Logique : introduction, logique des propositions

Logique et Intelligence Artificielle (Licence 3)

Introduction

La **logique** peut se définir comme l'étude des **« raisonnements »** :

- Raisonnements « formels » : démonstrations mathématiques, preuves.
- Raisonnements « informels » : étude des discours en langues naturelles, utilisation des connaissances.

Logique en informatique

La logique apparaît dans différents aspects de l'informatique :

1. La logique mathématique sert pour l'*analyse formelle de programmes* : formalisation des spécifications, preuves de programmes, optimisations...
2. L'informatique peut aider les mathématiques dans l'établissement de preuves : assistants de preuves, démonstrateurs automatiques.
3. Plus généralement, la logique est une base de **l'intelligence artificielle**, avec de nombreux aspects :
   - représentation, acquisition et utilisation de connaissances (*systèmes experts ou à base de connaissances*) ;
   - résolution de problèmes (*robots, jeux*) ;
   - compréhension et traitement des langues naturelles (*linguistique informatique*).

Approche du cours

Ce cours comprend deux parties :

- Une présentation des logiques les plus simples (logique propositionnelle, puis logique des prédicats), dans l'optique d'une utilisation par un ordinateur.
- L'apprentissage d'un langage de **programmation logique** (Prolog) dans l'optique de résolution de problèmes de logique et de *contraintes*.

Logique : langage, syntaxe, sémantique

Toute *formule* (toute écriture, tout programme) n'est qu'une suite (finie) de symboles.

```
10001001011
```

```
(a => b) <=> (non b => non A)
```

```
Félix est un chat.
```

```
int main () { printf("Hello, world !\n"); return 0; }
```

Une formule est (ou n'est pas) « correcte » dans un **langage**. Le « sens » d'une formule « correcte » dans un langage est appelé sa **sémantique**.

Langage : syntaxe

Un *langage formel* est un langage *formalisé* (strictement défini).

La **syntaxe** d'un langage formel est l'ensemble des règles définissant si un mot est « correct » dans le langage (on parle de mots **acceptés** par le langage).

Exemples : langages réguliers...

Exemple : en C, un entier non signé est écrit sous forme décimale (`123`, `410`, ...), en octal (`0123`, `0410`...), en binaire (`1010010101b`, ...) ou en hexadécimal (`0x123`, `0x410`). Tous ces mots peuvent définir un seul langage : on peut écrire une fonction C qui dit si une chaîne de caractère représente (ou pas) un entier non signé.

Langage : sémantique

La **sémantique** d'un langage est une *relation* entre les formules acceptées par le langage et leurs *significations*.

Cette *signification* est souvent faite dans un univers mathématique (dont le sens est déjà connu).

Exemple : sémantiques possibles des entiers écrits en C :

- une fonction associant un mot (`123`, `0x123`, `1001b`, ...) à un entier mathématique (`123` $\mapsto$ 123, `0x123` $\mapsto$ 291, `1001b` $\mapsto$ 9, ...).
- une fonction associant un mot (`123`, `0x123`, `1001b`, ...) à un représentation mémoire sur 4 octets (`123` $\mapsto$ |00000000|00000000|00000000|01111011|).

Sémantique en logique : interprétation

En logique, une formule doit être considérée comme une « phrase » (comme, en langue naturelle, « Nous sommes à Brest », « Félix est un chat », etc.), qui peut être « vraie » ou « fausse ».

Mais la vérité n'est pas absolue, elle peut dépendre de la « réalité », du « monde » dans lequel on est.

Ainsi, on définit en général la sémantique des formules ainsi :

- on crée un ensemble d'**univers** (on parle aussi de **mondes** ou de **modèles**) possibles ;
- la *sémantique* d'une formule dit, pour chaque univers, si cette formule est vraie ou non dans cet univers : c'est l'**interprétation** de la formule.

