_n$ *telle que pour tout* $1 \leq i \leq n$ *:*

- *soit* F_i *est une instance d'axiome,*
- *soit* $F_i \in \Gamma$*,*
- *soit* F_i *est la conclusion d'une instance de règle dont les prémisses sont des formules* F_j *avec* $j < i$*.*

On note $\Gamma \vdash F_n$*.*

Pour le calcul propositionnel :

- 1 seule règle, le **modus ponens** : à partir de A et de $A \Rightarrow B$, on déduit B.

$$\frac{A \quad (A \Rightarrow B)}{B}\ MP$$

- 11 axiomes :

H_1 : $(A \Rightarrow (B \Rightarrow A))$
H_2 : $((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$
H_3 : $(A \Rightarrow \neg\neg A)$
H_4 : $(\neg\neg A \Rightarrow A)$
H_5 : $((A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A))$
H_6 : $(A \Rightarrow (B \Rightarrow (A \wedge B)))$
H_7 : $((A \wedge B) \Rightarrow A)$
H_8 : $((A \wedge B) \Rightarrow B)$
H_9 : $(A \Rightarrow (A \vee B))$
H_{10} : $(B \Rightarrow (A \vee B))$
H_{11} : $((((A \vee B) \wedge (A \Rightarrow C)) \wedge (B \Rightarrow C)) \Rightarrow C)$

Exemple

Soit F, G, H trois formules propositionnelles. Une preuve de $F_7 = (F \Rightarrow H)$ à partir de $\{(F \Rightarrow G), (G \Rightarrow H)\}$ est la suivante :

$F_1 =$	$(G \Rightarrow H)$	(hypothèse)
$F_2 =$	$((G \Rightarrow H) \Rightarrow (F \Rightarrow (G \Rightarrow H)))$	(axiome H_1)
$F_3 =$	$(F \Rightarrow (G \Rightarrow H))$	(MP sur F_1 et F_2)
$F_4 =$	$((F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H)))$	(axiome H_2)
$F_5 =$	$((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$	(MP sur F_3 et F_4)
$F_6 =$	$(F \Rightarrow G)$	(hypothèse)
$F_7 =$	$(F \Rightarrow H)$	(MP sur F_6 et F_5)

Théorème (Theorème de la déduction)

Soit Γ un ensemble de formules propositionnelles, F et G deux formules propositionnelles.

$$\Gamma \vdash F \Rightarrow G \text{ ssi } \Gamma \cup \{F\} \vdash G$$

Théorème (Validité)

Toute formule propositionnelle prouvable est une tautologie.

Les axiomes sont des tautologies, tautologie stable par substitution et MP. Récurrence sur la longueur de la preuve.

Théorème (Complétude)

Toute tautologie est prouvable (par modus ponens).