LOGIQUE

ARNAUD BODIN

Introduction

« Il existe des énoncés vrais mais indémontrables. » Cette phrase, qui peut sembler paradoxale, est une conséquence du célèbre théorème d'incomplétude de Gödel, démontré en 1931. Elle révèle une séparation fondamentale entre deux notions que l'on avait longtemps tendance à confondre : la vérité d'un énoncé et l'existence d'une démonstration de cet énoncé. Ce livre suit le chemin qui mène à cette découverte.

Pour comprendre ce résultat, il faut d'abord apprendre le langage de la logique. Les mathématiques reposent sur des raisonnements dont les règles, souvent utilisées sans être explicitées, peuvent être décrites avec précision. La logique permet ainsi de voir plus clairement comment une démonstration est construite et sur quelles règles elle repose.

Le livre est organisé en trois parties correspondant à trois niveaux de difficulté :

La logique Vrai/Faux (Niveau 1). C'est le langage logique le plus élémentaire. Nous y étudions les connecteurs « non », « et », « ou », « si…alors… », « si et seulement si » et les règles qui permettent de construire des démonstrations simples. Cette première étape permet de préciser la notion de vérité logique.

La logique du premier ordre (Niveau 2). L'introduction des quantificateurs « pour tout » et « il existe » permet d'exprimer les raisonnements mathématiques dans leur généralité. Nous étudions alors un langage suffisamment riche pour formaliser une grande partie des mathématiques usuelles.

Le théorème d'incomplétude (Niveau 3). Cette dernière partie rassemble progressivement les outils nécessaires à la démonstration du théorème de Gödel. Le lecteur découvrira comment un raisonnement mathématique peut être appliqué à lui-même et conduire à l'existence d'énoncés vrais mais impossibles à démontrer dans un système donné.

Les deux premières parties s'adressent à tout étudiant en mathématiques souhaitant mieux comprendre la logique qui se cache derrière les démonstrations. Elles peuvent être lues en parallèle d'un cours de mathématiques : les notions logiques prennent leur sens à travers les exemples mathématiques, tandis que la logique apporte un cadre plus précis aux raisonnements rencontrés. La troisième partie s'adresse au lecteur motivé qui souhaite aller plus loin dans l'étude de la logique moderne et comprendre la démonstration du théorème d'incomplétude de Gödel.

Les deux premières parties sont accompagnées d'exercices (non corrigés).
Prochainement disponibles : des vidéos et les fichiers sources.

Sommaire

Résumé des chapitres

Logique vrai/faux

La logique « Vrai/Faux », aussi appelée logique propositionnelle, s'attache à étudier quelle valeur, parmi « Vrai » ou « Faux », peut prendre une formule définie à partir des connecteurs logiques ET, OU, NON…

Preuves

Qu'est-ce qu'une preuve ? C'est une suite de déductions à partir d'hypothèses et d'un petit nombre de règles élémentaires, afin d'aboutir à une conclusion. Nous allons détailler ces règles et expliquer comment les utiliser en pratique.

Quels connecteurs sont utiles ?

Les connecteurs $\neg$ (NON), $\wedge$ (ET), $\vee$ (OU) suffisent pour construire toutes les formules de la logique Vrai/Faux.

Théorème de compacité

Le théorème de compacité affirme que lorsqu'il y a une infinité d'hypothèses, il est suffisant d'en utiliser un nombre fini.

Logique du premier ordre

On considère un langage avec des variables et les quantificateurs « pour tout » et « il existe ». On enrichit ce langage avec des prédicats et des fonctions.

Variables et structures

Notre but est d'utiliser un langage logique pour décrire des situations mathématiques concrètes. Pour cela, on introduit la notion de structure qui permet de relier les formules abstraites à des objets mathématiques réels. Avant de décider si, dans une structure donnée, une formule est vraie ou fausse, il est essentiel de distinguer les variables libres des variables liées.

Satisfaction

Pour attribuer une valeur Vrai/Faux à une formule de la logique du premier ordre, il faut à la fois définir une structure mathématique et affecter une valeur aux variables. On pourra alors définir la conséquence logique.

Preuves

Nous adaptons la construction des preuves aux quantificateurs de la logique du premier ordre.

Théorème de validité

Nous faisons le lien entre « preuve » et « vérité » grâce au théorème de validité (*Soundness Theorem*) qui affirme que toute formule prouvable est une formule vraie.

Théorème de complétude

Le théorème de complétude dit que, en logique du premier ordre, tout ce qui est vrai est prouvable. Nous énonçons ce théorème, ses variantes et ses applications.

Preuve du théorème de complétude

Nous prouvons le théorème de complétude.

Un aperçu de l'incomplétude de Gödel

Avant de se lancer dans la construction des outils nécessaires à l'énoncé précis des théorèmes de Gödel et à leur preuve, on donne un aperçu d'ensemble des théorèmes d'incomplétude et du long chemin pour y parvenir.

Arithmétique de Peano

Nous étudions les axiomes qui définissent l'arithmétique usuelle des entiers naturels.

Codage de Gödel

On transforme les symboles, les formules et même les preuves en des entiers. On expliquera ensuite les grandes lignes de la preuve du premier théorème d'incomplétude.

Fonctions récursives primitives

Nous définissons et étudions les fonctions qui expriment les calculs « faciles » de l'arithmétique.

Fonctions représentables

Il s'agit de justifier que l'arithmétique de Peano permet d'exprimer toutes les fonctions récursives primitives.

Diagonalisation

Nous rassemblons tous les outils construits à présent afin de démontrer l'étape cruciale de l'incomplétude : le lemme de diagonalisation.

Preuve du théorème d'incomplétude

Nous rassemblons tous les outils obtenus afin de prouver le premier théorème d'incomplétude de Gödel.

Le second théorème d'incomplétude

Nous clôturons ce livre avec le second théorème d'incomplétude de Gödel.

Première partie

logique vrai/faux

Chapitre 1

Logique vrai/faux

La logique « Vrai/Faux », aussi appelée logique propositionnelle, s'attache à étudier quelle valeur, parmi « Vrai » ou « Faux », peut prendre une formule définie à partir des connecteurs logiques ET, OU, NON...

1. Vrai/faux, connecteurs

1.1. Vrai/faux

Nous définissons deux valeurs V (Vrai) et F (Faux), appelées ***valeurs booléennes*** ou ***valeurs de vérité***. On utilisera aussi la notation **0** (Faux) et **1** (Vrai). La notation anglo-saxonne est T (*True*) et F (*False*).
Une ***variable propositionnelle*** est une variable qui peut prendre soit la valeur V (Vrai), soit la valeur F (Faux).
Par exemple, si P : « $2+2=4$ », alors P est vraie ; en revanche, Q : « $2^2<3$ » est fausse.

1.2. ¬ (NON), ∧ (ET), ∨ (OU)

À partir d'une ou plusieurs variables propositionnelles et de ***connecteurs***, on forme de nouvelles formules.
Soient P et Q deux variables propositionnelles.

- ***Négation*** $\neg P$. $\neg P$ est vraie si et seulement si P est fausse.
- ***Conjonction*** $P \wedge Q$. $P \wedge Q$ est vraie si et seulement si P et Q sont vraies.
- ***Disjonction*** $P \vee Q$. $P \vee Q$ est vraie si et seulement si P est vraie ou Q est vraie.

Reprenons P : « $2+2=4$ » et Q : « $2^2<3$ ». Alors :

- $\neg P$ est fausse (car P est vraie),
- $\neg Q$ est vraie (car Q est fausse),
- $P \wedge Q$ est fausse (car Q n'est pas vraie),
- $P \vee Q$ est vraie (car P est vraie).

Évidemment, il y a un problème de logique dans nos définitions, car par exemple nous avons défini le connecteur « $\wedge$ » (ET) à l'aide du mot français « et ». Pour remédier à ce problème, on définit l'action de chacun de ces connecteurs par sa ***table de vérité*** :

P	$\neg P$
V	F
F	V

P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
F	F	F

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

Bien sûr, à partir de ces connecteurs on peut composer des formules plus compliquées :

$$\neg(P \vee Q) \quad , \quad P \wedge (\neg P) \quad , \quad P \wedge (Q \vee R) \quad , \quad (P \wedge Q) \vee R \quad , \ldots$$

Les parenthèses sont importantes.

1.3. → (implication)

L'*implication* $P \to Q$ est définie par :

$$\boxed{P \to Q = (\neg P) \vee Q}$$

Comme le connecteur $\neg$ est prioritaire sur les autres, on écrit simplement : $\neg P \vee Q$.
On lit « si P alors Q » ou bien « P implique Q ». Sa table de vérité est :

P	Q	$P \to Q$
V	V	V
V	F	F
F	V	V
F	F	V

L'usage mathématique classique est de noter ce connecteur $P \implies Q$. Mais au fil du temps son usage a été dévoyé pour signifier « donc », ce qu'il n'est pas ! On y reviendra dans la Section 3.
Ainsi « $P \to Q$ » signifie « si P est vraie alors Q est vraie ». Voici quelques exemples :

- « il pleut » → « le sol est mouillé » est une implication vraie. Dans ce livre, on évitera les exemples en français pour limiter les formulations ambiguës et les discussions interminables du genre « pourquoi le sol de ma cuisine serait mouillé s'il pleut dehors ».
- « 2+2=4 » → « $3 \times 3 = 7$ » est une implication fausse.
- Soit n un entier. « n pair » → « n^2 pair » est vraie.

Passons aux aspects surprenants de l'implication.

- « 2+2=4 » → « un entier est pair ou impair » est une implication vraie (car si P et Q sont vraies alors $P \to Q$ est vraie), même s'il n'y a aucun lien de causalité entre P et Q.
- « $3 \times 3 = 7$ » → « $2+2=4$ » et « $3 \times 3 = 7$ » → « $2+2=5$ » sont aussi des implications vraies. En effet, dès que P est fausse, alors $P \to Q$ est vraie. C'est bien sûr clair par la définition de l'implication via $\neg P \vee Q$. Il faut accepter cette conséquence de la définition. Nous y reviendrons aussi dans la Section 3.

1.4. ↔ (équivalence)

L'*équivalence* $P \leftrightarrow Q$ est définie par la double implication $P \to Q$ et $Q \to P$:

$$\boxed{P \leftrightarrow Q = (P \to Q) \wedge (Q \to P)}$$

On dit « P équivaut à Q » ou bien « P si et seulement si Q ».
Autrement dit, $P \leftrightarrow Q$ est vraie lorsque P et Q sont vraies en même temps ou fausses en même temps.

P	Q	$P \leftrightarrow Q$
V	V	V
V	F	F
F	V	F
F	F	V

Voici des exemples d'équivalences vraies :

- $\neg\neg P \leftrightarrow P$: la négation d'une négation revient à la proposition initiale.
- $P \to Q \leftrightarrow (\neg P) \vee Q$: par définition de l'implication.

En revanche, l'équivalence suivante est *fausse* :

$$P \wedge Q \leftrightarrow P \vee Q$$

Plus précisément, cette équivalence n'est pas vraie pour toutes les valeurs attribuées à P et Q. En effet, l'implication $P \wedge Q \to P \vee Q$ est toujours vraie, mais l'implication $P \vee Q \to P \wedge Q$ est fausse lorsque P est vraie et Q est fausse par exemple.

1.5. Résumé

Il y a deux valeurs booléennes V/F (Vrai/Faux, ou T/F *True/False*, ou **1/0**). Nous avons défini les cinq connecteurs logiques de base.

Symbole	Nom	Définition
$\neg$	NON	$\neg P$ est vrai ssi P est faux
$\wedge$	ET	$P \wedge Q$ est vrai ssi P et Q sont vrais
$\vee$	OU	$P \vee Q$ est vrai ssi P ou Q est vrai
$\to$	implication	$\neg P \vee Q$
$\leftrightarrow$	équivalence	$(P \to Q) \wedge (Q \to P)$

Le connecteur $\neg$ est un connecteur ***unaire***, car il s'applique à un seul paramètre (comme une fonction d'une variable) alors que les autres sont des connecteurs ***binaires*** (par exemple on pourrait écrire $P \wedge Q$ sous la forme $\wedge(P,Q)$ comme une fonction de deux variables).

2. Formules

2.1. Objectifs

Nous disposons de ***variables propositionnelles*** P, Q, R... chacune pouvant prendre la valeur Vrai ou Faux. À l'aide des connecteurs, on compose une *formule* $\mathcal{P}$, par exemple :

$$\mathcal{P} : \quad P \wedge \big(Q \vee (\neg R)\big)$$

Voici les questions qu'on se pose. Si on remplace P, Q, R... par des valeurs V/F, est-ce que la formule $\mathcal{P}$ est vraie ou fausse ? On peut même être plus précis :

- Est-ce qu'il existe un choix de valeurs V/F qui rende l'évaluation de $\mathcal{P}$ vraie ?
- Pour tous les choix possibles de valeurs V/F, l'évaluation de $\mathcal{P}$ est-elle vraie ?

Par exemple, pour la proposition $\mathcal{P} : P \wedge \big(Q \vee (\neg R)\big)$, si on attribue la valeur V à P, F à Q et F à R, alors $\mathcal{P}$ s'évalue en $\mathrm{V} \wedge (\mathrm{F} \vee (\neg \mathrm{F}))$ qui vaut V. En revanche, si on remplace (P,Q,R) par $(\mathrm{V},\mathrm{F},\mathrm{V})$, alors $\mathcal{P}$ s'évalue en F.

Bien sûr, on pourrait se poser les mêmes questions pour savoir quand $\mathcal{P}$ s'évalue en faux, mais cela revient à décider quand la formule $\neg\mathcal{P}$ s'évalue en vrai.

Si on dispose de deux formules $\mathcal{P}$ et $\mathcal{Q}$, alors on peut composer des formules plus complexes $\mathcal{P} \wedge \mathcal{Q}$, $\mathcal{P} \vee \mathcal{Q}$, etc.

Lorsque l'on dispose de deux formules $\mathcal{P}$ et $\mathcal{Q}$, l'objectif le plus important sera de comprendre le lien entre $\mathcal{P}$ et $\mathcal{Q}$. Par exemple, est-ce que $\mathcal{P} \to \mathcal{Q}$ est vraie ? Est-ce que $\mathcal{P} \leftrightarrow \mathcal{Q}$ est vraie ?

2.2. Définition d'une formule

On considère un ***alphabet*** formé par :
- un ensemble Var $= \{P, Q, R, \ldots\}$ de variables propositionnelles,
- un ensemble Conn $= \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ de connecteurs logiques,
- de symboles de parenthèses (et).

Une ***formule*** est un mot fini, formé à partir des caractères de cet alphabet ; ce mot doit être correctement parenthésé et cohérent avec l'usage des connecteurs.

Définition.
Plus précisément, l'ensemble Form de toutes les ***formules*** est le plus petit ensemble de mots, contenant les variables propositionnelles, tel que :
- si $\mathcal{P}$ est une formule, alors $\neg\mathcal{P}$ aussi,
- si $\mathcal{P}$ et $\mathcal{Q}$ sont des formules, alors $(\mathcal{P} \wedge \mathcal{Q})$, $(\mathcal{P} \vee \mathcal{Q})$, $(\mathcal{P} \rightarrow \mathcal{Q})$, $(\mathcal{P} \leftrightarrow \mathcal{Q})$ sont aussi des formules.

Remarque : on notera $P, Q, R \ldots$ (caractère droit) pour les variables propositionnelles et $\mathcal{P}, \mathcal{Q}, \mathcal{R} \ldots$ (caractère calligraphique) pour les formules.
Voici des exemples de formules :

$$P \quad , \quad \neg P \quad , \quad (Q \rightarrow P) \quad , \quad ((Q \rightarrow P) \leftrightarrow Q) \quad , \quad (\neg R \rightarrow (P \vee Q))$$

Par abus de notation :
- On se permettra d'enlever les parenthèses quand elles sont totalement à l'extérieur, par exemple on écrira $Q \rightarrow P$ au lieu de $(Q \rightarrow P)$ ou lorsqu'il n'y a pas d'ambiguïté, par exemple $P \vee Q \vee R$.
- À l'inverse, on rajoutera parfois des parenthèses superflues. Par exemple, on pourra écrire $(\neg P) \wedge Q$ au lieu de $\neg P \wedge Q$ afin d'éviter la confusion avec $\neg(P \wedge Q)$.

2.3. Table de vérité, évaluation

Pour une formule $\mathcal{P}$ qui dépend de deux variables propositionnelles P et Q, on peut déterminer quand la formule prend la valeur Vrai ou Faux selon les valeurs attribuées à P et Q. Cela se résume sous la forme d'une ***table de vérité*** de la formule $\mathcal{P}$:

P	Q	$\mathcal{P}$
V	V	V/F ?
V	F	V/F ?
F	V	V/F ?
F	F	V/F ?

Plus généralement, si la formule $\mathcal{P}$ dépend des variables $P_1, P_2, \ldots, P_n$, une ***distribution de valeurs*** est un choix V/F pour la variable P_1, un choix V/F pour la variable P_2,... On obtient alors une ***évaluation*** V/F pour la formule $\mathcal{P}$.
Dans la pratique on dresse la table de vérité de la formule $\mathcal{P}$.

> **Exemple.**
> Soit :
>
> $$\mathcal{P} : \qquad (P \wedge Q) \vee (\neg P)$$
>
> On dresse la table de vérité de $\mathcal{P}$ à l'aide des tables intermédiaires de $P \wedge Q$ et $\neg P$.

P	Q	$P \wedge Q$	$\neg P$	$\mathcal{P}$
V	V	V	F	V
V	F	F	F	F
F	V	F	V	V
F	F	F	V	V

Dans les chapitres « Quels connecteurs sont utiles ? » et « Théorème de compacité », on présentera une distribution de valeurs comme une fonction $v : \{\mathrm{V}, \mathrm{F}\}^n \to \{\mathrm{V}, \mathrm{F}\}$ qui s'étend sur l'ensemble des formules afin d'obtenir une évaluation $v(\mathcal{P})$.

2.4. Équivalence logique

Définition.
Deux formules $\mathcal{P}$ et $\mathcal{Q}$ sont ***logiquement équivalentes*** si elles ont la même table de vérité. On note alors :

$$\mathcal{P} \equiv \mathcal{Q}$$

Cela signifie que les formules $\mathcal{P}$ et $\mathcal{Q}$ s'évaluent à Vrai en même temps et à Faux en même temps, quel que soit le choix des valeurs de vérité.
Il existe un lien direct avec le connecteur d'équivalence $\leftrightarrow$. En effet, $\mathcal{P} \equiv \mathcal{Q}$ signifie que $(\mathcal{P} \leftrightarrow \mathcal{Q})$ est une tautologie (c'est-à-dire toujours évaluée à Vrai), voir la Section 4.

Exemple.

1. $\neg\neg P \equiv P$ (double négation)
2. $\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q)$ (loi de De Morgan)
3. $(P \wedge Q) \to R \equiv \big(\neg(P \wedge Q)\big) \vee R \equiv \big((\neg P) \vee (\neg Q)\big) \vee R$

2.5. Équivalences logiques usuelles

idempotence	$\mathcal{P} \wedge \mathcal{P} \equiv \mathcal{P}$	$\mathcal{P} \vee \mathcal{P} \equiv \mathcal{P}$
commutativité	$\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P}$	$\mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P}$
associativité	$(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R})$	$(\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R})$
distributivité	$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$	$\mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R})$
absorption	$\mathcal{P} \wedge (\mathcal{P} \vee \mathcal{Q}) \equiv \mathcal{P}$	$\mathcal{P} \vee (\mathcal{P} \wedge \mathcal{Q}) \equiv \mathcal{P}$
lois de De Morgan	$\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv (\neg\mathcal{P}) \vee (\neg\mathcal{Q})$	$\neg(\mathcal{P} \vee \mathcal{Q}) \equiv (\neg\mathcal{P}) \wedge (\neg\mathcal{Q})$
contraposée	$\mathcal{P} \to \mathcal{Q} \equiv (\neg\mathcal{Q}) \to (\neg\mathcal{P})$	

Ce sont les plus classiques. Il en existe beaucoup d'autres !