Exemple

L'*énigme « d'Einstein »* commence ainsi :

- *Dans une rue, il y a 5 maisons de 5 couleurs différentes (rouge, bleu, vert, blanc et jaune) ;*
- *dans chaque maison vit une personne de nationalité différente (anglais, suédois, danois, norvégien, allemand) ;*
- *chaque personne boit un certain type de boisson (thé, café, lait, bière, eau), fume un certain type de cigare (Pall Mall, Dunhill, Blend, Blue Master, Prince) et garde un certain animal domestique (chiens, oiseaux, chats, cheval, poissons).*

Avec ces informations, on dispose de $5! \times 5! \times 5! \times 5! \times 5! = 24\,883\,200\,000$ « mondes » possibles. La **sémantique** de la phrase « La maison verte est à gauche de la maison blanche. » est la fonction qui prend en paramètre la description d'un monde, et retourne vrai si dans ce monde la maison verte est à gauche de la maison blanche, et faux sinon. Ce résultat est l'*interprétation* de la phrase dans ce monde.

Raisonnements logiques

Le principe d'un raisonnement logique est de « faire » de nouvelles formules à partir d'anciennes formules.

Exemple : à partir des deux formules :

1. L'anglais boit du thé.
2. Un voisin de l'allemand boit du thé.

on « déduit » :

– L'anglais est voisin de l'allemand.

Cette déduction est **correcte** parce que, dans tous les mondes possibles pour lesquels les deux premières formules sont vraies, la troisième est vraie.

Méthode de preuve

Une *méthode de preuve* est une procédure (formelle) pour faire des raisonnements logiques (donc créer de nouvelles formules logiques à partir d'autres). Une telle méthode est normalement **syntaxique**, dans le sens qu'elle manipule les formules sans chercher à les interpréter.

- une telle méthode est **correcte** si les déductions faites sont **correctes**, c'est-à-dire « sémantiquement vraies ».
- une telle méthode est **complète** si toutes les déductions « sémantiquement vraies » peuvent être obtenues avec cette méthode.

En informatique, en plus de la correction et la complétude, on pourra s'intéresser à deux autres propriétés :

- un ordinateur peut-il **vérifier** une preuve, si on la lui donne ?
- un ordinateur peut-il **fabriquer** une preuve, si on lui pose une question ?

Logique propositionnelle

La **logique propositionnelle** ou **logique des propositions** est une des logiques les plus simples : juste des **variables** (des affirmations « élémentaires ») et des **connecteurs logiques** (*et*, *ou*, *implique*, ...).

Les formules contiennent :

– Un **Ensemble de variables** noté $\mathcal{X}$. Par exemple,

$$\mathcal{X} = \{a, b, c, d, e, f, \ldots\}.$$

– Des **connecteurs logiques**, en nombre fini. Par exemple :

$$\{ \vee, \ \wedge, \ \rightarrow, \ \neg \}$$

Syntaxe

Construction par **induction** de l'ensemble des formules $\mathcal{F}$:

1. Toute variable est une formule : $\mathcal{X} \subseteq \mathcal{F}$.
2. Si Φ et Φ' sont des formules, alors $\Phi \vee \Phi'$, $\Phi \wedge \Phi'$, $\Phi \rightarrow \Phi'$ et $\neg\Phi$ sont des formules.

L'ensemble $\mathcal{F}$ est **le plus petit ensemble** qui vérifie ces deux propriétés. Toute formule est construite par *application* finie de ces deux règles.

Exemple de formules

Sont des formules :

- a ;
- $a \to b$;
- $\neg\neg a$;
- $a \wedge b \vee c$;
- $a \wedge (b \vee c)$ (utilisation du parenthèsage pour expliciter l'*arbre de dérivation*) ;
- $a \to a \to a \to a \to a \to a \to a \to a \to a$.

Ne sont pas des formules :

- $a \to\to a$: expression *mal formée*.
- $a \to a \to a \ldots$ à l'infini : une formule est forcément **finie**.

Parenthésage

Priorités : $\neg$ est prioritaire sur $\wedge$, qui l'est sur $\vee$, qui l'est sur $\rightarrow$.

Application : $c \rightarrow a \vee \neg b \wedge d$ est équivalent à $c \rightarrow (a \vee ((\neg b) \wedge d))$.

Associativité : $\vee$ et $\wedge$ sont **associatifs à gauche**, $\rightarrow$ est **associatif à droite**.

Application : $a \rightarrow b \rightarrow c$ est équivalent à $a \rightarrow (b \rightarrow c)$.

Sémantique

En logique classique, les stressvaleurs de vérité sont le *vrai* et le *faux*. Nous noterons ces valeurs par les symboles $\bot$ (pour faux) et $\top$ (pour vrai).

$$\mathbb{B} = \{\bot, \top\}$$

Les *mondes* en logique des propositions donnent l'**interprétation** des variables propositionnelles (vraie ou fausse).

Définition : une **interprétation** ou une **affectation** ρ est une application de $\mathcal{X}$ dans $\mathbb{B}$.

Exemple

Si $\mathcal{X} = \{a, b, x\}$, on a $2^3 = 8$ *mondes* possibles, définis par 8 affectations :

1. $\rho_1(a) = \bot$, $\rho_1(b) = \bot$, $\rho_1(x) = \bot$.
2. $\rho_2(a) = \bot$, $\rho_2(b) = \bot$, $\rho_2(x) = \top$.
3. $\rho_3(a) = \bot$, $\rho_3(b) = \top$, $\rho_3(x) = \top$.
4. $\rho_4(a) = \bot$, $\rho_4(b) = \top$, $\rho_4(x) = \bot$.
5. $\rho_5(a) = \top$, $\rho_5(b) = \top$, $\rho_5(x) = \bot$.
6. $\rho_6(a) = \top$, $\rho_6(b) = \top$, $\rho_6(x) = \top$.
7. $\rho_7(a) = \top$, $\rho_7(b) = \bot$, $\rho_7(x) = \top$.
8. $\rho_8(a) = \top$, $\rho_8(b) = \bot$, $\rho_8(x) = \bot$.

Sémantique d'une formule

La **sémantique** $[\![\Phi]\!]\,\rho$ d'une formule Φ dans l'interprétation ρ est définie par *récurrence structurelle* sur Φ :

- pour une variable x, $[\![x]\!]\,\rho = \rho(x)$.
- $[\![\Phi \vee \Phi']\!]\,\rho = [\![\Phi]\!]\,\rho \oplus [\![\Phi']\!]\,\rho$.
- $[\![\Phi \wedge \Phi']\!]\,\rho = [\![\Phi]\!]\,\rho \otimes [\![\Phi']\!]\,\rho$.
- $[\![\Phi \to \Phi']\!]\,\rho = [\![\Phi]\!]\,\rho \Rightarrow [\![\Phi']\!]\,\rho$.
- $[\![\neg\Phi]\!]\,\rho = \neg\,[\![\Phi]\!]$.

où les opérateurs $\oplus$, $\otimes$, $\Rightarrow$ et $\neg$ sont définis à l'aide de *tables de vérité* (premier argument en colonne, deuxième en ligne) :

$\oplus$	$\bot$	$\top$
$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$

$\otimes$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\top$

$\Rightarrow$	$\bot$	$\top$
$\bot$	$\top$	$\bot$
$\top$	$\top$	$\top$

$\neg$	
$\bot$	$\top$
$\top$	$\bot$

Note explicative sur les symboles

L'usage de $\vee$, $\wedge$, ... et $\oplus$, $\otimes$, ... peut prêter à confusion :

- $\vee$, $\wedge$, $\neg$, $\rightarrow$ sont des **symboles** utilisés pour écrire des formules en logique propositionnelle. Ce sont juste des *dessins* dans une *écriture*.
- $\oplus$, $\otimes$, $\overline{\neg}$, $\Longrightarrow$ sont des **fonctions** de $B \times B$ dans B qui servent à construire la sémantique des formules. Ce sont des *objets mathématiques*.

La différence peut s'apparenter à la différence entre `"+"` et `+` en C. De même, le calcul de $[\![\Phi]\!]\,\rho$ est proche de de ce que ferait un interpréteur C pour interpréter une expression arithmétique en fonction de la mémoire courante.