3. Connecteurs (suite)

Nous apportons des précisions et des commentaires sur nos connecteurs favoris.

3.1. Vrai/Faux, négation

Il y a exactement deux valeurs booléennes $\{V, F\}$ (ou $\{\mathbf{0}, \mathbf{1}\}$), ni plus, ni moins ! Ce qui fait que si une variable P ou une formule $\mathcal{P}$ n'est pas évaluée à Faux, c'est qu'elle vaut Vrai. Et réciproquement. Il n'y a donc pas de réponse « peut-être ». Cela a les conséquences suivantes :

- ***Principe du tiers exclu*** :

$$\boxed{\mathcal{P} \vee \neg\mathcal{P} \quad \text{est toujours Vrai}}$$

  L'expression « tiers exclu » signifie qu'il n'y a pas d'autre choix que V/F.
- Principe de non-contradiction :

$$\mathcal{P} \wedge \neg\mathcal{P} \quad \text{est toujours Faux}$$

- Loi de la double négation :

$$\neg\neg\mathcal{P} \equiv \mathcal{P}$$

On verra un peu plus tard dans la Section 4 une nouvelle écriture de ce qui est vrai/faux :

$$\top \quad : \text{ formule qui s'évalue toujours à Vrai}$$

Ce symbole désigne une formule qui s'évalue toujours à V (T en anglais, d'où le symbole). Il se lit « toujours Vrai » ou « tautologie » ou *top*.
Sa contrepartie est le symbole renversé :

$$\bot \quad : \text{ formule qui s'évalue toujours à Faux}$$

Ainsi on peut écrire :

$$\mathcal{P} \vee \neg\mathcal{P} \equiv \top \qquad \mathcal{P} \wedge \neg\mathcal{P} \equiv \bot$$

Quelle est la différence entre V et $\top$? V est une valeur, alors que $\top$ est une formule qui s'évalue toujours à V. Si on identifie plutôt les valeurs booléennes à $\mathbf{0}/\mathbf{1}$, alors $\mathbf{1}$ est une constante, tandis que $\top$ est la fonction constante égale à $\mathbf{1}$.

3.2. $\wedge$ (ET), $\vee$ (OU)

Dans notre liste d'équivalences logiques, nous avions :

$$\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P} \qquad \mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P} \qquad \text{commutativité}$$

et aussi :

$$(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R}) \qquad (\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R}) \qquad \text{associativité}$$

Par conséquent, on s'autorise à écrire $\mathcal{P} \wedge \mathcal{Q} \wedge \mathcal{R}$ et $\mathcal{P} \vee \mathcal{Q} \vee \mathcal{R}$, sans parenthèses. Enfin :

$$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R}) \qquad \mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R}) \qquad \text{distributivité}$$

Le connecteur $\wedge$ (ET) est analogue à la multiplication $\times$, tandis que $\vee$ (OU) est analogue à l'addition $+$: $a \times b = b \times a$ (commutativité), $(a \times b) \times c = a \times (b \times c)$ (associativité), $a \times (b + c) = a \times b + a \times c$ (distributivité).
Les lois de De Morgan se résument par l'incantation « la négation d'un ET est un OU » et « la négation d'un OU est un ET » :

$$\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv \neg\mathcal{P} \vee \neg\mathcal{Q} \qquad \neg(\mathcal{P} \vee \mathcal{Q}) \equiv \neg\mathcal{P} \wedge \neg\mathcal{Q}$$

3.3. → (implication)

Par définition :

$$\mathcal{P} \to \mathcal{Q} \equiv \neg\mathcal{P} \vee \mathcal{Q}$$

et sa table de vérité est :

$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{P} \to \mathcal{Q}$
V	V	V
V	F	F
F	V	V
F	F	V

Il y a plusieurs points à intégrer concernant les implications.

Pas de causalité.
Il faut bien comprendre que $P \to Q$ (ici avec des variables propositionnelles) ne signale pas un lien de causalité mais plutôt une constatation : est-ce que si P est vraie alors Q est aussi vraie ?
Voici un exemple qui devrait faire hurler tous vos profs de maths.

> **Exemple.**
> Soit n un entier :
>
> $$n \text{ est impair} \to n \text{ est premier}$$
>
> Tous vos profs vous diront que cette implication est fausse, mais ce n'est pas vrai ! La confusion vient du fait qu'on a l'habitude de comprendre cette phrase par « tout nombre impair est aussi un nombre premier », ce qui est bien sûr faux. Mais ici n est fixé et sa valeur n'était pas précisée et donc l'implication peut être vraie ou fausse selon la valeur de n.
> - si $n = 3$, alors « n est impair $\to$ n est premier » est vraie, car P : « 3 est impair » et Q : « 3 est premier » sont vraies.
> - si $n = 15$, alors « n est impair $\to$ n est premier » est fausse, car P : « 15 est impair » est vraie alors que Q : « 15 est premier » est fausse.
> - si $n = 4$, alors « n est impair $\to$ n est premier » est vraie ! En effet P : « 4 est impair » est fausse et donc $P \to Q$ est vraie.

Lorsque P est faux alors $P \to Q$ est vrai !
C'est bien sûr clair d'après la définition $(\neg P) \vee Q$. Essayons de justifier pourquoi c'est naturel. Posons-nous la question : « quand est-ce qu'on veut que P implique Q soit faux ? ». La réponse est : « si P est vrai et Q est faux » (c'est comme avoir trouvé un contre-exemple). Dans tous les autres cas, $P \to Q$ est vrai, donc y compris lorsque P est faux.
Cela se retrouve dans le langage courant :

« Si tu as raison, alors moi je suis le pape. »

Sous-entendu, « Tu as tort. » Dans le même genre :

« Je le ferai quand les poules auront des dents. »

Sous-entendu, « Je ne le ferai pas. » C'est l'implication « poules avec dents $\to$ action » qui est toujours vraie avec ou sans action.

À propos des démonstrations.
De $P \to Q$, on ne peut pas déduire que Q est vrai. Ce que l'on verra dans la suite s'appelle le *modus ponens* et est la base de nos raisonnements :

Si P est vrai et si $P \to Q$ est vrai, alors Q est vrai.

C'est l'erreur que l'on fait à force d'habitude quand on écrit $P \implies Q$: on suppose implicitement que P est vrai et on en déduit que Q est vrai.

Enfin $\mathcal{P} \to \mathcal{Q} \to \mathcal{R}$ ne dénote pas une chaîne de conséquences, mais c'est par définition la formule :

$$\mathcal{P} \to (\mathcal{Q} \to \mathcal{R})$$

3.4. $\leftrightarrow$ (équivalence)

Pour qu'une équivalence soit vraie, il faut que $\mathcal{P}$ et $\mathcal{Q}$ soient vraies en même temps, et fausses en même temps. Par définition :

$$\mathcal{P} \leftrightarrow \mathcal{Q} \equiv (\mathcal{P} \to \mathcal{Q}) \wedge (\mathcal{Q} \to \mathcal{P})$$

On peut formuler les mêmes avertissements que pour l'implication : quand on écrit $\mathcal{P} \leftrightarrow \mathcal{Q}$ cela ne signifie pas que $\mathcal{P}$ et $\mathcal{Q}$ sont toujours évaluées à vrai, mais juste qu'elles s'évaluent à vrai en même temps.

3.5. D'autres connecteurs

Un ***connecteur à n places*** est une fonction $f : \{\mathrm{V},\mathrm{F}\}^n \to \{\mathrm{V},\mathrm{F}\}$, ou, si vous préférez, une fonction $f : \{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$.

- $n = 1$. $\neg$ est un connecteur unaire (c'est une fonction à une variable). On peut aussi avoir besoin de l'identité id qui envoie V sur V et F sur F.
- $n = 2$. $\wedge$ est un connecteur binaire (comme une fonction de deux variables). Sauf qu'on écrit $P \wedge Q$ au lieu de $\wedge(P,Q)$.
- $n = 2$. Le ***ou exclusif*** $\oplus$/XOR est un autre connecteur binaire :
  $P \oplus Q$ est vrai si exactement l'une des variables est vraie.
- $n = 2$. Le connecteur $\uparrow$/NAND, aussi appelé symbole de Sheffer, est défini comme la négation du ET :
  $$P \uparrow Q = \neg(P \wedge Q)$$
  On listera tous les connecteurs binaires dans le chapitre « Quels connecteurs sont utiles ? ».
- Pour n quelconque, on peut définir un connecteur à n places, en donnant la table de vérité en fonction des valeurs affectées aux variables $P_1, P_2, \ldots, P_n$. Cette table a 2^n lignes. On verra, toujours dans le chapitre « Quels connecteurs sont utiles ? », pourquoi on n'a pas vraiment besoin de ces connecteurs.

4. Formules (suite)

4.1. Formules satisfaisables

Soient $P_1, P_2, \ldots, P_n$ des variables propositionnelles. Une distribution de valeurs v associe à chaque variable P_i une valeur de vérité $v(P_i)$ qui est V ou F (ou une valeur $\mathbf{0}/\mathbf{1}$). On peut alors associer une valeur V/F, notée $v(\mathcal{P})$, à n'importe quelle formule $\mathcal{P}$.

> **Exemple.**
> On considère trois variables P, Q, R et la formule :
>
> $$\mathcal{P} : \quad (P \wedge \neg Q) \vee (Q \wedge R)$$
>
> Considérons la distribution de valeurs (V, F, F) (c'est-à-dire $v(P) = \mathrm{V}$, $v(Q) = \mathrm{F}$, $v(R) = \mathrm{F}$), alors, lorsque l'on substitue ces valeurs dans $\mathcal{P}$, on obtient :
>
> $$v(\mathcal{P}) \;=\; (\mathrm{V} \wedge \neg\mathrm{F}) \vee (\mathrm{F} \wedge \mathrm{F})$$
>
> et donc $v(\mathcal{P}) = \mathrm{V}$. Par contre, pour la distribution (V, V, F) alors $\mathcal{P}$ s'évalue à F.

Définition.

- Une formule $\mathcal{P}$ est ***satisfaite*** pour la distribution de valeurs v si $\mathcal{P}$ s'évalue à Vrai pour cette distribution.
- Une formule $\mathcal{P}$ est ***satisfaisable***, s'il existe une distribution de valeurs pour laquelle $\mathcal{P}$ s'évalue à Vrai.

La formule $\mathcal{P} : (P \wedge \neg Q) \vee (Q \wedge R)$ de l'exemple précédent est satisfaisable, car on avait trouvé une distribution de valeurs qui rend $\mathcal{P}$ vraie.

4.2. Tautologie

Définition.

- Une ***tautologie*** est une formule $\mathcal{P}$ qui s'évalue toujours à Vrai, quelle que soit la distribution des valeurs de vérité.
- Une ***contradiction*** est une formule $\mathcal{P}$ qui s'évalue toujours à Faux, quelle que soit la distribution des valeurs de vérité.

Notation :

tautologie : $\top$	contradiction : $\bot$

Quelques remarques :

- Ainsi une tautologie est une formule toujours vraie (plus précisément qui s'évalue toujours à la valeur Vrai). Le terme vient du grec *tautos* (le même) et *logos* (parole) : c'est toujours la même valeur (Vrai).
- Une contradiction est une formule toujours fausse.
- Cependant, la plupart des formules ne sont ni des tautologies, ni des contradictions : elles s'évaluent à des valeurs vraies et fausses selon le choix de la distribution de valeurs.
- Une formule $\mathcal{P}$ est une contradiction si et seulement si $\neg\mathcal{P}$ est une tautologie. C'est pourquoi on s'intéressera essentiellement aux tautologies.

> **Exemple.**
>
> 1. $P \vee \neg P$ est une tautologie (si P est Vrai, $P \vee \neg P$ est Vrai ; si P est Faux, $P \vee \neg P$ est aussi Vrai).
> 2. $P \wedge \neg P$ est une contradiction (c'est une formule toujours fausse, quelle que soit la valeur attribuée à P).
> 3. $(P \vee Q \vee R) \vee (\neg P \vee \neg Q \vee \neg R)$ est une tautologie.
> 4. $(P \wedge Q) \rightarrow Q$ est une tautologie.
> 5. $\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$ est une tautologie.
> 6. $\big((P \rightarrow Q) \wedge (Q \rightarrow R)\big) \rightarrow (P \rightarrow R)$ est une tautologie.

Voici des exemples d'utilisation des symboles $\top/\bot$:

$$\mathcal{P} \wedge \top \equiv \mathcal{P} \qquad \mathcal{P} \vee \top \equiv \top \qquad \mathcal{P} \wedge \bot \equiv \bot \qquad \mathcal{P} \vee \bot \equiv \mathcal{P}$$

Nous pouvons maintenant préciser le lien entre l'équivalence logique ($\equiv$) et l'équivalence ($\leftrightarrow$) :

$$\mathcal{P} \equiv \mathcal{Q} \quad \text{ssi} \quad \mathcal{P} \leftrightarrow \mathcal{Q} \text{ est une tautologie}$$

Si c'est le cas, c'est-à-dire si $\mathcal{P} \equiv \mathcal{Q}$ (ou bien si $\mathcal{P} \leftrightarrow \mathcal{Q}$ est toujours vraie), alors $\mathcal{P}$ et $\mathcal{Q}$ s'évaluent à vrai en même temps et à faux en même temps, quelle que soit l'évaluation.

Dans les sections suivantes, nous allons découvrir trois autres visions des formules et de leurs évaluations.

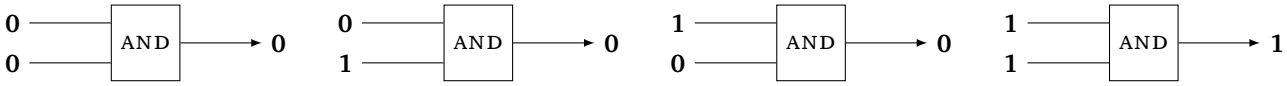
4.3. Portes logiques et circuits

Pour se rapprocher de ce qu'il se passe au cœur d'un ordinateur on peut considérer les connecteurs comme des portes logiques. Ici les valeurs de vérité sont $\{\mathbf{0}, \mathbf{1}\}$.

Porte NON/NOT.

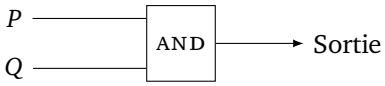
Elle a une seule entrée et change **0** en **1**, et **1** en **0**.

Porte ET/AND.

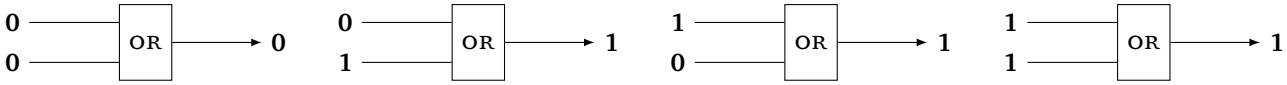


On résume l'action de la porte AND par le tableau suivant :

P	Q	Sortie
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**



Porte OU/OR.

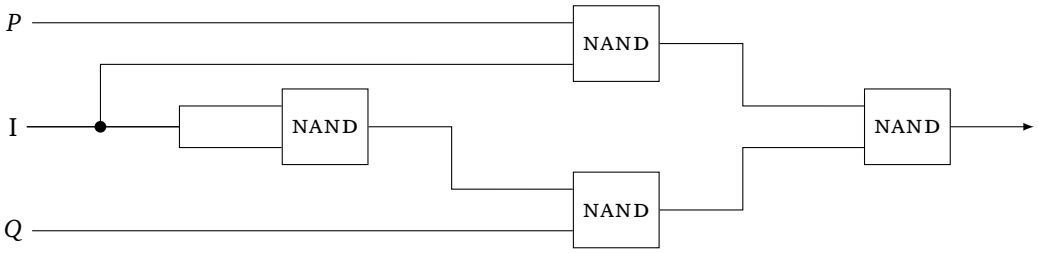


Porte NAND.

Elle peut se réaliser comme la porte NOT(AND), c'est-à-dire la porte AND suivie de la porte NOT.

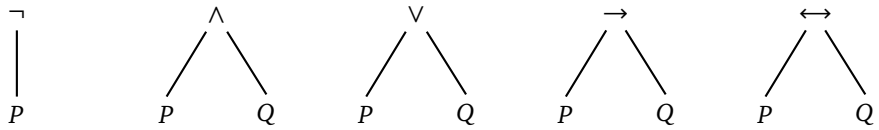
Mais nous préférons la voir comme une porte à part entière.

> **Exemple.**
> Un *multiplexeur* est un circuit qui aiguille un signal entre deux entrées P et Q à l'aide d'un sélecteur I. Si I vaut **1** alors la sortie du circuit est la valeur de P. Si I vaut **0** alors la sortie est la valeur de Q. Cela peut aussi être vu comme un test rudimentaire : « Si $I = \mathbf{1}$ faire ceci, sinon faire cela. »



4.4. Arbres

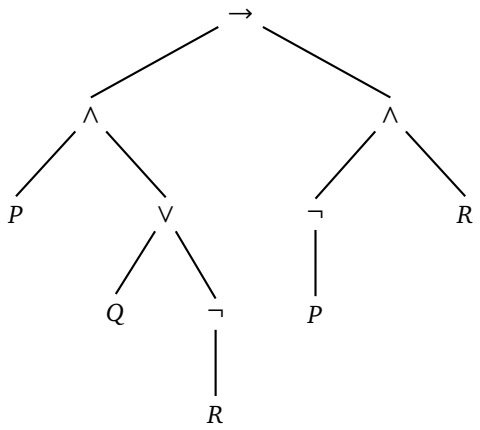
On peut aussi représenter les connecteurs par des ***arbres*** :



Exemple.
Considérons la formule :

$$\big(P \wedge (Q \vee \neg R)\big) \rightarrow (\neg P \wedge R)$$

L'arbre correspondant est :



L'intérêt est que la structure d'arbre est adaptée au traitement algorithmique. Un parcours préfixe en profondeur considère la racine, puis explore le sous-arbre de gauche, puis le sous-arbre de droite.

On peut aussi adopter la ***notation polonaise***, c'est-à-dire écrire $\wedge PQ$ (comme une fonction de deux variables $\wedge(P,Q)$) au lieu de $P \wedge Q$. En notation polonaise, la formule de l'exemple précédent s'écrit :

$$\rightarrow \wedge P \vee Q \neg R \wedge \neg P R$$

C'est une expression particulièrement adaptée au traitement par ordinateur, car elle ne nécessite pas de parenthèses. Pour un humain, c'est plus lisible de l'écrire ainsi :

$$\rightarrow \Big(\wedge \big(P\big)\big(\vee (Q)(\neg R)\big)\Big)\Big(\wedge (\neg P)(R)\Big)$$

4.5. Formules algébriques

On note $\mathbb{B} = \{\mathbf{0}, \mathbf{1}\}$ les valeurs booléennes. Dans cette section, on fait les calculs modulo 2, c'est-à-dire qu'on écrit $\mathbf{1} + \mathbf{1} = \mathbf{0}$ (en toute rigueur on devrait écrire $\mathbf{1} + \mathbf{1} \equiv \mathbf{0} \pmod 2$). Noter que modulo 2, on a $+\mathbf{1} = -\mathbf{1}$.
On note $x, y \in \{\mathbf{0}, \mathbf{1}\}$. Voici les expressions algébriques qui transposent l'action des connecteurs :

$$\begin{aligned} \neg x &= \mathbf{1} + x \\ x \wedge y &= x \times y \\ x \vee y &= x + y - xy \qquad \text{ou inclusif} \\ x \oplus y &= x + y \qquad \text{ou exclusif} \end{aligned}$$

Cela se justifie facilement, par exemple, si $x = \mathbf{0}$, alors $\mathbf{1} + x = \mathbf{1}$, et si $x = \mathbf{1}$ alors $\mathbf{1} + x = \mathbf{1} + \mathbf{1} = \mathbf{0}$. Donc $\mathbf{1} + x$ correspond bien à $\neg x$.
De même $x \times y$ ne peut valoir $\mathbf{1}$ que si $x = \mathbf{1}$ et $y = \mathbf{1}$.

> **Exemple.**
> Ces expressions algébriques peuvent fournir des preuves. Par exemple la double négation se justifie par :
> $$\neg\neg x = \neg(\mathbf{1} + x) = \mathbf{1} + (\mathbf{1} + x) = \mathbf{1} + \mathbf{1} + x = x$$
> Voici une preuve de la loi de De Morgan sur la négation d'un ET, $\neg(x \wedge y) = (\neg x) \vee (\neg y)$:
> $$(\neg x)\vee(\neg y) = (\mathbf{1}+x)\vee(\mathbf{1}+y) = (\mathbf{1}+x)+(\mathbf{1}+y)-(\mathbf{1}+x)(\mathbf{1}+y) = x+y-\mathbf{1}-x-y-xy = \mathbf{1}+xy = \neg(xy)$$
> On a utilisé $-\mathbf{1} = +\mathbf{1}$, donc $-xy = xy$.

Exercices

Vrai/faux, connecteurs

Exercice 1. Soit P : « $2 \times 2 = 5$ » et Q : « $4 + 4 > 7$ ». Dire si les propositions suivantes sont vraies ou fausses : $\neg P$, $\neg Q$, $P \wedge Q$, $P \vee Q$, $P \to Q$, $Q \to P$, $P \leftrightarrow Q$.

Exercice 2. On suppose P vraie. Pour quelles valeurs de Q les propositions suivantes sont-elles vraies : $P \wedge Q$, $P \vee Q$, $P \to Q$, $Q \to P$, $P \leftrightarrow Q$? Même question si cette fois on suppose P fausse.

Formules

Exercice 3. Parmi les mots suivants lesquels sont des formules (au sens strict de la définition) :

$$\neg\neg P \qquad (\neg P \vee Q \wedge R) \qquad (P \to (Q \wedge (R \wedge S))) \qquad (P, Q \to R) \qquad ((P \wedge \neg Q \to)(Q \vee R))$$

Exercice 4. Dresser la table de vérité pour $\neg(P \vee Q)$ et celle pour $(\neg P) \wedge (\neg Q)$. En déduire la loi de De Morgan pour $\vee$.

Exercice 5. Prouver les formules d'associativité et de distributivité suivantes :

$$P \vee (Q \vee R) \equiv (P \vee Q) \vee R \qquad P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

Exercice 6. Montrer les équivalences logiques suivantes :

$$P \to Q \wedge R \equiv (P \to Q) \wedge (P \to R) \qquad P \vee Q \to R \equiv (P \to R) \wedge (Q \to R) \qquad P \leftrightarrow Q \equiv (P \vee Q) \to (P \wedge Q)$$

Traduire ces équivalences par une phrase en français.

Connecteurs (suite)

Exercice 7. Montrer que les formules suivantes sont des tautologies (elles sont toujours vraies, quelles que soient les valeurs de P et Q) :

$$P \vee (P \to Q) \qquad (P \wedge Q) \to P \qquad (P \to Q) \vee (P \to \neg Q) \qquad (P \to Q) \vee (\neg P \to Q)$$

Exercice 8. Pour chacune des formules suivantes, trouver une formule d'expression la plus simple possible qui lui est logiquement équivalente :

$$Q \vee (P \wedge Q) \qquad P \wedge (P \vee Q) \qquad P \wedge (\neg P \vee Q) \qquad (P \vee Q) \to (P \wedge Q)$$

Exercice 9. Soit $x \in \mathbb{R}$, P : « $x \geqslant 2$ » et Q : « $x^2 \geqslant 4$ ». Trouver toutes les valeurs $x \in \mathbb{R}$, telles que $P \to Q$ soit vraie. Même question avec $Q \to P$, puis $P \leftrightarrow Q$.

Exercice 10.

1. Déterminer la table de vérité de $P \to (Q \to R)$, puis celle de $(P \to Q) \to R$. Conclusion ?
2. Montrer que $P \leftrightarrow (Q \leftrightarrow R) \equiv (P \leftrightarrow Q) \leftrightarrow R$.
3. Montrer que $P \to (Q \to R) \equiv (P \wedge Q) \to R \equiv Q \to (P \to R)$.

Exercice 11. On rappelle que $P \oplus Q$ désigne le « ou exclusif » (c'est vrai lorsque P est vrai, ou Q est vrai, mais pas les deux).

1. Calculer la table de vérité de $P \oplus Q$.
2. Montrer que $P \oplus Q$ revient à « $P \neq Q$ », c'est-à-dire qu'on a $P \oplus Q \equiv (P \wedge \neg Q) \vee (\neg P \wedge Q)$. Quel est le lien entre $\oplus$ et $\leftrightarrow$?
3. Montrer que $\oplus$ est commutative et associative.
4. Exprimer $P \wedge (Q \oplus R)$ uniquement avec les connecteurs $\wedge$, $\vee$ et $\neg$. Idem avec $P \oplus (Q \wedge R)$.

Exercice 12. Soit $P_1, P_2, \ldots, P_n$, avec $n \geqslant 1$. Quand est-ce que $P_1 \oplus P_2 \oplus \cdots \oplus P_n$ est vraie ?

Formules (suite)

Exercice 13. Les formules suivantes sont-elles satisfaisables ? Sont-elles des tautologies ?

$$P \wedge Q \to P \qquad (P \vee Q) \wedge (\neg P \vee \neg Q) \qquad P \vee (Q \wedge \neg Q) \leftrightarrow P \qquad (P \to Q) \wedge (Q \to \neg P)$$
$$\big((P \to Q) \wedge (R \to S)\big) \to \big((P \vee R) \to (Q \vee S)\big)$$

Exercice 14. Montrer que les formules suivantes sont des tautologies :

1. $P \wedge (P \to Q) \to Q$ (*modus ponens*)
2. $(P \to Q) \wedge \neg Q \to \neg P$ (*modus tollens*)
3. $(P \to Q) \leftrightarrow (\neg Q \to \neg P)$ (contraposition)
4. $(P \to Q) \leftrightarrow \big((P \wedge \neg Q) \to \bot\big)$ (raisonnement par l'absurde)
5. $\big((P \to Q) \wedge (Q \to R)\big) \to (P \to R)$ (syllogisme/transitivité de l'implication).
6. $(P \wedge \top) \leftrightarrow P$

Exercice 15.

1. Construire le circuit pour $P \wedge (Q \vee R)$ et celui pour $(P \wedge Q) \vee (P \wedge R)$. Vérifier qu'ils produisent les mêmes sorties.
2. Montrer qu'en utilisant uniquement des portes NAND, on peut construire des circuits dont les sorties sont identiques à l'action de $\neg$, $\wedge$, $\vee$.

Exercice 16. Pour chacune des formules suivantes, construire l'arbre associé et donner l'expression selon la notation polonaise :

$$\neg P \wedge \big(Q \vee (P \wedge R)\big) \qquad \big((P \vee Q) \wedge \neg R\big) \to \big(P \wedge (\neg Q \wedge R)\big)$$

Exercice 17.

1. Justifier la formule algébrique $x \vee y = x + y - xy$.
2. À l'aide des formules algébriques, montrer :
$$x \wedge (y \wedge z) = (x \wedge y) \wedge z \qquad x \vee (y \vee z) = (x \vee y) \vee z \qquad x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$
3. Calculer $x \wedge (y \oplus z)$ et $(x \wedge y) \oplus (x \wedge z)$. Conclusion ?
4. Trouver une formule algébrique pour $x \to y$. Idem pour $x \leftrightarrow y$.
5. À l'aide des formules algébriques, montrer $x \to y = \neg y \to \neg x$.

Chapitre 2

Preuves

Qu'est-ce qu'une preuve ? C'est une suite de déductions à partir d'hypothèses et d'un petit nombre de règles élémentaires, afin d'aboutir à une conclusion. Nous allons détailler ces règles et expliquer comment les utiliser en pratique.

1. Preuve par table de vérité

1.1. Hypothèses & conclusion

On dispose de formules $\mathcal{P}_1$, …, $\mathcal{P}_l$ qui vont être nos ***hypothèses*** (ou ***prémisses***) et d'une formule $\mathcal{Q}$ qui va être notre ***conclusion***. Notre but est bien sûr de passer d'hypothèses vraies à une conclusion vraie. Il va y avoir deux façons de faire et c'est vraiment le cœur de la théorie de Gödel.

Remarque. S'il y a un nombre fini d'hypothèses $\mathcal{P}_1$, …, $\mathcal{P}_l$, on pourrait toujours se ramener à la seule hypothèse $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_l$; dans la pratique on ne le fera pas. On pourrait aussi vouloir plusieurs formules comme conclusion, mais on ne le fera pas non plus car cela reviendrait à les étudier une par une.

1.2. Conséquence logique $\vDash$

Commençons par la notion de *conséquence logique*, aussi nommée *conséquence sémantique*.

Définition.
On dit que $\mathcal{Q}$ est une ***conséquence logique*** de $\mathcal{P}_1$, …, $\mathcal{P}_l$ si chaque évaluation rendant les formules $\mathcal{P}_1$, …, $\mathcal{P}_l$ vraies, rend également la formule $\mathcal{Q}$ vraie. On note :

$$\boxed{\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}}$$

Le symbole $\vDash$ est le *double turnstile*, il se lit *entraîne*. Autrement dit, si les hypothèses sont vérifiées, alors la conclusion est vraie. (Si l'une des hypothèses $\mathcal{P}_i$ est fausse, alors on n'exige rien sur $\mathcal{Q}$ qui peut être vraie ou fausse.)

1.3. Preuve par table de vérité

Obtenir une conséquence logique

$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}$$

relève de la constatation. On dresse la table de vérité associée à toutes les variables propositionnelles $P_1, \ldots, P_n$ intervenant dans les formules $\mathcal{P}_1, \ldots, \mathcal{P}_l$ et $\mathcal{Q}$. Pour chaque ligne où toutes les formules $\mathcal{P}_1, \ldots, \mathcal{P}_l$ s'évaluent à Vrai, alors $\mathcal{Q}$ doit aussi s'évaluer à Vrai.

1.4. Exemples

Exemple.

$$P \vee Q,\ \neg Q \quad \vDash \quad P$$

On a ici :

- deux variables propositionnelles P et Q,
- deux hypothèses : $\mathcal{P}_1$: « $P \vee Q$ » et $\mathcal{P}_2$: « $\neg Q$ »,
- une conclusion : « P ».

Voici la table de vérité :

P	Q	$P \vee Q$	$\neg Q$	P
V	V	V	F	—
V	F	V	V	V
F	V	V	F	—
F	F	F	V	—

Seule la seconde ligne donne les deux hypothèses vraies et on contrôle alors que la conclusion est aussi vraie. On n'a pas besoin de savoir si la conclusion est vraie ou fausse dans les autres cas. Ainsi on a bien la conséquence logique : $P \vee Q, \neg Q \vDash P$.

Insistons sur le fait que lorsque l'on écrit :

$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}$$

on n'exige pas que les formules $\mathcal{P}_1, \ldots, \mathcal{P}_l$ s'évaluent toujours à Vrai. Ce que l'on veut c'est que *lorsque* elles s'évaluent à Vrai, alors $\mathcal{Q}$ aussi. La situation ultime est par exemple :

$$P,\ \neg P \quad \vDash \quad \text{n'importe quoi}$$

Cette conséquence logique est valide, quelle que soit la conclusion, car P et $\neg P$ ne s'évaluent jamais à vrai en même temps.

Exemple.

$$P \to Q,\ Q \to R \quad \vDash \quad P \to R$$

On a ici trois variables propositionnelles P, Q, R, deux hypothèses et une conclusion.

P	Q	R	$P \to Q$	$Q \to R$	$P \to R$
V	V	V	V	V	V
V	V	F	V	F	—
V	F	V	F	V	—
V	F	F	F	V	—
F	V	V	V	V	V
F	V	F	V	F	—
F	F	V	V	V	V
F	F	F	V	V	V

On regarde seulement les lignes où les deux hypothèses sont vérifiées et on contrôle alors que la conclusion est aussi vraie. On n'a pas besoin de savoir si la conclusion est vraie ou fausse dans les autres cas. Ainsi on a bien la conséquence logique voulue.

Dans l'exemple suivant on n'a pas une conséquence logique !

Exemple.
Soient les hypothèses :

$$\neg P,\ P \vee Q$$

Est-ce que cela entraîne :

$$P \wedge Q \ ?$$

On dresse la table de vérité :

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$
⋮	⋮			
F	V	V	V	F
⋮	⋮			

On s'aperçoit qu'une ligne est problématique : en effet si P est faux et Q est vrai, alors les deux hypothèses sont vraies et pourtant la conclusion souhaitée est fausse. Ainsi $P \wedge Q$ n'est pas une conséquence logique des hypothèses $\neg P$ et $P \vee Q$.

1.5. Limitations

On vient de voir que si on a n variables $P_1, \ldots, P_n$ dans nos hypothèses $\mathcal{P}_1, \ldots, \mathcal{P}_l$ et la conclusion $\mathcal{Q}$, alors il faut dresser une table de vérité avec 2^n lignes afin de vérifier la conséquence logique.
Lorsque n croît, alors 2^n croît de façon exponentielle (car $2^n = e^{n \ln 2}$) et il devient impraticable de calculer à la main une telle table. Mais surtout nous sommes encore dans le cadre assez limité de la logique Vrai/Faux. Cette technique de table de vérité ne sera plus possible lorsque l'on introduira les paramètres de la logique du premier ordre. Considérons le problème suivant où $n \in \mathbb{N}$ est un entier. Nous serons tous d'accord pour dire que l'on a la conséquence logique suivante :

$$n > 0,\ n \text{ pair} \quad \models \quad n^2 - 1 \text{ entier naturel impair}$$

Mais si on souhaite le justifier avec une table de vérité, celle-ci a une infinité de lignes dépendant du paramètre n, et cela ne peut pas être accepté comme une preuve.

n	$n > 0$	n pair	$n^2 - 1$ entier naturel impair
0	F	V	–
1	V	F	–
2	V	V	V
3	V	F	–
4	V	V	V
⋮	⋮	⋮	⋮

2. Preuve formelle

2.1. Qu'est-ce qu'une preuve ?

Une preuve a pour but de convaincre les autres qu'un énoncé est vrai. Une preuve doit impérativement comporter un nombre fini d'étapes. Chacun doit pouvoir vérifier chacune de ces étapes et le passage d'une étape à une autre doit être facile à comprendre et ne doit pas nécessiter une réflexion approfondie. Il y a une similarité naturelle entre la notion d'algorithme (une suite d'instructions élémentaires, destinées à être lues par une machine) et celle de preuve (destinée à un humain). Avec l'arrivée des assistants de preuve, cette différence s'amenuise.

2.2. Preuve formelle $\vdash$

Soient $\mathcal{P}_1, \ldots, \mathcal{P}_l$ des formules, qui seront nos hypothèses, et une formule $\mathcal{Q}$ (la conclusion). On note :

$$\boxed{\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q}}$$

s'il existe une *preuve* de $\mathcal{Q}$ à partir de $\mathcal{P}_1, \ldots, \mathcal{P}_l$. Cette écriture s'appelle alors un ***théorème***. Le symbole $\vdash$ est le *turnstile* et se lit « prouve » ou « permet de démontrer ».
Mais comment construire une preuve ? C'est une suite finie de formules, qui part des hypothèses et conduit à la conclusion. Chaque ligne de la preuve est obtenue à partir d'un petit nombre de règles logiques autorisées, appliquées aux lignes précédentes.
Plus précisément :

Définition.
Une ***preuve*** est une suite finie de formules $\mathcal{A}_1$, $\mathcal{A}_2$, …, $\mathcal{A}_N$ où chaque $\mathcal{A}_i$ est :

- soit l'une des hypothèses $\mathcal{P}_1, \ldots, \mathcal{P}_l$ de l'énoncé,
- soit une conséquence obtenue à partir des *règles d'inférence* appliquées aux lignes précédentes $\mathcal{A}_1$,…, $\mathcal{A}_{i-1}$,
- et $\mathcal{A}_N$ est la conclusion $\mathcal{Q}$.

Les règles d'inférence seront détaillées un peu plus loin. Par exemple cela peut être une règle issue des formules logiques élémentaires telles que :

- si $\mathcal{P} \wedge \mathcal{Q}$ alors $\mathcal{P}$,
- si $\mathcal{P}$ et $\mathcal{Q}$ alors $\mathcal{P} \wedge \mathcal{Q}$,
- …

Résumons le travail d'un mathématicien. Il énonce un théorème :
Théorème.

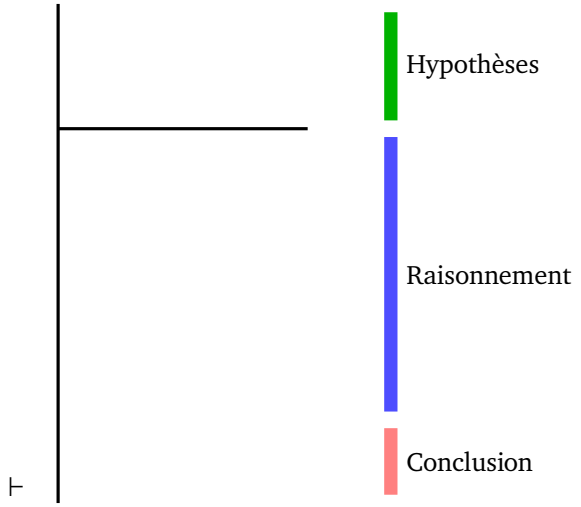
$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q}$$

Puis il en donne la preuve.

- Remarque. « Théorème » signifie ici un énoncé prouvé. On ne fait pas de distinction qualitative entre Lemme/Proposition/Théorème/Corollaire selon leur importance ou leur intérêt.
- Avertissement. À priori, aucune notion de vrai ou de faux n'intervient explicitement dans cette section consacrée aux preuves. Le lien avec la logique Vrai/Faux est en réalité implicite : il réside dans le choix des règles d'inférence, qui ont été conçues pour respecter cette logique.

2.3. Un premier exemple

Une ***preuve en déduction naturelle*** se présente comme un tableau contenant une succession de lignes : d'abord les hypothèses, les raisonnements et la conclusion.



Avant de dresser la liste des règles d'inférence, voici un exemple pratique d'un énoncé et de sa preuve.
Théorème.

$$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \quad \vdash \quad (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$$

Preuve du théorème.

	$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R})$	Hypothèse
	$\mathcal{P}$	Terme de gauche de l'hypothèse
	$\mathcal{Q} \vee \mathcal{R}$	Terme de droite de l'hypothèse
	$\quad \mathcal{Q}$	Sous-hypothèse. Premier cas du OU
	$\quad \mathcal{P} \wedge \mathcal{Q}$	Car on a déjà $\mathcal{P}$ et ici on a $\mathcal{Q}$
	$\quad (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$	Car le terme de gauche est vrai
	$\quad \mathcal{R}$	Sous-hypothèse. Second cas du OU
	$\quad \mathcal{P} \wedge \mathcal{R}$	Car on a déjà $\mathcal{P}$ et ici on a $\mathcal{R}$
	$\quad (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$	Car le terme de droite est vrai
$\vdash$	$(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$	Conclusion. Les deux cas mènent au même résultat.

Les règles qu'on a utilisées sont celles d'introduction et d'élimination du ET et du OU :

$$\mathcal{P} \wedge \mathcal{Q} \quad \text{signifie} \quad \mathcal{P} \text{ et } \mathcal{Q}$$
$$\mathcal{P} \vee \mathcal{Q} \quad \text{signifie} \quad \mathcal{P} \text{ ou } \mathcal{Q}$$

3. Règles d'inférence

Nous donnons ici la liste des règles autorisées dans une preuve. Certaines sont naturelles, d'autres un peu moins évidentes. On commence par une série de six *règles de base* qui constituent le fondement des preuves en logique classique. On présentera ensuite des *règles dérivées*, qui découlent des règles de base et facilitent l'écriture des preuves.