Exemples

Soit $\Phi = a \to (b \vee c) \wedge (\neg a \to a \to b)$.

Soit ρ avec $\rho(a) = \top$, $\rho(b) = \top$ et $\rho(c) = \bot$.

1. Construction (mentale ou non) de l'arbre de dérivation.
2. Sémantique de chaque sous-formule.

Au final : $[\![\Phi]\!]\,\rho = \top$.

Modèle, validité, satisfiabilité

On dit que Φ est *vraie* dans l'affectation ρ lorsque $[\![\Phi]\!]\,\rho = \top$, et Φ est *fausse* dans l'affectation ρ lorsque $[\![\Phi]\!]\,\rho = \bot$.

Autres définitions :

- ρ est un **modèle** de Φ, ou ρ **satisfait** Φ, si $[\![\Phi]\!]\,\rho = \top$. On note aussi $\rho \models \Phi$.
- soit Γ un ensemble de formules. Γ **entraîne** Φ, si toute affectation satisfaisant toutes les formules de Γ, satifait Φ (les *modèles* de Γ sont des *modèles* de Φ). On note $\Gamma \models \Phi$.
- Φ est **valide** si elle est vraie dans **toutes** les affectations. Sinon elle est **invalide** (il existe au moins **une** affectation dans laquelle est est fausse). Si Φ est valide, on dit aussi que Φ est une **tautologie** et on note $\models \Phi$.
- Φ est **satisfiable** si elle a au moins **un** modèle. Elle est **insatisfiable** (fausse dans tout modèle) sinon.

Exemples

A vérifier :

- Soit $\Gamma = \{a, a \to b\}$. Alors $\Gamma \models b$.
- $\models a \to b \to a$.
- $a \to b \to \neg a$ n'est pas valide, mais est satisfiable.
- $\neg b \wedge a \wedge (\neg b \to \neg a)$ est insatisfiable.

Introduction à la preuve

On dispose d'une formule Φ. Comment savoir si Φ est valide ? invalide ? insatisfiable ?

Même chose avec un ensemble de formule Γ et une formule Φ : est ce que $\Gamma \models \Phi$?

Méthode des tables de vérité

On peut **énumérer** les affectations possibles (sur les variables libres Φ), soit 2^n cas où n est le nombre de variables libres de Φ. Cette méthode « *sémantique* » revient à faire un *table de vérité* de la formule.

Exemple :

a	b	$a \to b \to \neg a$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\bot$

Avantages : simple à comprendre, donne tout de suite les modèles.

Inconvénient : *très* lent, complexité exponentielle.

Approche « calculatoire » : systèmes de réécriture

Utilisation de **règles de réécriture** (associativité, commutativité, ...) pour passer d'une formule à une formule « équivalente », jusqu'à arriver à une formule « évidente ».

Exemple :

$$a \to b \to a \;=\; \neg a \vee (b \to a) \;=\; \neg a \vee \neg b \vee a \;=\; \neg a \vee a \vee \neg b \;=\; \mathbf{V} \vee \neg b \;=\; \mathbf{V}$$

Inconvénient : les calculs sont assez loin d'un raisonnement logique. Leur caractère bidirectionnel ($a \to b \to a$ est plus simple que $\neg a \vee (b \to a)$ ou l'inverse ?) les rend difficile à programmer.

L'étude des systèmes de réécriture sort du cadre de ce cours, mais on se servira parfois de règles de réécriture avant l'utilisation d'un système de preuve.

Note : sauf dans des cas précisés au cours du cours (et pour des règles spécifiques), **ne pas utiliser la réécriture** dans les exercices (même les formes les plus évidentes). Rien que réécrire $a \vee (a \vee b)$ en $(a \vee a) \vee b$ n'est pas trivial pour un ordinateur.

Méthodes « logiques » : les systèmes de preuve

Une autre approche est de créer des règles pour écrire des suites de formules, pour que l'application de ces règles permette d'obtenir des formules *valides*.

Une suite de formules respectant ces règles est une **preuve**, on parle de **systèmes de preuves**. Les conclusions (dernières formules) de ces preuves sont appelées **théorèmes**.