3.1. Les règles de base

Règle 1. Réutilisation.

$$\boxed{\mathcal{P} \vdash \mathcal{P}}$$

La réutilisation est tout simplement le fait de pouvoir réutiliser une formule déjà obtenue précédemment.

Règle 2. Conjonction.

$$\boxed{\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P} \qquad \text{et} \qquad \mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}}$$

$$\boxed{\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}}$$

Les règles pour le ET sont naturelles, par exemple si on a déjà obtenu la formule $\mathcal{P}$ et aussi la formule $\mathcal{Q}$ alors on a prouvé la formule $\mathcal{P} \wedge \mathcal{Q}$.

Règle 3. Double négation.

$$\boxed{\neg\neg\mathcal{P} \vdash \mathcal{P}}$$

Cette règle sera discutée dans une remarque plus bas.
Voici comment on écrit ces règles dans une preuve en déduction naturelle :

$$\left|\begin{array}{l}\vdots\\ \mathcal{P}\\ \vdots\\ \mathcal{P}\\ \vdots\end{array}\right. \qquad \vdash \left|\begin{array}{l}\vdots\\ \mathcal{P} \wedge \mathcal{Q}\\ \vdots\\ \mathcal{P}\end{array}\right. \qquad \left|\begin{array}{l}\vdots\\ \mathcal{P}\\ \vdots\\ \mathcal{Q}\\ \vdots\end{array}\right. \vdash \mathcal{P} \wedge \mathcal{Q} \qquad \left|\begin{array}{l}\vdots\\ \neg\neg\mathcal{P}\\ \vdots\\ \mathcal{P}\end{array}\right. \vdash$$

Dans l'écriture d'une preuve, on peut utiliser toutes les formules obtenues auparavant. On peut bien sûr appliquer plusieurs règles successivement avant d'obtenir la conclusion finale souhaitée, mais une ligne correspond à l'application d'une seule règle.

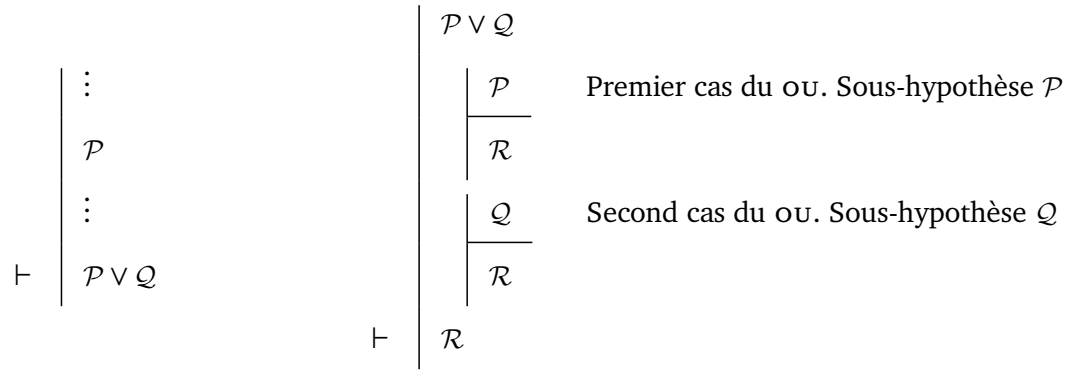
Règle 4. Disjonction.

$$\boxed{\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q} \qquad \text{et} \qquad \mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}}$$

Il est plus difficile d'écrire formellement l'autre sens : « $\mathcal{P} \vee \mathcal{Q}$ prouve $\mathcal{P}$ ou $\mathcal{Q}$ ». Pour l'exprimer correctement, on fait intervenir une formule annexe $\mathcal{R}$ comme conclusion d'implications :

$$\boxed{\mathcal{P} \vee \mathcal{Q},\ \mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{R} \quad \vdash \quad \mathcal{R}}$$

Dans la pratique, c'est assez facile (tableau de droite ci-dessous) : si on a $\mathcal{P} \vee \mathcal{Q}$ alors on considère deux cas. Un premier cas, avec comme hypothèse $\mathcal{P}$, puis un second cas, avec comme hypothèse $\mathcal{Q}$. Si on prouve la même conclusion $\mathcal{R}$ dans ces deux cas, alors $\mathcal{R}$ est prouvé.



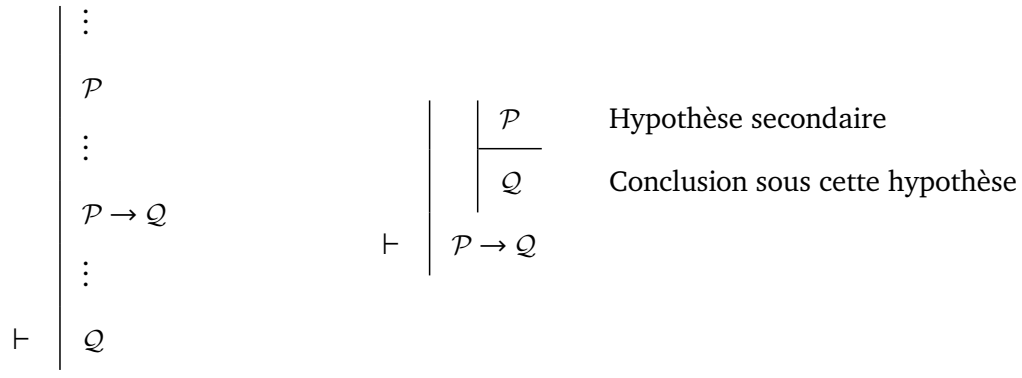
Règle 5. Implication.

$$\boxed{\mathcal{P},\ \mathcal{P} \to \mathcal{Q} \quad \vdash \quad \mathcal{Q}}$$

Cette règle s'appelle le *modus ponens*. C'est une sorte de mini-preuve de base : si on a une hypothèse $\mathcal{P}$ et que l'on sait que $\mathcal{P}$ implique $\mathcal{Q}$, alors on a prouvé $\mathcal{Q}$!
Obtenir une réciproque, c'est-à-dire l'introduction de l'implication, est plus difficile à exprimer :

$$\boxed{\text{Si} \quad \mathcal{P} \vdash \mathcal{Q} \quad \text{alors} \quad \vdash \mathcal{P} \to \mathcal{Q}.}$$

Si l'on peut prouver $\mathcal{Q}$ sous l'hypothèse $\mathcal{P}$, alors on a une preuve de $\mathcal{P} \to \mathcal{Q}$.
Heureusement, cette seconde règle est de nouveau facile à comprendre et à utiliser avec un tableau de preuve (figure de droite ci-dessous).



Règle 6. Explosion.

$$\boxed{\bot \quad \vdash \quad \mathcal{P}}$$

La règle d'explosion (*ex falso*) dit que : « Faux implique n'importe quoi. » Cela peut sembler un peu étrange, mais ce sera utile par exemple dans les raisonnements par l'absurde.

Remarque.

- On énoncera d'autres règles un peu plus loin, mais ces règles pourront se déduire des règles de base.
- Pour chacun des connecteurs $\wedge$, $\vee$, $\to$, il y a deux règles associées : une règle d'***élimination*** qui fait disparaître le symbole (par exemple $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$) et une règle d'***introduction*** qui fait apparaître le symbole (par exemple $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$).
- Les règles de base que l'on vient d'énoncer correspondent à ce qu'on nomme la ***logique classique***. Il s'agit de choix qui correspondent à notre expérience mathématique, ce sont des choix conventionnels, mais d'autres approches sont possibles.
- La ***logique minimaliste*** refuse les règles $\neg\neg\mathcal{P} \vdash \mathcal{P}$ et $\bot \vdash \mathcal{P}$. La ***logique intuitionniste*** ne rejette que $\neg\neg\mathcal{P} \vdash \mathcal{P}$.
- Quelles sont les raisons de rejeter l'élimination de la double négation ? Cette règle est en fait équivalente à la règle du tiers exclu qui dit qu'on a toujours $\mathcal{P} \vee \neg\mathcal{P}$. Lorsque l'on intègre la règle $\neg\neg\mathcal{P} \vdash \mathcal{P}$, on autorise le type de preuve suivant : « Si $\neg\mathcal{P}$ ne peut pas se produire, alors on a une preuve de $\mathcal{P}$. » Les intuitionnistes contestent ce type de raisonnement car cela ne fournit pas une preuve constructive.
- La règle d'explosion $\bot \vdash \mathcal{P}$ est exclue dans certains types de logique, car d'une part cette règle ne donne pas une preuve constructive de $\mathcal{P}$ et d'autre part, dès que l'on trouve une contradiction (même locale), alors la règle permet de prouver n'importe quelle formule, ce qui fait que tout raisonnement devient trivial, donc inutilisable.

3.2. Exemples

Exemple.

$$(\mathcal{P} \wedge \mathcal{Q}) \vee \mathcal{R} \quad \vdash \quad \mathcal{Q} \vee \mathcal{R}$$

	$(\mathcal{P} \wedge \mathcal{Q}) \vee \mathcal{R}$	Hypothèse
	$\mathcal{P} \wedge \mathcal{Q}$	Premier cas du OU
	$\mathcal{Q}$	Par élimination dans $\mathcal{P} \wedge \mathcal{Q}$
	$\mathcal{Q} \vee \mathcal{R}$	Puisqu'on a $\mathcal{Q}$
	$\mathcal{R}$	Second cas du OU
	$\mathcal{Q} \vee \mathcal{R}$	Puisqu'on a $\mathcal{R}$
$\vdash$	$\mathcal{Q} \vee \mathcal{R}$	Les deux cas donnent $\mathcal{Q} \vee \mathcal{R}$

Exemple.

$$\mathcal{P} \vee \mathcal{Q},\ \mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{S} \quad \vdash \quad \mathcal{R} \vee \mathcal{S}$$

	$\mathcal{P} \vee \mathcal{Q}$	Hypothèses
	$\mathcal{P} \to \mathcal{R}$	
	$\mathcal{Q} \to \mathcal{S}$	
	| $\mathcal{P}$	Premier cas du OU
	| $\mathcal{R}$	On applique $\mathcal{P} \to \mathcal{R}$
	| $\mathcal{R} \vee \mathcal{S}$	
	| $\mathcal{Q}$	Second cas du OU
	| $\mathcal{S}$	
	| $\mathcal{R} \vee \mathcal{S}$	
$\vdash$	$\mathcal{R} \vee \mathcal{S}$	Les deux cas conduisent à la même conclusion

Exemple.

$$\mathcal{P} \to \mathcal{Q},\ \mathcal{P} \to \mathcal{R} \quad \vdash \quad \mathcal{P} \to (\mathcal{Q} \wedge \mathcal{R})$$

	$\mathcal{P} \to \mathcal{Q}$	Hypothèses
	$\mathcal{P} \to \mathcal{R}$	
	| $\mathcal{P}$	Sous-hypothèse
	| $\mathcal{Q}$	Puisqu'on a $\mathcal{P}$ et $\mathcal{P} \to \mathcal{Q}$
	| $\mathcal{R}$	Puisqu'on a $\mathcal{P}$ et $\mathcal{P} \to \mathcal{R}$
	| $\mathcal{Q} \wedge \mathcal{R}$	
$\vdash$	$\mathcal{P} \to (\mathcal{Q} \wedge \mathcal{R})$	Car chaque fois qu'on a $\mathcal{P}$ on a aussi $\mathcal{Q} \wedge \mathcal{R}$

Exemple.

$$\mathcal{P} \vee \mathcal{Q},\ \neg\mathcal{P} \quad \vdash \quad \mathcal{Q}$$

C'est un exemple d'utilisation de l'explosion : une fois qu'on a prouvé $\bot$, on prouve n'importe quoi.

	$\mathcal{P} \vee \mathcal{Q}$	
	$\neg\mathcal{P}$	
	$\mathcal{P}$	Premier cas du OU
	$\bot$	Contradiction car on a $\mathcal{P}$ et $\neg\mathcal{P}$
	$\mathcal{Q}$	Tout est vrai par explosion
	$\mathcal{Q}$	Second cas du OU
	$\mathcal{Q}$	
$\vdash$	$\mathcal{Q}$	Conclusion issue des deux cas

3.3. Règles dérivées

À partir des règles d'inférence de base, on en déduit un nombre limité d'autres règles. Ces nouvelles règles permettent de simplifier l'écriture des preuves.

Règles dérivées. Équivalence.
Comme on s'y attend, on prouve une équivalence par double implication et réciproquement.

$$\boxed{\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{P} \to \mathcal{Q} \quad \text{et} \quad \mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{Q} \to \mathcal{P}}$$

$$\boxed{\mathcal{P} \to \mathcal{Q},\ \mathcal{Q} \to \mathcal{P} \vdash \mathcal{P} \leftrightarrow \mathcal{Q}}$$

Règles dérivées. Négation.
Tout d'abord, voici la définition de la négation à partir de l'implication et la formule toujours fausse $\bot$.

$$\neg\mathcal{P} \text{ est défini par } \mathcal{P} \to \bot$$

On a donc, par définition les deux règles :

$$\boxed{\neg\mathcal{P} \vdash \mathcal{P} \to \bot \quad \text{et} \quad \mathcal{P} \to \bot \vdash \neg\mathcal{P}}$$

Ainsi, si on a $\mathcal{P}$ et $\neg\mathcal{P}$ alors, à l'aide de l'implication, on obtient une contradiction :

$$\boxed{\mathcal{P}, \neg\mathcal{P} \vdash \bot}$$

Bien sûr $\neg\top$ est $\bot$, et réciproquement $\neg\bot$ est $\top$.
On dispose également de l'autre sens de la règle de double négation :

$$\boxed{\mathcal{P} \vdash \neg\neg\mathcal{P}}$$

On a aussi le principe du tiers exclu :

$$\boxed{\vdash \mathcal{P} \vee \neg\mathcal{P}}$$

C'est-à-dire que, sans aucune hypothèse, on a toujours une preuve de $\mathcal{P} \vee \neg\mathcal{P}$.

Règles dérivées. Conjonction/disjonction.
Par exemple :

$$\boxed{\mathcal{P} \vee \mathcal{Q},\ \neg\mathcal{Q} \quad \vdash \quad \mathcal{P}}$$

On peut aussi considérer toutes les preuves liées aux lois de De Morgan, par exemple :

$$\boxed{\neg(\mathcal{P} \wedge \mathcal{Q}) \quad \vdash \quad \neg\mathcal{P} \vee \neg\mathcal{Q}}$$

On verra comment dériver cette règle à partir des règles de base dans les exemples ci-dessous.

Règles dérivées. Implications.
Voici d'abord la règle nommée *modus tollens* :

$$\boxed{\mathcal{P} \to \mathcal{Q},\ \neg\mathcal{Q} \quad \vdash \quad \neg\mathcal{P}}$$

Si $\mathcal{P}$ implique $\mathcal{Q}$, mais que l'on n'a pas $\mathcal{Q}$, alors on a une preuve de $\neg\mathcal{P}$. Dans les exemples ci-dessous, on explique comment dériver cette règle à partir des règles de base.
Le principe des tiers exclu permet une preuve par disjonction des cas :

$$\boxed{\mathcal{P} \to \mathcal{Q},\ \neg\mathcal{P} \to \mathcal{Q} \quad \vdash \quad \mathcal{Q}}$$

Cela permet de faire des raisonnements selon les deux cas de $\mathcal{P} \vee \neg\mathcal{P}$ (quelle que soit la formule $\mathcal{P}$).

$$\begin{array}{lll} & \begin{array}{l}\mathcal{P}\\ \hline \mathcal{Q}\end{array} & \text{Premier cas} \\ & \begin{array}{l}\neg\mathcal{P}\\ \hline \mathcal{Q}\end{array} & \text{Second cas} \\ \vdash & \mathcal{Q} & \end{array}$$

Règles dérivées. Absurde et contraposée.
Raisonnement par l'absurde :

$$\boxed{\neg\mathcal{P} \to \bot \quad \vdash \quad \mathcal{P}}$$

Si $\neg\mathcal{P}$ entraîne une contradiction, alors on a prouvé $\mathcal{P}$.
Contraposition :

$$\boxed{\mathcal{P} \to \mathcal{Q} \quad \vdash \quad \neg\mathcal{Q} \to \neg\mathcal{P}}$$

Ou encore :

$$\boxed{\neg\mathcal{Q} \to \neg\mathcal{P} \quad \vdash \quad \mathcal{P} \to \mathcal{Q}}$$

3.4. Exemples

> **Exemple.**
>
> $$\mathcal{P} \vee \mathcal{Q} \to \mathcal{P} \wedge \mathcal{Q} \quad \vdash \quad \mathcal{P} \leftrightarrow \mathcal{Q}$$

	$\mathcal{P} \vee \mathcal{Q} \to \mathcal{P} \wedge \mathcal{Q}$	
	$\mathcal{P}$	On suppose $\mathcal{P}$
	$\mathcal{P} \vee \mathcal{Q}$	
	$\mathcal{P} \wedge \mathcal{Q}$	Par l'hypothèse de départ
	$\mathcal{Q}$	
	$\mathcal{P} \to \mathcal{Q}$	Premier sens de l'équivalence
	$\mathcal{Q}$	On suppose $\mathcal{Q}$
	$\mathcal{P} \vee \mathcal{Q}$	
	$\mathcal{P} \wedge \mathcal{Q}$	Par l'implication dans l'hypothèse de départ
	$\mathcal{P}$	
	$\mathcal{Q} \to \mathcal{P}$	Second sens de l'équivalence
$\vdash$	$\mathcal{P} \leftrightarrow \mathcal{Q}$	

Exemple.
Prouvons directement le *modus tollens* par un raisonnement par l'absurde :

$$\mathcal{P} \to \mathcal{Q},\ \neg\mathcal{Q} \quad \vdash \quad \neg\mathcal{P}$$

	$\mathcal{P} \to \mathcal{Q}$	
	$\neg\mathcal{Q}$	
	$\mathcal{P}$	Par l'absurde, on suppose $\mathcal{P}$
	$\mathcal{Q}$	On obtient $\mathcal{Q}$ par l'hypothèse de départ
	$\bot$	Contradiction car on a $\mathcal{Q}$ et $\neg\mathcal{Q}$
$\vdash$	$\neg\mathcal{P}$	

Exemple.
Prouvons directement la règle de De Morgan suivante :

$$\neg(\mathcal{P} \wedge \mathcal{Q}) \quad \vdash \quad \neg\mathcal{P} \vee \neg\mathcal{Q}$$

	$\neg(\mathcal{P} \wedge \mathcal{Q})$	
	$\quad \mathcal{P}$	Premier cas : on suppose $\mathcal{P}$
	$\quad\quad \mathcal{Q}$	Par l'absurde, on suppose $\mathcal{Q}$
	$\quad\quad \mathcal{P} \wedge \mathcal{Q}$	
	$\quad\quad \bot$	Contradiction, obtenue grâce à l'hypothèse de départ
	$\quad \neg\mathcal{Q}$	
	$\quad \neg\mathcal{P} \vee \neg\mathcal{Q}$	
	$\quad \neg\mathcal{P}$	Second cas : on suppose $\neg\mathcal{P}$
	$\quad \neg\mathcal{P} \vee \neg\mathcal{Q}$	
$\vdash$	$\neg\mathcal{P} \vee \neg\mathcal{Q}$	Conclusion obtenue dans les deux cas du tiers exclu

3.5. Absurde et contraposée

Dans la logique classique, on peut toujours remplacer un raisonnement par l'absurde par un raisonnement par contraposition.

Absurde. $\neg\mathcal{P} \to \bot \quad \vdash \quad \mathcal{P}$.

Contraposée. $\neg\mathcal{Q} \to \neg\mathcal{P} \quad \vdash \quad \mathcal{P} \to \mathcal{Q}$ (ou encore $\mathcal{P} \to \mathcal{Q} \quad \vdash \quad \neg\mathcal{Q} \to \neg\mathcal{P}$ par double négation).

Contraposée vers absurde :

	$\neg\mathcal{P} \to \bot$	
	$\neg\bot \to \neg\neg\mathcal{P}$	Par contraposition
	$\top \to \neg\neg\mathcal{P}$	Car $\neg\bot = \top$
	$\top \to \mathcal{P}$	Car $\neg\neg\mathcal{P}$ donne $\mathcal{P}$
	$\top$	Or $\top$ toujours vrai
$\vdash$	$\mathcal{P}$	Car $\top \to \mathcal{P}$

Absurde vers contraposée :

	$\neg\mathcal{Q} \to \neg\mathcal{P}$	
	$\quad \mathcal{P}$	
	$\quad\quad \neg\mathcal{Q}$	
	$\quad\quad \neg\mathcal{P}$	Car $\neg\mathcal{Q} \to \neg\mathcal{P}$
	$\quad\quad \bot$	Contradiction car on a $\mathcal{P}$ et $\neg\mathcal{P}$
	$\quad \mathcal{Q}$	Car on a prouvé que supposer $\neg\mathcal{Q}$ entraîne contradiction
$\vdash$	$\mathcal{P} \to \mathcal{Q}$	Car chaque fois qu'on a $\mathcal{P}$ on a aussi $\mathcal{Q}$

3.6. Résumé

Voici les principales règles d'inférence utilisées dans ce chapitre avec en gras les règles de base de la logique classique.

Nom	Règle
Réutilisation	$\mathcal{P} \vdash \mathcal{P}$
Élimination de $\wedge$	$\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$ $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}$
Introduction de $\wedge$	$\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$
Élimination de la double négation	$\neg\neg\mathcal{P} \vdash \mathcal{P}$
Introduction de $\vee$	$\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q}$ $\mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}$
Élimination de $\vee$	$\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \to \mathcal{R}, \mathcal{Q} \to \mathcal{R} \vdash \mathcal{R}$
Implication (modus ponens)	$\mathcal{P}, \mathcal{P} \to \mathcal{Q} \vdash \mathcal{Q}$
Introduction de $\to$	Si $\mathcal{P} \vdash \mathcal{Q}$ alors $\vdash \mathcal{P} \to \mathcal{Q}$
Explosion (ex falso)	$\bot \vdash \mathcal{P}$
Équivalence	$\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{P} \to \mathcal{Q}$ $\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{Q} \to \mathcal{P}$ $\mathcal{P} \to \mathcal{Q}, \mathcal{Q} \to \mathcal{P} \vdash \mathcal{P} \leftrightarrow \mathcal{Q}$
Définition de la négation	$\neg\mathcal{P}$ est définie par $\mathcal{P} \to \bot$
Introduction de la double négation	$\mathcal{P} \vdash \neg\neg\mathcal{P}$
Tiers exclu	$\vdash \mathcal{P} \vee \neg\mathcal{P}$
Modus tollens	$\mathcal{P} \to \mathcal{Q}, \neg\mathcal{Q} \vdash \neg\mathcal{P}$
Lois de De Morgan	$\neg(\mathcal{P} \wedge \mathcal{Q}) \vdash \neg\mathcal{P} \vee \neg\mathcal{Q}$ …
Raisonnement par l'absurde	$\neg\mathcal{P} \to \bot \vdash \mathcal{P}$
Contraposée	$\mathcal{P} \to \mathcal{Q} \vdash \neg\mathcal{Q} \to \neg\mathcal{P}$ $\neg\mathcal{Q} \to \neg\mathcal{P} \vdash \mathcal{P} \to \mathcal{Q}$

4. Un aperçu du théorème de complétude de Gödel

Le théorème d'*incomplétude* de Gödel (1931) est le but ultime de ce livre. Ce théorème va bouleverser les fondements de la logique. Mais pour vraiment appréhender l'importance du théorème, il faut d'abord comprendre le théorème de *complétude* (1929) qui, lui, est plutôt rassurant. L'énoncé est même banal car c'est exactement ce à quoi tout le monde s'attend.

Théorème 1 (Théorème de complétude).
Pour la logique « Vrai/Faux » :

$$\mathcal{P} \vDash \mathcal{Q} \quad \textit{si et seulement si} \quad \mathcal{P} \vdash \mathcal{Q}.$$

Le point clé est que l'on se place ici dans le cadre de la logique propositionnelle, celle présentée dans le chapitre « Logique vrai/faux ». C'est aussi vrai dans le cadre plus général de la logique du premier ordre (voir le chapitre « Logique du premier ordre »). C'est d'ailleurs dans ce cadre qu'on prouvera ce théorème (voir le chapitre « Théorème de complétude »).
Détaillons chacune des implications de l'équivalence de l'énoncé.

Sens réciproque $\Leftarrow$. $\mathcal{P} \vdash \mathcal{Q}$ implique $\mathcal{P} \vDash \mathcal{Q}$.
Cette implication s'appelle le « théorème de validité » (*Soundness Theorem*), c'est un énoncé de « bon sens ». Si on dispose d'une preuve que $\mathcal{P}$ entraîne $\mathcal{Q}$ et que $\mathcal{P}$ est vrai alors $\mathcal{Q}$ est vrai. Autrement dit quand on a prouvé un théorème, et si les hypothèses sont vraies, alors la conclusion est vraie ! Noter une nouvelle fois, que dans la définition formelle de preuve, il n'y a pas de concept de vrai ou faux. Le théorème de validité signifie que les règles d'inférence choisies pour définir ce qu'est une preuve sont cohérentes et compatibles avec les règles de la logique propositionnelle.

Sens direct $\Rightarrow$. $\mathcal{P} \vDash \mathcal{Q}$ implique $\mathcal{P} \vdash \mathcal{Q}$.
Ce sens correspond au *théorème de complétude*. Cela signifie que, pour tout énoncé que l'on constate comme étant vrai, alors on peut trouver une preuve qui en fait un théorème. Ainsi, si on constate qu'à chaque fois que $\mathcal{P}$ est vrai alors $\mathcal{Q}$ est aussi vrai, alors il existe une preuve que $\mathcal{P}$ entraîne $\mathcal{Q}$. Ce résultat est valable dans le cadre de la logique propositionnelle (notre logique Vrai/Faux).
Cela permet de faire de beaux rêves : en logique propositionnelle, tout ce qui est vrai est prouvable et réciproquement. Le cauchemar commencera plus tard dans ce livre…

Exercices

Preuve par table de vérité

Exercice 1. Dire si les conséquences logiques suivantes sont valides ou pas. Justifier votre réponse.

$$\begin{aligned}
(\mathcal{P} \wedge \neg\mathcal{Q}) \vee (\neg\mathcal{P} \wedge \mathcal{Q}) &\overset{?}{\vDash} \mathcal{P} \vee \mathcal{Q} \\
\mathcal{P} \to \mathcal{Q},\ \neg\mathcal{P} &\overset{?}{\vDash} \neg\mathcal{Q} \\
\mathcal{P} \to \mathcal{Q},\ \mathcal{R} \to \mathcal{Q} &\overset{?}{\vDash} \mathcal{P} \to \mathcal{R} \\
\mathcal{P} \to \neg\mathcal{Q} &\overset{?}{\vDash} \mathcal{Q} \to \neg\mathcal{P} \\
\mathcal{P} \to \mathcal{Q},\ \mathcal{P} \vee \neg\mathcal{Q} &\overset{?}{\vDash} \mathcal{Q} \\
\mathcal{P} \vee \neg\mathcal{Q},\ \mathcal{P} \vee \neg\mathcal{R},\ \mathcal{R} \vee \neg\mathcal{Q} &\overset{?}{\vDash} \neg\mathcal{P} \to \mathcal{Q} \wedge \mathcal{R}
\end{aligned}$$

Preuve formelle

Exercice 2. Expliquer en français les théorèmes suivants :

$$\begin{array}{rcl} \mathcal{P},\ \mathcal{Q} & \vdash & \mathcal{P} \wedge \mathcal{Q} \\ \mathcal{P} \wedge (\mathcal{P} \to \mathcal{Q}) & \vdash & \mathcal{Q} \\ \mathcal{P} \to \mathcal{Q},\ \mathcal{Q} \to \mathcal{R} & \vdash & \mathcal{P} \to \mathcal{R} \\ \neg\mathcal{Q} \to \neg\mathcal{P} & \vdash & \mathcal{P} \to \mathcal{Q} \\ & \vdash & \mathcal{P} \vee \neg\mathcal{P} \end{array}$$

Exercice 3. On considère le théorème suivant :

$$\mathcal{P} \vee \mathcal{Q},\ \mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{S} \quad \vdash \quad \mathcal{R} \vee \mathcal{S}$$

Compléter les lignes manquantes de la preuve et commenter toutes les lignes.

	…	Hypothèse 1
	…	…
	…	…
	$\mathcal{P}$	Sous-hypothèse. Premier cas du OU
	…	On a utilisé l'implication …
	…	Sous-conclusion
	…	Sous-hypothèse. …
	…	…
	…	…
$\vdash$	…	Conclusion car …

Règles d'inférence

Exercice 4. Donner la preuve en déduction naturelle des théorèmes suivants :

$$\begin{array}{rcl} \mathcal{P} \to \mathcal{Q},\ \mathcal{Q} \to \mathcal{R} & \vdash & \mathcal{P} \to \mathcal{R} \\ \mathcal{P} & \vdash & (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \neg\mathcal{Q}) \\ (\mathcal{P} \to \mathcal{Q}) \to \mathcal{R} & \vdash & \mathcal{P} \to (\mathcal{Q} \to \mathcal{R}) \\ \mathcal{P} \leftrightarrow \mathcal{Q} & \vdash & (\mathcal{P} \vee \mathcal{Q}) \to (\mathcal{P} \wedge \mathcal{Q}) \\ \neg\mathcal{P} \vee \neg\mathcal{Q} & \vdash & \neg(\mathcal{P} \wedge \mathcal{Q}) \end{array}$$

Exercice 5. Donner la preuve en déduction naturelle des théorèmes suivants :

$$\begin{array}{rcl} \neg\mathcal{B} \to (\mathcal{A} \vee \mathcal{B}) & \vdash & \mathcal{A} \\ \mathcal{C} \to (\mathcal{D} \to \mathcal{E}) & \vdash & (\mathcal{C} \wedge \mathcal{D}) \to \mathcal{E} \\ \mathcal{F} \leftrightarrow \mathcal{G},\ \mathcal{G} \leftrightarrow \mathcal{H} & \vdash & \mathcal{F} \leftrightarrow \mathcal{H} \\ & \vdash & (\mathcal{I} \to \mathcal{J}) \vee (\mathcal{J} \to \mathcal{I}) \\ \mathcal{K} \vee \mathcal{L},\ \mathcal{L} \vee \mathcal{M},\ \neg\mathcal{L} & \vdash & \mathcal{K} \wedge \mathcal{M} \end{array}$$

Exercice 6. Le but de l'exercice est de montrer que la règle du tiers exclu « $\vdash \mathcal{P} \vee \neg\mathcal{P}$ » (sans hypothèses) se déduit des règles de base. On rappelle que $\neg\mathcal{P}$ signifie par définition $\mathcal{P} \to \bot$.

1. Vérifier que $\neg\neg\mathcal{Q}$ s'écrit $(\mathcal{Q} \to \bot) \to \bot$.
2. Voici la preuve de :

$$\vdash \quad \big((\mathcal{P} \vee \neg\mathcal{P}) \to \bot\big) \to \bot$$

$$
\begin{array}{l|l}
 & \varnothing \\
\hline
 & \quad (\mathcal{P} \vee \neg\mathcal{P}) \to \bot \\
 & \quad\quad \mathcal{P} \\
 & \quad\quad\quad \mathcal{P} \vee \neg\mathcal{P} \\
 & \quad\quad\quad \bot \\
 & \quad\quad \mathcal{P} \to \bot \\
 & \quad\quad \neg\mathcal{P} \\
 & \quad\quad \mathcal{P} \vee \neg\mathcal{P} \\
 & \quad\quad \bot \\
\vdash & \quad \big((\mathcal{P} \vee \neg\mathcal{P}) \to \bot\big) \to \bot
\end{array}
$$

Expliquer chaque ligne et vérifier que seules les règles de base sont utilisées.

3. Conclure à l'aide d'une règle de base.

Exercice 7. Le but est d'obtenir des règles dérivées à partir des règles de base.

1. Prouver la règle dérivée
$$\mathcal{P} \quad \vdash \quad \neg\neg\mathcal{P}$$
uniquement à l'aide des règles de base. On rappelle que $\neg\mathcal{P}$ signifie $\mathcal{P} \to \bot$.
2. On s'autorise ici les règles de base et le principe du tiers exclu (qu'on a déduit des règles de base). On peut donc faire des raisonnement en distinguant les cas $\mathcal{P}$ et $\neg\mathcal{P}$. Prouver la règle du raisonnement par l'absurde :
$$\neg\mathcal{P} \to \bot \quad \vdash \quad \mathcal{P}$$
3. En déduire qu'on pourrait dériver cette forme de la contraposition, uniquement à l'aide des règles de base :
$$\mathcal{P} \to \mathcal{Q} \quad \vdash \quad \neg\mathcal{Q} \to \neg\mathcal{P}$$

Un aperçu du théorème de complétude de Gödel

Exercice 8. Montrer qu'on a à la fois
$$\neg(\mathcal{P} \vee \mathcal{Q}) \quad \vDash \quad (\neg\mathcal{P}) \wedge (\neg\mathcal{Q})$$
et
$$\neg(\mathcal{P} \vee \mathcal{Q}) \quad \vdash \quad (\neg\mathcal{P}) \wedge (\neg\mathcal{Q})$$

Exercice 9. Montrer qu'on a à la fois
$$\neg\mathcal{P} \vee \mathcal{Q},\ \neg\mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{R} \quad \vDash \quad \mathcal{R}$$
et
$$\neg\mathcal{P} \vee \mathcal{Q},\ \neg\mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{R} \quad \vdash \quad \mathcal{R}$$

Chapitre 3

Quels connecteurs sont utiles ?

Les connecteurs ¬ (NON), ∧ (ET), ∨ (OU) suffisent pour construire toutes les formules de la logique Vrai/Faux.

1. Évaluation des formules

1.1. Rappels

On rappelle qu'une ***formule*** est un mot fini composé à partir :

- d'un ensemble $\text{Var} = \{P_1, P_2, \ldots, P_n\}$ de variables propositionnelles, où $n \geqslant 1$,
- de parenthèses (et),
- et d'un ensemble Conn de connecteurs logiques.

Jusqu'ici notre choix de connecteurs logiques était :

$$\text{Conn} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$$

Ce choix est-il pertinent ? Est-ce qu'il est suffisant pour exprimer des phénomènes complexes ? Pour répondre à ces questions, nous allons modéliser l'évaluation d'une formule comme l'évaluation d'une fonction n'ayant en entrées et en sortie que des valeurs **0**/**1**.
Ainsi dans ce chapitre, les valeurs booléennes sont $\mathbb{B} = \{\mathbf{0}, \mathbf{1}\}$ avec **1** pour V (Vrai) et **0** (Faux).

1.2. Fonctions abstraites de $\{0,1\}^n \to \{0,1\}$

Soit $n \geqslant 1$ un entier fixé. On appelle ***fonction abstraite*** une fonction :

$$f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$$

Soit $x = (\varepsilon_1, \ldots, \varepsilon_n) \in \{\mathbf{0}, \mathbf{1}\}^n$. C'est un vecteur dont toutes les coordonnées sont **0** ou **1**. De plus $f(x)$ vaut **0** ou **1**.

Exemple.

Soit $n = 3$. Définissons une fonction $f : \{\mathbf{0},\mathbf{1}\}^3 \to \{\mathbf{0},\mathbf{1}\}$ par ses $2^n = 8$ valeurs.

x	$f(x)$
$(\mathbf{0},\mathbf{0},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{0},\mathbf{0},\mathbf{1})$	$\mathbf{0}$
$(\mathbf{0},\mathbf{1},\mathbf{0})$	$\mathbf{1}$
$(\mathbf{0},\mathbf{1},\mathbf{1})$	$\mathbf{1}$
$(\mathbf{1},\mathbf{0},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{1},\mathbf{0},\mathbf{1})$	$\mathbf{1}$
$(\mathbf{1},\mathbf{1},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{1},\mathbf{1},\mathbf{1})$	$\mathbf{1}$

Rappel. Soient E et F deux ensembles finis. Alors le nombre total de fonctions différentes $f : E \to F$ est $(\operatorname{Card} F)^{\operatorname{Card} E}$. Ici $\operatorname{Card} E = \operatorname{Card}\{\mathbf{0},\mathbf{1}\}^n = 2^n$ et $\operatorname{Card} F = \operatorname{Card}\{\mathbf{0},\mathbf{1}\} = 2$. Ainsi le nombre total de fonctions abstraites $f : \{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$ vaut $2^{(2^n)}$.

1.3. Fonctions à partir d'une formule

Nous allons maintenant construire une fonction $f_{\mathcal{P}} : \{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$ à partir d'une formule.
Soit $\mathcal{P}$ une formule, exprimée à partir des n variables propositionnelles $P_1,\dots,P_n$. L'*évaluation* de $\mathcal{P}$ en $x = (\varepsilon_1,\dots,\varepsilon_n) \in \{\mathbf{0},\mathbf{1}\}^n$ consiste à attribuer la valeur ε_1 ($\mathbf{0}$ ou $\mathbf{1}$) à P_1, la valeur ε_2 à P_2,... Le résultat de cette évaluation est une valeur $\mathbf{0}$ ou $\mathbf{1}$.
Note : Une construction rigoureuse de cette fonction d'évaluation sera proposée dans le chapitre « Théorème de compacité ».
L'évaluation de $\mathcal{P}$ correspond donc à une fonction :

$$\begin{array}{rcl} f_{\mathcal{P}} : \{\mathbf{0},\mathbf{1}\}^n & \longrightarrow & \{\mathbf{0},\mathbf{1}\} \\ x = (\varepsilon_1,\dots,\varepsilon_n) & \longmapsto & f_{\mathcal{P}}(\varepsilon_1,\dots,\varepsilon_n) \end{array}$$

C'est une autre façon de présenter la table de vérité de la formule $\mathcal{P}$.

Exemple.
Soit $\mathcal{P}$ la formule suivante :

$$\mathcal{P} : \qquad (P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)$$

On obtient une fonction $f_{\mathcal{P}} : \{\mathbf{0},\mathbf{1}\}^3 \to \{\mathbf{0},\mathbf{1}\}$ qu'on calcule valeur par valeur. Par exemple en $x = (\mathbf{0},\mathbf{1},\mathbf{0})$ on attribue (F, V, F) aux variables (P_1, P_2, P_3) et alors la formule $\mathcal{P}$ s'évalue à $(\mathrm{F} \vee \mathrm{V}) \wedge (\neg \mathrm{F} \vee \mathrm{F})$ et vaut donc V. Ainsi $f_{\mathcal{P}}(\mathbf{0},\mathbf{1},\mathbf{0}) = \mathbf{1}$.

x	$f_{\mathcal{P}}(x)$
$(\mathbf{0},\mathbf{0},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{0},\mathbf{0},\mathbf{1})$	$\mathbf{0}$
$(\mathbf{0},\mathbf{1},\mathbf{0})$	$\mathbf{1}$
$(\mathbf{0},\mathbf{1},\mathbf{1})$	$\mathbf{1}$
$(\mathbf{1},\mathbf{0},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{1},\mathbf{0},\mathbf{1})$	$\mathbf{1}$
$(\mathbf{1},\mathbf{1},\mathbf{0})$	$\mathbf{0}$
$(\mathbf{1},\mathbf{1},\mathbf{1})$	$\mathbf{1}$

Ici on remarque que notre fonction $f_{\mathcal{P}}$ est exactement la même que notre fonction abstraite f définie

> dans l'exemple précédent.