Questions : étant donné un système de preuve :

1. tout théorème est-il une tautologie (**correction**) ?
2. toute tautologie peut-elle être un théorème (**complétude**) ?

Systèmes de Hilbert

Un système de preuve en **format de Hilbert** comprend deux ensembles :

1. un ensemble $\mathcal{A}$ d'**axiomes**, qui sont des formules ;
2. un ensemble $\mathcal{R}$ de **règles d'inférence**, relations entre ensembles de formules (les *prémisses*) et des formules (les *conclusions*).

Dérivation

A partir d'un ensemble Γ de formules (appelées **hypothèses**), une **dérivation** est une suite de formules Φ_1, Φ_2, ..., Φ_n, $n > 0$, telle que chaque formule Φ_i est :

- soit un axiome, $\Phi_i \in \mathcal{A}$;
- soit une hypothèse, $\Phi_i \in \Gamma$;
- soit une *déduction* des formules précédentes : Φ_i est la conclusion d'une règle d'inférence, donc les prémisses sont dans les formules précédentes.

Une **preuve** d'une formule Φ à partir de Γ est alors une dérivation dont la dernière formule est Φ. Lorsqu'une telle preuve existe, on dit que Φ est **prouvable** à partir de Γ, et on note $\Gamma \vdash \Phi$.

Schémas d'axiomes, de règles Les axiomes (et les règles) sont en nombre infinis, présentés par **schémas** : la définition se fait à substitution près.

Exemple : schéma d'axiome : $\Phi \to \Phi' \to \Phi$.

- $a \to b \to a$.
- $a \to a \to a$.
- $(a \vee b) \to (c \to a) \to (a \vee b)$.
- ...

Exemple : schéma de règle : « à partir de Φ et de $\Phi \to \Phi'$, déduire Φ' ».

- de a et de $a \to a$, déduire a.
- de $a \wedge b$ et de $(a \wedge b) \to (a \vee b)$, déduire $a \vee b$.
- ...

On note souvent les règles sous la forme $\dfrac{\text{prémisses}}{\text{conclusion}}$.

Système « $\mathcal{SKC}$ » Un système minimal pour les opérateurs $\neg$ et $\rightarrow$.

Trois schémas d'axiome :

$(\mathcal{K})$ $\quad \Phi \rightarrow \Phi' \rightarrow \Phi.$

$(\mathcal{S})$ $\quad (\Phi \rightarrow \Phi' \rightarrow \Phi'') \rightarrow (\Phi \rightarrow \Phi') \rightarrow \Phi \rightarrow \Phi''.$

$(\mathcal{C})$ $\quad (\neg\Phi \rightarrow \neg\Phi') \rightarrow \Phi' \rightarrow \Phi.$

Un schéma de règle : le **modus ponens**.

$$\frac{\Phi \quad \Phi \rightarrow \Phi'}{\Phi'}$$

Autres opérateurs

Le système de Hilbert ne contient que les opérateurs $\rightarrow$ et $\neg$, il faut donc un moyen de transformer une formule comprenant d'autres opérateurs.

Les autres opérateurs sont *réécrits* en fonction de $\rightarrow$ et $\neg$:

$$\begin{aligned} \Phi \wedge \Phi' &= \neg(\Phi \rightarrow \neg\Phi') \\ \Phi \vee \Phi' &= \neg\Phi \rightarrow \Phi' \end{aligned}$$

Seules règles de réécriture autorisées, et uniquement de la gauche vers la droite : il s'agit en fait de *définition* des opérateurs $\wedge$ et $\vee$ à partir de $\rightarrow$.

Exemple : preuve de $a \vdash b \vee a$

$b \vee a$ est en fait $\neg b \to a$. On veut donc prouver

$$a \vdash \neg b \to a$$

1. $a \to \neg b \to a$ $(\mathcal{K})$ (en remplaçant Φ par a et Φ' par $\neg b$).
2. a (hypothèse).
3. $\neg b \to a$ (MP avec (1) et (2)).