1.4. Réaliser toutes les fonctions ?

Dans l'exemple précédent on avait une formule $\mathcal{P}$ dont la fonction associée $f_{\mathcal{P}}$ était la même qu'une fonction abstraite définie auparavant. Mais est-ce toujours possible ? Pour une fonction abstraite $f : \{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$, peut-on trouver une formule $\mathcal{P}$ telle que la fonction $f_{\mathcal{P}}$ soit exactement cette fonction f ?
Si la réponse était négative, cela voudrait dire que nos connecteurs ne sont pas suffisants pour modéliser toutes les fonctions et qu'il faudrait inventer de nouveaux connecteurs pour réaliser certaines fonctions. La question se pose d'autant plus que, lorsque n est grand, le nombre de fonctions f à réaliser est 2^{2^n}, ce nombre étant vraiment très grand.

1.5. Connecteurs unaires

En attendant d'aborder un résultat théorique général, commençons par étudier à la main les cas $n = 1$ et $n = 2$.
Pour $n = 1$, il y a $4 = 2^{2^1}$ fonctions $f : \{\mathbf{0},\mathbf{1}\} \to \{\mathbf{0},\mathbf{1}\}$. Voici les 4 tables de vérité possibles

	$\top$
0	**1**
1	**1**

	$\bot$
0	**0**
1	**0**

	id
0	**0**
1	**1**

	$\neg$
0	**1**
1	**0**

Ces 4 fonctions sont $\top$ (toujours vrai), $\bot$ (toujours faux), id (identité), $\neg$ (négation).
On résume le cas $n = 1$ en une seule table.

Symbole	Réalisation	Nom
$\top$	$P \vee \neg P$	tautologie / toujours vrai
$\bot$	$P \wedge \neg P$	contradiction / toujours faux
id	P	identité
$\neg$	$\neg P$	négation

La seconde colonne est une réalisation par une formule à l'aide des connecteurs $\neg$, $\wedge$, $\vee$.

1.6. Connecteurs binaires

Pour $n = 2$, il y a $16 = 2^{2^2}$ fonctions $f : \{\mathbf{0},\mathbf{1}\}^2 \to \{\mathbf{0},\mathbf{1}\}$. On dresse ci-dessous la liste de toutes ces fonctions et leur réalisation par des formules. La seconde colonne propose une réalisation par une formule à l'aide des connecteurs $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$. La dernière colonne indique les 4 valeurs de la fonction prises lorsque (P,Q) vaut successivement $(\mathbf{0},\mathbf{0})$, $(\mathbf{0},\mathbf{1})$, $(\mathbf{1},\mathbf{0})$, $(\mathbf{1},\mathbf{1})$.

Symbole	Réalisation	Nom	Valeurs
$\bot$	$P \wedge \neg P$	toujours faux / contradiction	**0000**
$\wedge$	$P \wedge Q$	ET/ conjonction	**0001**
	$P \wedge \neg Q$	P et (non Q)	**0010**
	P	projection gauche	**0011**
	$\neg P \wedge Q$	(non P) et Q	**0100**
	Q	projection droite	**0101**
$\oplus$	$(P \vee Q) \wedge \neg(P \wedge Q)$	XOR / ou exclusif	**0110**
$\vee$	$P \vee Q$	OU/ disjonction	**0111**
$\downarrow$	$\neg(P \vee Q)$	NOR	**1000**
$\leftrightarrow$	$P \leftrightarrow Q$	équivalence	**1001**
	$\neg Q$	négation de Q	**1010**
	$Q \rightarrow P$	Q implique P	**1011**
	$\neg P$	négation de P	**1100**
$\rightarrow$	$P \rightarrow Q$	P implique Q	**1101**
$\uparrow$	$\neg(P \wedge Q)$	NAND / Sheffer	**1110**
$+\ \top$	$P \vee \neg P$	toujours vrai / tautologie	**1111**

Par exemple le connecteur $\downarrow$/NOR vérifie : $\mathbf{0} \downarrow \mathbf{0} = \mathbf{1}$, $\mathbf{0} \downarrow \mathbf{1} = \mathbf{0}$, $\mathbf{1} \downarrow \mathbf{0} = \mathbf{0}$, $\mathbf{1} \downarrow \mathbf{1} = \mathbf{0}$. Mais en fait $P \downarrow Q$ se réalise par la formule $\neg(P \vee Q)$. Ainsi on n'a pas vraiment besoin d'utiliser un symbole spécial pour cette opération.
Cependant il y a deux nouveaux connecteurs utiles en pratique : XOR et NAND.

- Le ***ou exclusif***, noté $\oplus$/XOR, est vrai lorsque l'un des termes est vrai, mais pas les deux. Il correspond à l'addition $x + y$ modulo 2.
- Le connecteur $\uparrow$/NAND, aussi appelé symbole de Sheffer, se réalise comme la négation du ET : $P \uparrow Q = \neg(P \wedge Q)$.

2. Un système complet de connecteurs

2.1. Théorème

Théorème 1.
L'ensemble $\{\neg, \wedge\}$ des connecteurs NON et ET forme un système complet de connecteurs.

On verra la preuve dans la prochaine section. Cela signifie que, quelle que soit $n \geqslant 1$ et quel que soit $f : \{\mathbf{0}, \mathbf{1}\}^n \rightarrow \{\mathbf{0}, \mathbf{1}\}$, alors il existe une formule $\mathcal{P}$ ne faisant intervenir que les connecteurs $\neg$ et $\wedge$, telle que la fonction $f_{\mathcal{P}}$ issue de $\mathcal{P}$ soit exactement cette fonction f. Autrement dit f et $f_{\mathcal{P}}$ ont la même table de vérité.
Ce qu'il est important de retenir, c'est qu'avec un nombre fini de connecteurs, on réalise toutes les fonctions booléennes imaginables, quel que soit n.
Une autre façon de comprendre ce théorème est de dire que pour n'importe quelle formule (avec n'importe quels connecteurs), on peut trouver une formule logiquement équivalente qui ne contient que les connecteurs $\neg$ et $\wedge$. On rappelle que deux formules $\mathcal{P}$ et $\mathcal{Q}$ sont logiquement équivalentes (noté $\mathcal{P} \equiv \mathcal{Q}$) si elles ont la même table de vérité.
Question pas si bête ! Si on n'a besoin que de $\{\neg, \wedge\}$ pour réaliser toutes nos fonctions, alors pourquoi ajoute-t-on les autres connecteurs $\vee$, $\rightarrow$, $\leftrightarrow$? Tout simplement pour des raisons pratiques, afin de raccourcir l'écriture des formules et surtout pour faciliter la compréhension logique des formules.

2.2. Exemples

1. Pour $n = 1$ et $n = 2$, nous avons déjà vu comment réaliser toutes les fonctions (quitte à utiliser aussi $\vee$, $\rightarrow$, $\leftrightarrow$).
2. Une même fonction peut être réalisée par des formules différentes. Par exemple $P \wedge \neg Q$, $\neg Q \wedge P \wedge \neg Q$ et $(\neg\neg P) \wedge (\neg Q)$ réalisent la même fonction. C'est encore moins unique si on autorise tous les connecteurs $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$, par exemple $P \rightarrow Q$, $\neg P \vee Q$ et $\neg(P \wedge \neg Q)$ sont des formules qui réalisent la même fonction.
3. Soit $n = 3$ et f la fonction qui vaut **1** en $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ et $(\mathbf{1}, \mathbf{1}, \mathbf{1})$ et qui vaut **0** partout ailleurs. Cette fonction est réalisée par la formule :

   $$\mathcal{P} : \qquad (\neg P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge R)$$

   Problème : il y a un $\vee$ (OU) dans la formule, alors qu'on voudrait utiliser seulement $\neg$ et $\wedge$, on verra la solution dans le paragraphe suivant.

2.3. Connecteurs usuels à partir de $\neg$ et $\wedge$

Puisque les connecteurs $\neg$ et $\wedge$ suffisent, ils doivent permettre de retrouver nos autres connecteurs favoris : $\vee$, $\rightarrow$, $\leftrightarrow$.
Effectivement :

$$\begin{array}{rcl} P \vee Q & \equiv & \neg(\neg P \wedge \neg Q) \\ P \rightarrow Q & \equiv & \neg P \vee Q \quad \equiv \quad \neg(P \wedge \neg Q) \\ P \leftrightarrow Q & \equiv & (P \rightarrow Q) \wedge (Q \rightarrow P) \quad \equiv \quad \big(\neg(P \wedge \neg Q)\big) \wedge \big(\neg(Q \wedge \neg P)\big) \end{array}$$

On rappelle que la notation $\equiv$ signifie que les formules ont exactement les mêmes tables de vérité. Bien sûr, ces relations sont aussi valables pour des formules $\mathcal{P}$ et $\mathcal{Q}$ (à la place des variables P et Q).
Ainsi, toute formule exprimée à partir des connecteurs $\mathcal{C} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ peut s'écrire en une formule logiquement équivalente ne contenant que les connecteurs $\mathcal{C}_0 = \{\neg, \wedge\}$.

2.4. Autres systèmes complets de connecteurs

L'ensemble $\mathcal{C}_1 = \{\neg, \rightarrow\}$ est aussi un système complet de connecteurs. La preuve est simple à partir du théorème ; il suffit de montrer comment engendrer le connecteur $\wedge$ à partir de $\neg$ et $\rightarrow$:

$$P \wedge Q \equiv \neg(P \rightarrow \neg Q)$$

Ainsi, si on a une formule quelconque, le théorème nous permet d'obtenir une formule logiquement équivalente uniquement avec $\neg$ et $\wedge$. On peut alors remplacer chaque $\wedge$ via l'équivalence logique ci-dessus afin d'obtenir une formule ne contenant que des $\neg$ et des $\rightarrow$.

Encore plus fort, un seul connecteur permet de les engendrer tous ! $\mathcal{C}_2 = \{\uparrow\}$ est un système complet. On rappelle que $\uparrow$ / NAND est le connecteur correspondant à la négation du ET :

$$P \uparrow Q \equiv \neg(P \wedge Q)$$

Supposons ce connecteur $\uparrow$ défini. On retrouve alors $\neg$ et $\wedge$:

$$\neg P \equiv P \uparrow P \qquad\qquad P \wedge Q \equiv \neg(P \uparrow Q) \equiv (P \uparrow Q) \uparrow (P \uparrow Q)$$

Comme $\uparrow$ permet de reconstruire à la fois $\neg$ et $\wedge$, et que $\{\neg, \wedge\}$ est complet, il en résulte que $\{\uparrow\}$ est un système complet.
D'un point de vue informatique, cela signifie qu'un seul type de porte logique, la porte NAND, permet de modéliser n'importe quelle fonction booléenne.

3. Preuve

Cette section est consacrée à la preuve du théorème. On va prouver que $\mathcal{C} = \{\neg, \wedge, \vee\}$ est un système complet de connecteurs. Puisque l'on a vu que $\vee$ est engendré par $\{\neg, \wedge\}$ via $P \vee Q \equiv \neg(\neg P \wedge \neg Q)$, on aura bien prouvé que $\mathcal{C}_0 = \{\neg, \wedge\}$ est un système complet de connecteurs.
Nous allons réaliser chaque fonction abstraite $f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$ par une formule explicite, appelée ***forme normale disjonctive***.

3.1. Étape 1

Nous commençons par réaliser des fonctions très simples.
Soit $a = (\varepsilon_1, \ldots, \varepsilon_n) \in \{\mathbf{0}, \mathbf{1}\}^n$ fixé. Soit $f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$ la fonction telle que $f(a) = \mathbf{1}$ et $f(x) = \mathbf{0}$ si $x \neq a$.
La formule $\mathcal{P}_a$ suivante réalise cette fonction :

$$\mathcal{P}_a : \qquad \varepsilon_1 P_1 \wedge \varepsilon_2 P_2 \wedge \ldots \wedge \varepsilon_n P_n$$

Ici, P_i désigne la i-ème variable propositionnelle, autrement dit la i-ème coordonnée de $\{\mathbf{0}, \mathbf{1}\}^n$. On rappelle que $\varepsilon_i = \mathbf{0}$ ou $\varepsilon_i = \mathbf{1}$ et on définit :

$$\varepsilon_i P_i = \begin{cases} P_i & \text{si } \varepsilon_i = \mathbf{1} \\ \neg P_i & \text{si } \varepsilon_i = \mathbf{0} \end{cases}$$

On adoptera la notation suivante :

$$\mathcal{P}_a : \qquad \bigwedge_{i=1}^{n} \varepsilon_i P_i$$

Cette notation est analogue à $\sum_{i=1}^{n} x_i$ pour $x_1 + x_2 + \cdots + x_n$ ou à $\prod_{i=1}^{n} x_i$ pour $x_1 x_2 \cdots x_n$.
Par exemple, la formule $P \wedge \neg Q \wedge \neg R$ donne une fonction qui vaut $\mathbf{1}$ en $(\mathbf{1}, \mathbf{0}, \mathbf{0})$ et vaut $\mathbf{0}$ partout ailleurs.
On rappelle que $a = (\varepsilon_1, \ldots, \varepsilon_n)$. Prouvons que $\mathcal{P}_a$ réalise la fonction annoncée :

$$\begin{aligned} \mathcal{P}_a \text{ vrai} \quad &\text{ssi} \quad \varepsilon_i P_i \text{ vrai, pour tout } i = 1, \ldots, n \\ &\text{ssi} \quad \begin{cases} P_i \text{ vrai} & \text{si } \varepsilon_i = \mathbf{1} \\ P_i \text{ faux} & \text{si } \varepsilon_i = \mathbf{0} \end{cases} \quad \text{pour tout } i = 1, \ldots, n \end{aligned}$$

Donc la fonction $f_{\mathcal{P}_a}$ associée vérifie $f_{\mathcal{P}_a}(x_1, \ldots, x_n) = \mathbf{1}$ si et seulement si $x_i = \varepsilon_i$, pour tout $i = 1, \ldots, n$.
Autrement dit, $f_{\mathcal{P}_a}(x) = \mathbf{1}$ si et seulement si $x = a$.

3.2. Étape 2

Soit $a_1, a_2, \ldots, a_k \in \{\mathbf{0}, \mathbf{1}\}^n$ une liste de k éléments de $\{\mathbf{0}, \mathbf{1}\}^n$ (avec $k \geqslant 1$).
Soit $\mathcal{P}_{a_1,\ldots,a_k}$ la formule suivante :

$$\mathcal{P}_{a_1,\ldots,a_k} : \qquad \mathcal{P}_{a_1} \vee \mathcal{P}_{a_2} \vee \cdots \vee \mathcal{P}_{a_k}$$

Autrement dit :

$$\mathcal{P}_{a_1,\ldots,a_k} : \qquad \bigvee_{j=1}^{k} \mathcal{P}_{a_j} = \bigvee_{j=1}^{k} \bigwedge_{i=1}^{n} \varepsilon_{i,j} P_i$$

Où on a noté $a_j = (\varepsilon_{1,j}, \varepsilon_{2,j}, \ldots, \varepsilon_{n,j})$.
Notons $f_{a_1,\ldots,a_k}$ la fonction associée à cette formule $\mathcal{P}_{a_1,\ldots,a_k}$. Alors :

$$f_{a_1,\ldots,a_k}(x) = \mathbf{1} \quad \text{ssi} \quad x \in \{a_1, a_2, \ldots, a_k\}$$

Exemple. Pour la formule $\mathcal{P}_a \vee \mathcal{P}_b$, la fonction associée $f_{a,b}$ vérifie $f(a) = \mathbf{1}$, $f(b) = \mathbf{1}$ et $f(x) = \mathbf{0}$ pour tous les autres x.
Prouvons notre affirmation.

- Si $x \in \{a_1, a_2, \ldots, a_k\}$, alors l'un des $\mathcal{P}_{a_j}$ est vrai, donc $\mathcal{P}_{a_1,\ldots,a_k}$ est vrai. Ainsi $f_{a_1,\ldots,a_k}(x) = \mathbf{1}$.
- Si $x \notin \{a_1, a_2, \ldots, a_k\}$, alors tous les $\mathcal{P}_{a_1}, \ldots, \mathcal{P}_{a_k}$ sont faux et donc $\mathcal{P}_{a_1,\ldots,a_k}$ est faux. Ainsi $f_{a_1,\ldots,a_k}(x) = \mathbf{0}$.

3.3. Étape 3

Soit $f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$ une fonction abstraite quelconque.

- Si $f(x) = \mathbf{0}$, quel que soit $x \in \{\mathbf{0}, \mathbf{1}\}^n$ alors on pose $\mathcal{P} = P_1 \wedge \neg P_1$ (toujours faux) et on a bien $f_{\mathcal{P}} = f$.
- Sinon, on note $\{a_1, \ldots, a_k\}$ les éléments de $\{\mathbf{0}, \mathbf{1}\}^n$ tels que $f(x) = \mathbf{1}$. Alors $f = f_{a_1,\ldots,a_k}$, d'après l'étape 2.

Cela termine la preuve.

3.4. Remarques

- La décomposition
$$\mathcal{P} : \quad \bigvee_{j=1}^{k} \bigwedge_{i=1}^{n} \varepsilon_{i,j} P_i$$
s'appelle la ***forme normale disjonctive***.
Cette forme normale est intéressante car cette décomposition est unique à l'ordre près (on a bien sûr $P \vee Q \equiv Q \vee P$ et $P \wedge Q \equiv Q \wedge P$).
- Une variante est la ***forme normale conjonctive*** :
$$\mathcal{Q} : \quad \bigwedge_{j=1}^{k} \bigvee_{i=1}^{n} \delta_{i,j} P_i$$
où $\delta_{i,j} \in \{\mathbf{0}, \mathbf{1}\}$. Son écriture est également unique (à l'ordre près).

> **Exemple.**
> Soit $\mathcal{P}$ la formule $P \leftrightarrow Q$. La forme normale disjonctive est :
> $$\mathcal{P} \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$$
> La forme normale conjonctive est :
> $$\mathcal{P} \equiv (\neg P \vee Q) \wedge (P \vee \neg Q)$$

Exercices

Évaluation des formules

Exercice 1. Trouver une formule pour $P \oplus Q$ (le ou exclusif) qui ne fait intervenir que les connecteurs $\neg$, $\wedge$, $\vee$. Trouver ensuite une formule ne faisant intervenir que $\neg$ et $\leftrightarrow$.

Exercice 2.

1. Pour $n = 3$. Combien y a-t-il de fonctions $f : \{\mathbf{0}, \mathbf{1}\}^3 \to \{\mathbf{0}, \mathbf{1}\}$?
2. La fonction « majorité » renvoie $\mathbf{1}$, si au moins deux des variables P, Q, R valent $\mathbf{1}$. Trouver une formule pour cette fonction qui ne fait intervenir que les connecteurs $\neg$, $\wedge$, $\vee$.
3. Quand est-ce que $P \oplus Q \oplus R$ est vrai ? Quel est le lien avec $P \leftrightarrow (Q \leftrightarrow R)$?
4. La fonction « exactement un » renvoie $\mathbf{1}$ lorsque exactement une des variables P, Q, ou R vaut $\mathbf{1}$. Trouver une formule pour cette fonction qui ne fait intervenir que les connecteurs $\neg$, $\wedge$, $\vee$.

Exercice 3. Soit $n \geqslant 2$.

1. Trouver une formule qui renvoie vrai, sauf si toutes les variables $P_1,\ldots,P_n$ sont vraies.
2. Quelle formule renvoie vrai lorsqu'au moins deux variables sont vraies.

Un système complet de connecteurs

Exercice 4.