En utilisant uniquement l'hypothèse a, et le système de Hilbert, on a bien obtenu $\neg b \to a$, ce qui prouve $a \vdash \neg b \to a$.

Exemple : preuve de $a \vdash a \vee b$ En fait, on veut prouver : $a \vdash \neg a \to b$.

1. $(\neg a \to a \to b) \to (\neg a \to a) \to \neg a \to b$ $(\mathcal{S})$
2. $(\neg a \to (\neg b \to \neg a) \to a \to b) \to (\neg a \to \neg b \to \neg a) \to \neg a \to a \to b$ $(\mathcal{S})$
3. $(\neg b \to \neg a) \to a \to b$ $(\mathcal{C})$
4. $((\neg b \to \neg a) \to a \to b) \to \neg a \to (\neg b \to \neg a) \to a \to b$ $(\mathcal{K})$
5. $\neg a \to (\neg b \to \neg a) \to a \to b$ (MP avec (3) et (4))
6. $(\neg a \to \neg b \to \neg a) \to \neg a \to a \to b$ (MP avec (2) et (5))
7. $\neg a \to \neg b \to \neg a$ $(\mathcal{K})$
8. $\neg a \to a \to b$ (MP avec (6) et (7))
9. $(\neg a \to a) \to \neg a \to b$ (MP avec (1) et (8)).
10. $a \to \neg a \to a$ $(\mathcal{K})$
11. a (hypothèse)
12. $\neg a \to a$ (MP avec (10) et (11))
13. $\neg a \to b$ (MP avec (9) et (12), **CQFD**).

Améliorations

Pour rendre une preuve plus lisible :

1. Utiliser des arbres de preuves.
2. Présenter à l'aide de preuves intermédiaires (lemmes).
3. Utiliser des substitutions : *si* Φ *est prouvable sans hypothèses, alors pour toute substitution* ρ, $\Phi\sigma$ *est prouvable*. De même, on fait plusieurs fois des démonstrations de la forme $\Phi \vdash \Phi' \to \Phi$.

Malgré tout, les systèmes de Hilbert sont peu utilisables.

Correction

Le système $\mathcal{SKC}$ est **correct**, c'est-à-dire que pour tout ensemble de formules Γ et toute formule Φ :

$$\Gamma \vdash \Phi \;\Rightarrow\; \Gamma \models \Phi$$

En particulier, toute formule **prouvable** sans hypothèses est **valide**.

Cela découle de la compatibilité entre chaque axiome, chaque règle de déduction, et la sémantique des opérateurs.

Complétude

Le système $\mathcal{SKC}$ est **complet**, c'est-à-dire que si Γ entraîne Φ, alors Φ est prouvable à partir de Γ :

$$\Gamma \models \Phi \;\Rightarrow\; \Gamma \vdash \Phi$$

En particulier, toute **tautologie** est **prouvable** (ce qui ne veut pas dire que sa preuve soit facile à faire).

Autres axiomes

On peut définir des systèmes à partir d'autres opérateurs : par exemple, sans $\neg$ mais avec un symbole $\mathbf{F}$ d'arité 0 ($\mathbf{F}$ est une formule à lui seul), on remplace le schéma $(\mathcal{C})$ par :

$$((\Phi \to \mathbf{F}) \to \mathbf{F}) \to \Phi$$

Un tel système est aussi correct et complet (avec la bonne sémantique de $\mathbf{F}$). $\neg\Phi$ devient alors équivalent à $\Phi \to \mathbf{F}$.