1. Montrer que $\mathcal{C} = \{\neg, \leftrightarrow\}$ est un système complet de connecteurs.
2. Montrer que $\mathcal{C} = \{\vee, \leftrightarrow, \bot\}$ est un système complet de connecteurs. On rappelle que $\bot$ est la formule toujours fausse.
3. Montrer que $\mathcal{C} = \{\wedge, \vee\}$ n'est pas un système complet de connecteurs.
4. Montrer que $\mathcal{C} = \{\wedge, \oplus\}$ n'est pas un système complet de connecteurs ($\oplus$ désigne le ou exclusif).

Exercice 5. Montrer que $\{\downarrow\}$ forme un système complet. On rappelle que $P \downarrow Q \equiv \neg(P \vee Q)$.

Preuve

Exercice 6. Soit $n = 2$. On n'autorise que les connecteurs $\neg, \wedge, \vee$.

1. Trouver une formule qui réalise la fonction $f : \{\mathbf{0},\mathbf{1}\}^2 \to \{\mathbf{0},\mathbf{1}\}$, qui vaut $\mathbf{1}$ en $(\mathbf{1},\mathbf{0})$ et vaut $\mathbf{0}$ partout ailleurs.
2. Même question avec $g : \{\mathbf{0},\mathbf{1}\}^2 \to \{\mathbf{0},\mathbf{1}\}$, qui vaut $\mathbf{1}$ en $(\mathbf{1},\mathbf{1})$ et $\mathbf{0}$ partout ailleurs.
3. En déduire une formule pour $h : \{\mathbf{0},\mathbf{1}\}^2 \to \{\mathbf{0},\mathbf{1}\}$, qui vaut $\mathbf{1}$ exactement en $(\mathbf{1},\mathbf{0})$ et $(\mathbf{1},\mathbf{1})$. Simplifier cette formule.

Exercice 7. Soit $n = 2$.

1. Quelle fonction est associée à la formule $P \to Q$?
2. Trouver la forme normale disjonctive associée.

Exercice 8. Soit $n = 3$. On n'autorise que les connecteurs $\neg, \wedge, \vee$.

1. Trouver une formule qui réalise la fonction f qui vaut $\mathbf{1}$ uniquement en $(\mathbf{1},\mathbf{0},\mathbf{1})$.
2. Trouver une formule qui réalise la fonction g qui vaut $\mathbf{0}$ uniquement en $(\mathbf{0},\mathbf{1},\mathbf{1})$.
3. Trouver une formule qui réalise la fonction h qui vaut $\mathbf{1}$ uniquement en $(\mathbf{1},\mathbf{0},\mathbf{0})$, $(\mathbf{0},\mathbf{1},\mathbf{0})$, $(\mathbf{0},\mathbf{0},\mathbf{1})$. En déterminer une forme normale disjonctive et une forme normale conjonctive.

Chapitre 4

Théorème de compacité

Le théorème de compacité affirme que lorsqu'il y a une infinité d'hypothèses, il est suffisant d'en utiliser un nombre fini.

Ce chapitre est d'un niveau plus difficile que les précédents et peut être passé en première lecture.

1. Formules, formules satisfaisables, conséquence logique

1.1. Formules : définition par le haut

Dans le chapitre « Logique vrai/faux », on a défini une formule comme un mot fini à partir des éléments suivants :

- des variables propositionnelles $\mathrm{Var} = \{P_i\}_{i\in I}$,
- des parenthèses (et) ,
- et des connecteurs logiques $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.

Plus précisément :

Définition.
L'ensemble Form des ***formules*** est construit selon les règles suivantes :

1. Les variables propositionnelles sont des formules.
2. Si $\mathcal{P}$ et $\mathcal{Q}$ sont des formules, alors
$$\neg\mathcal{P}, \quad (\mathcal{P}\wedge\mathcal{Q}), \quad (\mathcal{P}\vee\mathcal{Q}), \quad (\mathcal{P}\to\mathcal{Q}), \quad (\mathcal{P}\leftrightarrow\mathcal{Q})$$
sont aussi des formules.
3. Rien d'autre n'est une formule.

Cette définition s'appelle la « définition par le haut » ou « définition inductive » des formules. On a vu au chapitre « Quels connecteurs sont utiles ? » que l'on peut même se contenter des seuls connecteurs $\{\neg,\wedge\}$ pour décrire toutes les formules. Dans ce chapitre, on s'intéresse au cas où l'ensemble $\mathrm{Var} = \{P_i\}_{i\in I}$ des variables propositionnelles est infini mais dénombrable. Cela signifie que l'on peut choisir $I = \mathbb{N}$ pour l'ensemble des indices de ces variables : $P_0, P_1, P_2, \ldots$

Remarque.
Comme le nombre de variables est infini, il faudrait une infinité d'indices i pour décrire toutes les formules. Mais on souhaite qu'une formule soit un mot formé à partir d'un alphabet fini. Pour cela, on va seulement considérer deux symboles pour décrire toutes les variables P et $'$ (prime). Ainsi on note P pour P_0, P' pour P_1, P'' pour P_2, etc. Ainsi une formule est un mot fini composé à partir de l'alphabet fini : $\{P, ', (,), \neg, \wedge, \vee, \to, \leftrightarrow\}$. Bien sûr, dans la pratique on conservera notre notation P_i pour plus de clarté.

1.2. Formules : définition par le bas

La définition précédente est en fait assez abstraite et décrit assez mal l'ensemble des formules. Voici une nouvelle définition, dite « définition par le bas » ou « définition récursive ».

Définition.
On définit des ensembles de formules Form_n par récurrence.

- $\text{Form}_0 = \{P_i\}_{i\in I}$ est l'ensemble des variables.
- Pour $n \geqslant 0$,

$$\text{Form}_{n+1} = \text{Form}_n \cup \{\neg\mathcal{P} \mid \mathcal{P} \in \text{Form}_n\} \cup \{(\mathcal{P} \wedge \mathcal{Q}) \mid \mathcal{P}, \mathcal{Q} \in \text{Form}_n\} \cup \dots$$

  (En faisant l'union correspondante pour tous les connecteurs $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$.)
- $\text{Form} = \bigcup_{n\geqslant 0} \text{Form}_n$ est la réunion de tous les Form_n.
- La ***hauteur*** d'une formule $\mathcal{P}$ est le plus petit entier n tel que $\mathcal{P} \in \text{Form}_n$.

Proposition 1.
Si l'ensemble $Var = \{P_i\}_{i\in I}$ des variables propositionnelles est dénombrable (ou fini), alors l'ensemble Form des formules est aussi dénombrable.

Démonstration. L'ensemble $\text{Form} = \bigcup_{n\geqslant 0} \text{Form}_n$ est une union indexée sur un ensemble dénombrable, où chaque ensemble Form_n est un ensemble dénombrable, donc Form est dénombrable. □

Dans tout le reste de ce chapitre, on supposera que l'ensemble des variables propositionnelles est fini ou dénombrable et donc que l'ensemble de toutes les formules est dénombrable.

1.3. Évaluation d'une formule

Nous avons déjà vu qu'évaluer une formule $\mathcal{P}$ consiste à attribuer une valeur Vrai ou Faux à chaque variable propositionnelle P_i, afin d'attribuer une valeur Vrai/Faux à la formule $\mathcal{P}$ ainsi évaluée.
Voici la définition rigoureuse d'une évaluation à partir de la définition « par le haut » des formules. On note ici **1** et **0** les deux valeurs booléennes V/F.

Définition.

- Soit $\nu : \{P_i\}_{i\in I} \to \{\mathbf{0}, \mathbf{1}\}$ une fonction, appelée ***distribution de valeurs*** ou ***valuation*** : pour chaque $i \in I$ on a $\nu(P_i) = \mathbf{0}$ ou $\nu(P_i) = \mathbf{1}$.
- On étend cette fonction $\nu : \text{Form} \to \{\mathbf{0}, \mathbf{1}\}$ à l'ensemble Form des formules :
  — $\nu(\neg\mathcal{P}) = \mathbf{1}$ ssi $\nu(\mathcal{P}) = \mathbf{0}$,
  — $\nu(\mathcal{P} \wedge \mathcal{Q}) = \mathbf{1}$ ssi $\nu(\mathcal{P}) = \mathbf{1}$ et $\nu(\mathcal{Q}) = \mathbf{1}$,
  — et selon le même principe pour $\vee$, $\to$, $\leftrightarrow$.
- La valeur $\nu(\mathcal{P})$ est l'***évaluation*** de $\mathcal{P}$ en ν.

1.4. Satisfaisabilité, conséquence logique

Définition.

- Une formule $\mathcal{P}$ est ***satisfaisable***, s'il existe une valuation ν, telle que $\nu(\mathcal{P})$ soit vraie. On dit alors que ν ***satisfait*** $\mathcal{P}$, ou que ν est un ***modèle*** pour $\mathcal{P}$.
- Un ensemble de formules $\Gamma = \{\mathcal{P}_j\}_{j\in J}$ est ***satisfaisable***, s'il existe une valuation ν, telle que, pour tout $j \in J$, $\nu(\mathcal{P}_j)$ soit vraie.
- Une formule, ou un ensemble de formules, qui n'est pas satisfaisable, est dit ***insatisfaisable*** ou ***contradictoire***.

Définition.
Soit Γ un ensemble de formules et $\mathcal{Q}$ une formule. On dit que $\mathcal{Q}$ est une ***conséquence logique*** de Γ, si toute valuation v qui satisfait Γ, satisfait aussi $\mathcal{Q}$. On note :

$$\boxed{\Gamma \models \mathcal{Q}}$$

Cela signifie que $\Gamma \cup \{\neg\mathcal{Q}\}$ n'est pas satisfaisable, c'est-à-dire qu'aucune distribution de valeurs ne peut satisfaire à la fois Γ et $\neg\mathcal{Q}$.
Il faut comprendre Γ comme des hypothèses et $\mathcal{Q}$ comme une conclusion. À chaque fois que toutes les hypothèses sont vérifiées, on constate que la conclusion l'est également.

2. Théorème de compacité

2.1. Énoncé

Soit Γ un ensemble fini ou infini de formules.

Théorème 1.
Γ est satisfaisable si, et seulement si, pour toute partie finie $\Gamma_0 \subset \Gamma$, Γ_0 est satisfaisable.

Afin de faciliter l'écriture de la preuve, on dira que Γ est ***finiment satisfaisable***, si pour toute partie finie $\Gamma_0 \subset \Gamma$, Γ_0 est satisfaisable. Ainsi le théorème de compacité affirme :

$$\Gamma \text{ satisfaisable} \quad \text{ssi} \quad \Gamma \text{ finiment satisfaisable}$$

Remarques :

1. L'implication directe est évidente. Si Γ est satisfaisable, alors n'importe quel sous-ensemble de Γ est aussi satisfaisable, donc en particulier Γ est finiment satisfaisable.
2. Si Γ est un ensemble fini, alors le théorème est évident, il n'y a rien à démontrer.
3. Si Γ est infini, la réciproque est loin d'être évidente. Certains résultats vrais pour des ensembles finis, ne sont plus vrais nécessairement vrais pour des ensembles infinis. Par exemple, une intersection finie d'intervalles ouverts est un intervalle ouvert (éventuellement vide). En revanche, ce n'est plus vrai pour une intersection infinie :
$$\bigcap_{n\geqslant 1} \left]-\tfrac{1}{n}, 1+\tfrac{1}{n}\right[= [0,1]$$
C'est un exemple frappant où une intersection infinie d'intervalles ouverts aboutit à un intervalle fermé.

2.2. Corollaires

Les corollaires suivants sont également appelés « théorème de compacité ».

Corollaire 1.
Γ est insatisfaisable si, et seulement si, il existe un ensemble fini $\Gamma_0 \subset \Gamma$ tel que Γ_0 soit insatisfaisable.

Autrement dit, si l'on n'arrive pas à satisfaire un ensemble de formules, cela ne peut pas être dû au fait qu'on en ait un nombre infini.
Voici la contrepartie positive :

Corollaire 2.
$\Gamma \models \mathcal{Q}$ si, et seulement si, il existe un ensemble fini $\Gamma_0 \subset \Gamma$ tel que $\Gamma_0 \models \mathcal{Q}$.

C'est la version qui nous intéresse le plus : si on a une infinité de formules Γ comme hypothèses avec comme conséquence logique la formule $\mathcal{Q}$, alors on peut en extraire un sous-ensemble fini Γ_0 qui entraîne toujours la même conséquence logique $\mathcal{Q}$. On passe donc d'un nombre infini d'hypothèses à un nombre fini.

2.3. Preuve des corollaires

Sans attendre la preuve du théorème, voici la démonstration des corollaires.

Preuve du Corollaire 1. Le sens réciproque $\Leftarrow$ est clair : si Γ_0 n'est pas satisfaisable, alors aucun ensemble Γ contenant Γ_0 ne sera satisfaisable.
Le sens direct $\Rightarrow$ se prouve en utilisant le théorème de compacité qui affirme :

$$\forall \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ satisfaisable} \quad \rightarrow \quad \Gamma \text{ satisfaisable}$$

(On ne considère que les parties finies Γ_0.) La contraposition est donc :

$$\Gamma \text{ non satisfaisable} \quad \rightarrow \quad \text{non } (\forall \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ satisfaisable})$$

Ce qui s'écrit :

$$\Gamma \text{ insatisfaisable} \quad \rightarrow \quad \exists \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ insatisfaisable}$$

□

Preuve du Corollaire 2. On va utiliser « $\Gamma \vDash \mathcal{Q}$ si et seulement si $\Gamma \cup \{\neg \mathcal{Q}\}$ est insatisfaisable ».
Le sens réciproque $\Leftarrow$ est le sens facile : soit Γ_0 fini tel que $\Gamma_0 \vDash \mathcal{Q}$, donc $\Gamma_0 \cup \{\neg \mathcal{Q}\}$ est insatisfaisable. Alors, quel que soit l'ensemble Γ contenant Γ_0, on a à fortiori que $\Gamma \cup \{\neg \mathcal{Q}\}$ est insatisfaisable. Ainsi $\Gamma \vDash \mathcal{Q}$.
Prouvons le sens direct $\Rightarrow$ à l'aide du Corollaire 1. Supposons $\Gamma \vDash \mathcal{Q}$, donc $\Gamma \cup \{\neg \mathcal{Q}\}$ est insatisfaisable. D'après le Corollaire 1, il existe un ensemble fini $\Delta_0 \subset \Gamma \cup \{\neg \mathcal{Q}\}$ tel que Δ_0 soit insatisfaisable.

- Si $\neg \mathcal{Q} \in \Delta_0$ alors on écrit $\Delta_0 = \Gamma_0 \cup \{\neg \mathcal{Q}\}$ avec $\Gamma_0 \subset \Gamma$ fini. Ainsi $\Gamma_0 \cup \{\neg \mathcal{Q}\}$ est insatisfaisable, ce qui équivaut à $\Gamma_0 \vDash \mathcal{Q}$.
- Si $\neg \mathcal{Q} \notin \Delta_0$, alors on pose juste $\Gamma_0 = \Delta_0$ qui est insatisfaisable. À fortiori $\Gamma_0 \cup \{\neg \mathcal{Q}\}$ est insatisfaisable, donc $\Gamma_0 \vDash \mathcal{Q}$.

Dans les deux cas, on a prouvé $\Gamma_0 \vDash \mathcal{Q}$. □

3. Preuve

Le sens direct $\Rightarrow$ du théorème de compacité est clair : en effet, si Γ est satisfaisable, alors toute partie $\Gamma_0 \subset \Gamma$ (finie ou pas) est satisfaisable. On se concentre donc sur la preuve de la réciproque : si Γ est finiment satisfaisable, alors Γ est satisfaisable.

3.1. Partie 1

Soit Γ un ensemble de formules finiment satisfaisable. Dans un premier temps nous allons construire un ensemble Δ tel que :

1. $\Gamma \subset \Delta$,
2. quelle que soit la formule $\mathcal{Q}$ de l'ensemble Form de toutes les formules, soit $\mathcal{Q} \in \Delta$, soit $\neg \mathcal{Q} \in \Delta$,
3. Δ est finiment satisfaisable.

On va construire par récurrence des ensembles Δ_n finiment satisfaisables, de plus en plus gros. L'ensemble Δ sera la réunion de tous les Δ_n. Commençons par définir :

$$\Delta_0 = \Gamma$$

Par hypothèse Δ_0 est finiment satisfaisable.

Nous avons supposé que l'ensemble Var des variables propositionnelles était un ensemble dénombrable, donc par la Proposition 1, l'ensemble Form de toutes les formules est aussi un ensemble dénombrable. On peut donc en donner une énumération, indexée par des entiers :

$$\text{Form} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots\}$$

Montrons d'abord comment on construit Δ_1. On distingue deux cas :

- Si $\Delta_0 \cup \{\mathcal{Q}_1\}$ est finiment satisfaisable, on pose simplement :
$$\Delta_1 = \Delta_0 \cup \{\mathcal{Q}_1\}$$
- Sinon, on pose :
$$\Delta_1 = \Delta_0 \cup \{\neg\mathcal{Q}_1\}$$
Mais alors, dans ce cas aussi, Δ_1 est finiment satisfaisable, d'après le lemme ci-dessous.

Donc dans tous les cas, Δ_1 est finiment satisfaisable.

Lemme 1.
Soit Σ un ensemble finiment satisfaisable et soit $\mathcal{Q}$ une formule quelconque. Alors $\Sigma \cup \{\mathcal{Q}\}$ ou $\Sigma \cup \{\neg\mathcal{Q}\}$ est finiment satisfaisable.

Démonstration. Supposons par l'absurde que $\Sigma \cup \{\mathcal{Q}\}$ n'est pas finiment satisfaisable et que $\Sigma \cup \{\neg\mathcal{Q}\}$ n'est pas finiment satisfaisable non plus. Ainsi :

- il existe un ensemble fini $\Sigma_1 \cup \{\mathcal{Q}\}$ insatisfaisable (avec $\Sigma_1 \subset \Sigma$),
- et il existe un ensemble fini $\Sigma_2 \cup \{\neg\mathcal{Q}\}$ insatisfaisable (avec $\Sigma_2 \subset \Sigma$).

Notons alors $\Sigma_0 = \Sigma_1 \cup \Sigma_2$. C'est un ensemble fini, inclus dans Σ. Par hypothèse, il est donc satisfaisable. Soit ν une valuation qui satisfait Σ_0. Si ν satisfait $\mathcal{Q}$, alors $\Sigma_1 \cup \{\mathcal{Q}\}$ est satisfait, ce qui n'est pas possible. Mais si ν ne satisfait pas $\mathcal{Q}$, alors ν satisfait $\neg\mathcal{Q}$ et donc ν satisfait $\Sigma_2 \cup \{\neg\mathcal{Q}\}$, ce n'est pas possible non plus. On a donc obtenu une contradiction. □

Continuons notre construction par récurrence. On suppose que l'on a déjà construit Δ_n finiment satisfaisable, sous la forme :
$$\Delta_n = \Gamma \cup \{\varepsilon_1\mathcal{Q}_1, \ldots, \varepsilon_n\mathcal{Q}_n\}$$
où $\varepsilon_i\mathcal{Q}_i$ désigne $\mathcal{Q}_i$ ou bien $\neg\mathcal{Q}_i$. On définit alors :
$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{\mathcal{Q}_{n+1}\} & \text{s'il est finiment satisfaisable,} \\ \Delta_n \cup \{\neg\mathcal{Q}_{n+1}\} & \text{sinon.} \end{cases}$$
Par le Lemme 1, Δ_{n+1} est finiment satisfaisable.