Déduction naturelle

La **déduction naturelle** utilise des règles d'inférences pour *introduire* et *éliminer* des opérateurs, en cherchant à simuler le « raisonnement naturel ». Par exemple :

$$\frac{}{\Gamma, A \vdash A}(\text{Axiome}) \qquad \frac{\Gamma, \Phi \vdash \Phi'}{\Gamma \vdash \Phi \to \Phi'}(\to\text{I}) \qquad \frac{\Gamma \vdash \Phi \to \Phi' \quad \Gamma \vdash \Phi}{\Gamma \vdash \Phi'}(\to\text{E})$$

$$\frac{\Gamma \vdash \Phi \quad \Gamma \vdash \Phi'}{\Gamma \vdash \Phi \wedge \Phi'}(\wedge\text{I}) \qquad \frac{\Gamma \vdash \Phi \wedge \Phi'}{\Gamma \vdash \Phi}(\wedge\text{E1}) \qquad \frac{\Gamma \vdash \Phi \wedge \Phi'}{\Gamma \vdash \Phi'}(\wedge\text{E2})$$

$$\frac{\Gamma \vdash \Phi}{\Gamma \vdash \Phi \vee \Phi'}(\vee\text{I1}) \qquad \frac{\Gamma \vdash \Phi'}{\Gamma \vdash \Phi \vee \Phi'}(\vee\text{I2}) \qquad \frac{\Gamma \vdash \Phi \vee \Phi' \quad \Gamma, \Phi \vdash \Phi'' \quad \Gamma, \Phi' \vdash \Phi''}{\Gamma \vdash \Phi''}(\vee\text{E})$$

$$\frac{\Gamma, \Phi \vdash \mathbf{F}}{\Gamma \vdash \neg\Phi}(\neg\text{I}) \qquad \frac{\Gamma \vdash \Phi \quad \Gamma \vdash \neg\Phi}{\Gamma \vdash \Phi'}(\neg\text{E})$$

(règle $(\neg\text{I})$ inutile si $\neg\Phi$ est défini comme $\Phi \to \mathbf{F}$)

$$\frac{\Gamma \vdash \Phi}{\Gamma \vdash \neg\neg\Phi}(\neg\neg\text{I}) \qquad \frac{\Gamma \vdash \neg\neg\Phi}{\Gamma \vdash \Phi}(\neg\neg\text{E})$$

(règle $(\neg\neg\text{I})$ non indispensable)
(système NJ = pas de règle $(\neg\neg\text{E})$)

Exemple : preuve de $\vdash A \to \neg\neg A$ (sans utiliser ($\neg\neg$I))

$$\dfrac{\dfrac{\dfrac{A, \neg A \vdash A \qquad A, \neg A \vdash \neg A}{A, \neg A \vdash \mathbf{F}}}{A \vdash \neg\neg A}}{\vdash A \to \neg\neg A}$$

« Pour prouver "Si A alors non non A", on suppose que A est vrai et on va montrer "non non A", c'est-à-dire "non (non A)" (règle $\to$I). Pour cela, on va supposer que "non A" est vrai et obtenir une contradiction (démonstration dite *par l'absurde*, règle $\neg I$).

Comme on a supposé "A" et "non A", on a à la fois A et sa négation qui sont vraies, ce qui permet de déduire n'importe quoi (règle $\neg E$), donc par exemple la contradiction $\mathbf{F}$. Ce qui montre que "non A" ne peut pas être vrai, donc que "non non A" est vrai.

Donc, à partir de A on a montré "non non A", ce qui prouve "Si A alors non non A" ».

Correction et complétude

Théorème : le système de déduction naturelle ci-dessus est **correct** et **complet**.

Sans la règle $(\neg\neg E)$, le système est **correct**, mais pas **complet** (c'est la *logique intuitionniste*).

Exemple : preuve de $\vdash ((a \to b) \to a) \to a$ (loi de Pierce)

Cette tautologie n'est pas démontrable sans $(\neg\neg E)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(a \to b) \to a, a, \neg a \vdash a}\quad \dfrac{}{(a \to b) \to a, a, \neg a \vdash \neg a}}{\dfrac{}{(a \to b) \to a, a, \neg a \vdash b}}}{(a \to b) \to a, \neg a \vdash (a \to b) \to a}\quad \dfrac{}{(a \to b) \to a, \neg a \vdash a \to b}}{(a \to b) \to a, \neg a \vdash a}}{\dfrac{(a \to b) \to a, \neg a \vdash \neg a}{\dfrac{\neg a, (a \to b) \to a \vdash \mathbf{F}}{\dfrac{(a \to b) \to a \vdash \neg\neg a}{\dfrac{(a \to b) \to a \vdash a}{\vdash ((a \to b) \to a) \to a}}}}}$$

Calcul des séquents

La déduction naturelle est mieux mais pas parfaite : on ne sait pas toujours, à partir d'une formule, quelle règle appliquer.

Le **calcul des séquents** (de Gerhard Gentzen) est encore mieux. Un **séquent** de Gentzen est un couple d'ensemble de formules Γ, Δ, noté $\Gamma \vdash \Delta$. On trouve donc **plusieurs formules à droite du** $\vdash$.

Intuitivement, $\Gamma \vdash \Delta$ signifie que **sous les hypothèses** Γ**, au moins** ***une*** **formule de** Δ **est « vraie »** (conjonction à gauche du $\vdash$, **disjonction** à droite).

Toutes les règles (sauf deux) sont des règles d'introduction d'opérateurs, mais certaines se font *à droite* et d'autres *à gauche*.

Système LK_0 de Gentzen

$$\frac{}{\Gamma, \Phi \vdash \Delta, \Phi}(\text{Axiome})$$

$$\frac{\Gamma, \Phi, \Phi' \vdash \Delta}{\Gamma, \Phi \wedge \Phi' \vdash \Delta}(\wedge\text{-gauche}) \qquad \frac{\Gamma \vdash \Delta, \Phi \quad \Gamma \vdash \Delta, \Phi'}{\Gamma \vdash \Delta, \Phi \wedge \Phi'}(\wedge\text{-droite})$$

$$\frac{\Gamma, \Phi \vdash \Delta \quad \Gamma, \Phi' \vdash \Delta}{\Gamma, \Phi \vee \Phi' \vdash \Delta}(\vee\text{-gauche}) \qquad \frac{\Gamma \vdash \Delta, \Phi, \Phi'}{\Gamma \vdash \Delta, \Phi \vee \Phi'}(\vee\text{-droite})$$

$$\frac{\Gamma \vdash \Phi, \Delta \quad \Gamma, \Phi' \vdash \Delta}{\Gamma, \Phi \to \Phi' \vdash \Delta}(\to\text{-gauche}) \qquad \frac{\Gamma, \Phi \vdash \Delta, \Phi'}{\Gamma \vdash \Delta, \Phi \to \Phi'}(\to\text{-droite})$$

$$\frac{\Gamma \vdash \Delta, \Phi}{\Gamma, \neg\Phi \vdash \Delta}(\neg\text{-gauche}) \qquad \frac{\Gamma, \Phi \vdash \Delta}{\Gamma \vdash \Delta, \neg\Phi}(\neg\text{-droite})$$

$$\frac{\Gamma \vdash \Delta, \Phi \quad \Gamma', \Phi \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}(\text{Coupure})$$

Correction et complétude

Etant donné deux ensembles de formules Γ, Δ, on notera $\Gamma \models \Delta$ pour dire que, si une affectation satisfait **toutes** les formules de Γ, alors elle satisfait au moins **une** formule de Δ. On dit dans ce cas que Δ est une conséquence logique de Γ.

Théorème : le système $\mathbf{LK}_0$ de Gentzen en logique propositionnelle est **correct** et **complet**, ie :

$$\Gamma \vdash \Delta \iff \Gamma \models \Delta$$

Cela reste vrai même si on supprime la règle de *Coupure* (théorème dit *d'élimination des coupures*), qui sert dans les démonstrations « à la main ».

Intérêt du calcul des séquents Pour prouver $\Gamma \vdash \Delta$, on cherche à faire disparaître tous les connecteurs, jusqu'à n'avoir plus que des axiomes. Cette *stratégie* marche toujours (si effectivement $\Gamma \vdash \Delta$).

Exemple (preuve de $\vdash ((a \to b) \to a) \to a$) :

$$\dfrac{\dfrac{\dfrac{a \vdash a, b}{\vdash (a \to b), a} \quad a \vdash a}{(a \to b) \to a \vdash a}}{\vdash (((a \to b) \to a) \to a}$$