Définissons alors :
$$\Delta = \bigcup_{n \geqslant 0} \Delta_n$$
Comme on a les inclusions $\Delta_0 \subset \Delta_1 \subset \Delta_2 \subset \cdots$, il faut comprendre Δ comme la limite des Δ_n, lorsque n tend vers $+\infty$. Vérifions maintenant les points annoncés :

1. $\Gamma = \Delta_0 \subset \Delta$,
2. pour toute formule $\mathcal{Q}_n$, on a soit $\mathcal{Q}_n \in \Delta_n \subset \Delta$, soit $\neg\mathcal{Q}_n \in \Delta_n \subset \Delta$,
3. enfin, Δ est finiment satisfaisable. En effet, soit un ensemble fini $\Gamma_0 \subset \Delta$, alors il existe $n \geqslant 0$ tel que $\Gamma_0 \subset \Delta_n$. Mais alors Γ_0 est satisfaisable, car Δ_n est finiment satisfaisable.

3.2. Partie 2

Nous allons maintenant démontrer que toutes les formules $\mathcal{P}$ de Δ sont satisfaisables simultanément. Nous allons même préciser la valuation ν qui les réalise. Comme $\Gamma \subset \Delta$, Γ sera donc satisfaisable.

Définissons cette valuation ν. On note $\mathrm{Var} = \{P_i\}_{i \in I}$ l'ensemble des variables propositionnelles. Soit ν la valuation définie par :
$$\begin{cases} \nu(P_i) = \mathbf{1} & \text{si } P_i \in \Delta, \\ \nu(P_i) = \mathbf{0} & \text{sinon.} \end{cases}$$
Une fois définie sur les variables propositionnelles, ν s'étend à toutes les formules (voir la Section 1.3). Soit $\mathcal{P}$ une formule quelconque, nous allons prouver :

Lemme 2.
$\mathcal{P}$ est satisfaite par ν si et seulement si $\mathcal{P} \in \Delta$.

Ainsi, comme $\Gamma \subset \Delta$, toute formule $\mathcal{P} \in \Gamma$ est satisfaite par ν, et donc Γ est satisfaisable car satisfait par ν.
Il nous reste à prouver le lemme.

Démonstration. La preuve se fait par récurrence sur la hauteur de $\mathcal{P}$. Pour simplifier la preuve, on suppose que les seuls connecteurs utilisés sont $\neg$ et $\wedge$. C'est possible sans perte de généralité d'après le chapitre « Quels connecteurs sont utiles ? »

- Si $\mathcal{P}$ est l'une des variables propositionnelles P_i alors par définition de ν, ν satisfait $\mathcal{P}$ si et seulement si $\mathcal{P} \in \Delta$.
- Supposons que l'assertion du lemme soit vraie pour la formule $\mathcal{P}$; prouvons qu'elle est encore vraie pour $\neg\mathcal{P}$.

$$\begin{aligned}
(\neg\mathcal{P}) \text{ satisfait par } \nu &\iff \nu(\neg\mathcal{P}) = \mathbf{1} \\
&\iff \nu(\mathcal{P}) = \mathbf{0} \\
&\iff \text{non}(\mathcal{P} \text{ satisfait par } \nu) \\
&\iff \text{non}(\mathcal{P} \in \Delta) \\
&\iff \mathcal{P} \notin \Delta \\
&\iff \neg\mathcal{P} \in \Delta
\end{aligned}$$

  Pour la dernière ligne, il faut se souvenir que par la propriété (2) de Δ, on sait que $\mathcal{P}$ ou $\neg\mathcal{P}$ appartient à Δ.
- Supposons que l'assertion du lemme soit vraie pour la formule $\mathcal{P}$ et la formule $\mathcal{Q}$; prouvons qu'elle est encore vraie pour $\mathcal{P} \wedge \mathcal{Q}$.

  Sens direct $\Rightarrow$. Si $\mathcal{P} \wedge \mathcal{Q}$ est satisfait par ν, montrons que $\mathcal{P} \wedge \mathcal{Q} \in \Delta$.
  Comme ν satisfait $\mathcal{P} \wedge \mathcal{Q}$, alors ν satisfait $\mathcal{P}$, donc $\mathcal{P} \in \Delta$ et ν satisfait $\mathcal{Q}$, donc $\mathcal{Q} \in \Delta$.
  Par l'absurde, supposons que $\mathcal{P} \wedge \mathcal{Q} \notin \Delta$. Alors, par la propriété (2) de Δ, on a que $\neg(\mathcal{P} \wedge \mathcal{Q}) \in \Delta$. On considère alors l'ensemble $\{\mathcal{P}, \mathcal{Q}, \neg(\mathcal{P} \wedge \mathcal{Q})\}$ qui est donc un ensemble fini non satisfaisable de formules (aucune valuation ne peut satisfaire simultanément $\mathcal{P}$, $\mathcal{Q}$ et $\neg(\mathcal{P} \wedge \mathcal{Q})$). Cela contredit le fait que Δ est finiment satisfaisable. Conclusion : $\mathcal{P} \wedge \mathcal{Q} \in \Delta$.

  Sens réciproque $\Leftarrow$. Si $\mathcal{P} \wedge \mathcal{Q} \in \Delta$, montrons que $\mathcal{P} \wedge \mathcal{Q}$ est satisfait par ν.
  Par l'absurde, supposons que $\mathcal{P} \notin \Delta$. Alors $\neg\mathcal{P} \in \Delta$ (toujours par la propriété (2) de Δ). Ainsi $\{\mathcal{P} \wedge \mathcal{Q}, \neg\mathcal{P}\} \subset \Delta$ est un ensemble fini, mais qui n'est bien sûr pas satisfaisable. Cela contredit de nouveau le fait que Δ est finiment satisfaisable. Bilan : on a $\mathcal{P} \in \Delta$, donc par hypothèse de récurrence $\nu(\mathcal{P}) = \mathbf{1}$.
  On prouve de même que $\mathcal{Q} \in \Delta$ et donc $\nu(\mathcal{Q}) = \mathbf{1}$. Mais alors, $\nu(\mathcal{P} \wedge \mathcal{Q}) = \mathbf{1}$ (par la construction de la valuation) et donc $\mathcal{P} \wedge \mathcal{Q}$ est satisfaite par ν.

□

Remarque. Si on autorise les autres connecteurs classiques $\vee$, $\rightarrow$, $\leftrightarrow$, alors on peut prouver de la même façon que $\mathcal{P} \vee \mathcal{Q}$ est satisfaite par ν si et seulement si $\mathcal{P} \vee \mathcal{Q} \in \Delta$, etc. (À faire dans les exercices.)

4. $\{0,1\}^{\mathbb{N}}$ est compact

Pourquoi le théorème de compacité s'appelle-t-il ainsi ? Nous faisons ici le lien avec la notion d'ensemble compact. La lecture de cette section est destinée à ceux qui ont déjà une connaissance de la topologie, dont les notions ne seront pas rappelées ici.

4.1. Topologie de $\{0,1\}^{\mathbb{N}}$

- Soit $\mathbb{B} = \{\mathbf{0}, \mathbf{1}\}$ muni de la topologie discrète, c'est-à-dire que $\{\mathbf{0}\}$ et $\{\mathbf{1}\}$ sont à la fois des ensembles ouverts et fermés.
- $\mathbb{B}^{\mathbb{N}} = \{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}}$ est un espace produit. Il peut être vu comme l'ensemble $\{\nu : \mathbb{N} \to \{\mathbf{0}, \mathbf{1}\}\}$ des fonctions qui, à un entier i, associent une valeur $\mathbf{0}$ ou $\mathbf{1}$. Autrement dit comme l'ensemble des valuations qui, à chaque variable propositionnelle P_i, associent une valeur Vrai/Faux (ici $\text{Var} = \{P_i\}_{i\in\mathbb{N}}$ est dénombrable).
- On munit $\{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}}$ de la topologie produit, c'est-à-dire, par définition, la topologie la moins fine rendant continues les projections $\pi_i : \{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}} \to \{\mathbf{0}, \mathbf{1}\}$.
- Ainsi, pour i fixé, $\pi_i^{-1}(\{\mathbf{1}\}) = \{\nu : \mathbb{N} \to \{\mathbf{0}, \mathbf{1}\} \mid \nu(i) = \mathbf{1}\}$ est un fermé en tant qu'image réciproque d'un ensemble fermé par une application continue. C'est aussi un ouvert en tant qu'image réciproque d'un ensemble ouvert, ce qui fait que son complément est un ensemble fermé.
- Soit $\mathcal{P}$ une formule. Notons $V(\mathcal{P}) = \{\nu \in \{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}} \mid \nu(\mathcal{P}) = \mathbf{1}\}$. Cet ensemble $V(\mathcal{P})$ est un fermé. La preuve se fait par récurrence sur la hauteur de $\mathcal{P}$ (un peu comme dans la seconde partie de la preuve du théorème de compacité ; voir les exercices).
- Une formule $\mathcal{P}$ est satisfaisable si et seulement si $V(\mathcal{P})$ est non vide.

4.2. Un produit de compacts est compact

Théorème 2 (Théorème de Tychonoff)**.**
Un produit (quelconque) d'ensembles compacts est compact.

Le produit peut être fini ou infini (dénombrable ou pas).

Corollaire 3.
$\{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}}$ *est un ensemble compact.*

En effet, $\{\mathbf{0}, \mathbf{1}\}$ est compact car c'est un ensemble fini.
Voici une propriété importante des compacts.

Proposition 2.
Soient F_j, $j \in J$, des ensembles fermés dans un ensemble compact E. Si $\bigcap_{j\in J} F_j = \varnothing$, alors il existe un ensemble fini $J_0 \subset J$ tel que $\bigcap_{j\in J_0} F_j = \varnothing$.

C'est l'équivalent de la version plus connue pour les ouverts : « De tout recouvrement d'un compact par des ouverts, on peut en extraire un sous-recouvrement fini. »

4.3. Preuve du théorème de compacité

Soit $\Gamma = \{\mathcal{P}_0, \mathcal{P}_1, \ldots\}$ un ensemble de formules finiment satisfaisable.
On note $F_j = V(\mathcal{P}_j)$, qui est un ensemble fermé de $\{\mathbf{0}, \mathbf{1}\}^{\mathbb{N}}$. Notons que F_j est non vide car $\mathcal{P}_j$ est satisfaisable (puisque $\{\mathcal{P}_j\}$ est un sous-ensemble fini de Γ). Plus généralement, si $\Gamma_0 = \{\mathcal{P}_j\}_{j\in J_0}$ est un ensemble fini de formules inclus dans Γ, alors il existe une valuation ν qui satisfait simultanément tous les $\mathcal{P}_j$. Donc $\nu \in F_j$ pour tout $j \in J_0$. Autrement dit :

$$\bigcap_{j\in J_0} F_j \neq \varnothing$$

Ainsi, chaque intersection finie de fermés est non vide. Par contraposition de la Proposition 2, on a donc :

$$\bigcap_{j\in\mathbb{N}} F_j \neq \varnothing$$

Cela signifie qu'il existe $v : \mathbb{N} \to \{\mathbf{0},\mathbf{1}\}$ tel que $v(\mathcal{P}_j) = \mathbf{1}$, quel que soit $j \in \mathbb{N}$. Ainsi, v satisfait Γ. On a bien prouvé que Γ est satisfaisable.

Exercices

Formules, formules satisfaisables, conséquence logique

Exercice 1. On considère $\mathrm{Var} = \{P,Q,R\}$ et $\mathrm{Conn} = \{\neg,\wedge\}$. Énumérer toutes les formules de hauteur 0, puis 1. Combien y a-t-il de formules de hauteur 2 ? Mêmes questions avec $\mathrm{Var} = \{P,Q\}$ et $\mathrm{Conn} = \{\neg,\wedge,\vee,\to,\leftrightarrow\}$.

Exercice 2. Montrer que deux formules sont logiquement équivalentes exactement lorsque chacune est conséquence logique de l'autre.

$$\mathcal{P} \equiv \mathcal{Q} \quad \text{ssi} \quad \mathcal{P} \vDash \mathcal{Q} \text{ et } \mathcal{Q} \vDash \mathcal{P}$$

Exercice 3. Trouver toutes les valuations définies sur $\{P_i\}_{i\in\mathbb{N}}$ qui satisfont l'ensemble infini de formules :

$$\Gamma = \{P_0 \to P_1, P_1 \to P_2, P_2 \to P_3, \ldots\}$$

Théorème de compacité

Exercice 4. Pour les variables $\{P_i\}_{i\in\mathbb{N}}$, on définit pour $n \geqslant 0$, la formule $\mathcal{P}_n = \neg(P_n \wedge P_{n+1})$. Soit $\Gamma = \{\mathcal{P}_n \mid n \geqslant 0\}$.

1. Montrer que chaque partie finie de Γ est satisfaisable.
2. En déduire que Γ est satisfaisable.
3. Caractériser toutes les valuations définies sur $\{P_i\}_{i\in\mathbb{N}}$ qui satisfont Γ.

Exercice 5. Pour chaque entier $n \in \mathbb{N}$, on définit la formule :

$$\mathcal{P}_n = (P_n \vee P_{n+1}) \wedge (\neg P_{n+2}).$$

On considère l'ensemble $\Gamma = \{\mathcal{P}_n \mid n \geqslant 0\}$.

1. Pour $n \geqslant 0$ fixé, montrer que $\{\mathcal{P}_n\}$ est satisfaisable.
2. Pour $n \geqslant 0$ fixé, montrer que $\{\mathcal{P}_n, \mathcal{P}_{n+1}\}$ est satisfaisable.
3. Γ est-il satisfaisable ?

Exercice 6. En partant du Corollaire 2, supposé admis, donner une preuve du théorème de compacité.

Preuve

Exercice 7. Prolonger la preuve du Lemme 2 afin de montrer directement que :

$$v \text{ satisfait } \mathcal{P} \vee \mathcal{Q} \quad \text{ssi} \quad \mathcal{P} \vee \mathcal{Q} \in \Delta$$

$\{0,1\}^{\mathbb{N}}$ est compact

Exercice 8. Soit $\mathcal{P}$ une formule et $V(\mathcal{P}) = \{v \in \{\mathbf{0},\mathbf{1}\}^{\mathbb{N}} \mid v(\mathcal{P}) = \mathbf{1}\}$. Montrer que $V(\mathcal{P})$ est un ensemble fermé de $\{\mathbf{0},\mathbf{1}\}^{\mathbb{N}}$. On le justifiera d'abord pour les variables propositionnelles, puis on le prouvera par récurrence sur la hauteur de $\mathcal{P}$ en s'inspirant de la preuve du Lemme 2.

Deuxième partie

logique du premier ordre

Chapitre 5

Logique du premier ordre

On considère un langage avec des variables et les quantificateurs « pour tout » et « il existe ». On enrichit ce langage avec des prédicats et des fonctions.

1. Motivation

1.1. Quantificateurs

La logique Vrai/Faux des chapitres précédents est au final assez limitée et ne permet pas d'exprimer les phrases naturelles d'un cours de mathématiques. Dans ce chapitre, on considère un langage plus puissant : la logique du premier ordre. Elle permet d'écrire les phrases suivantes :

- « $\forall n \in \mathbb{Z} \quad n^2 \geqslant 0$ » : « pour tout entier relatif, son carré est positif ».
- « $\exists z \in \mathbb{C} \quad z^2 = -1$ » : « il existe un nombre complexe dont le carré vaut -1 ».
- « $\forall g \in G \quad \exists g' \in G \quad g \circ g' = e$ » : « tout élément de G admet un inverse », pour un groupe $(G, \circ)$ d'élément neutre e.

Avec ce langage, on peut définir des notions compliquées avec des nuances subtiles. Par exemple, soit $f : \mathbb{R} \to \mathbb{R}$ une fonction.

- La continuité sur $\mathbb{R}$ se définit par :
$$\forall x \in \mathbb{R} \quad \forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall y \in \mathbb{R} \qquad |x - y| < \delta \to |f(x) - f(y)| < \varepsilon$$
- Alors que la continuité uniforme se définit par :
$$\forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \qquad |x - y| < \delta \to |f(x) - f(y)| < \varepsilon$$

1.2. Nouveautés

La principale nouveauté, c'est bien sûr l'apparition des quantificateurs « $\forall$ » et « $\exists$ ». En une phrase « $\forall n \in \mathbb{N} \quad n^2 \geqslant 0$ », on exprime une infinité de vérités. Avec la logique Vrai/Faux on ne saurait pas prouver une telle affirmation, car on devrait considérer le cas $n = 0$, puis $n = 1$, etc., donc avec un nombre infini d'étapes. Ici on considère la phrase « $\forall n \in \mathbb{N} \quad n^2 \geqslant 0$ » comme une seule affirmation à prouver.

Le second intérêt de la logique du premier ordre est qu'elle s'applique à des situations variées. En logique propositionnelle, on considérait des formules et la satisfaction consistait à remplacer les variables par des valeurs V/F. Ici on souhaite toujours décider si une formule est vraie ou fausse, mais les variables correspondent à des objets plus généraux. Par exemple considérons la formule :

$$\big(\forall x \quad P(x)\big) \vee \big(\exists x \quad \neg P(x)\big)$$

Elle s'applique avec des x dans un ensemble quelconque, mais aussi avec n'importe quel prédicat $P(x)$. En français, elle signifie « une propriété est vraie pour tous les éléments, ou bien il existe un élément pour lequel elle n'est pas vraie ». Voici ce qui donne cette formule dans trois structures différentes :

- $\big(\forall x \in \mathbb{R} \quad x^2 \neq 0\big) \vee \big(\exists x \in \mathbb{R} \quad x = 0\big)$

- $\big(\forall n \in \mathbb{N} \quad n+1 > n\big) \vee \big(\exists n \in \mathbb{N} \quad n+1 \leqslant n\big)$
- $\big(\forall g \in G \quad g \circ g = e\big) \vee \big(\exists g \in G \quad \neg(g \circ g = e)\big)$, dans un groupe G.

Les langages du premier ordre s'adaptent à une grande variété de structures (ensembles, groupes, corps, espaces vectoriels…) et aux démonstrations qui vont avec.

1.3. Limitations

La logique du premier ordre est puissante, cependant il y a certaines choses qu'il n'est pas possible d'exprimer avec ces langages. Le plus flagrant est qu'il n'est pas possible de trouver une formule pour décider si un ensemble est fini ou non (ce qui entraîne par exemple qu'on ne peut pas définir la notion de dimension d'un espace vectoriel). Travailler en logique du premier ordre signifie que l'on considère des éléments dans un ensemble E, noté « $x \in E$ », mais on ne peut pas travailler avec l'ensemble des parties $A \subset E$. Ainsi, la phrase « toute partie non vide admet un plus petit élément » (qui est la définition d'un ensemble *bien ordonné*) est une phrase exprimée dans la *logique du second ordre* car elle porte sur les sous-ensembles et pas sur les éléments. En conclusion, la logique du premier ordre décrit bien les propriétés locales des objets, mais moins bien les propriétés globales liées à l'infini.

2. Langage

Il faut comprendre un langage de la logique du premier ordre comme un cadre général commun qu'on spécialisera ensuite à chaque situation (les entiers, les réels, les groupes…).

2.1. Éléments de langage

Les formules de la logique du premier ordre sont construites à partir de langages composés des éléments suivants.

Symboles logiques

- ***variables*** : x, y, z, $x_1, x_2, \ldots$
- parenthèses : (et)
- connecteurs : $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$
- ***quantificateurs*** : $\forall$ (pour tout), $\exists$ (il existe)

Symboles non logiques

- ***constantes*** : a, b, c, $c_1, c_2, \ldots$
- ***prédicats*** : $P(x)$, $P(x,y)$,…, $P(x_1, \ldots, x_n)$; dont l'égalité $x = y$
- ***fonctions*** : $f(x)$, $f(x,y)$,…, $f(x_1, \ldots, x_n)$

Nous revenons en détail ci-dessous sur les nouvelles notions. Dans un premier temps on va considérer des langages sans fonctions ; on les ajoutera à la fin de ce chapitre.

2.2. Symboles logiques

Le langage est beaucoup plus riche que celui de la logique Vrai/Faux. On retrouve bien sûr les connecteurs logiques (et les parenthèses) qui joueront ici le même rôle qu'en logique propositionnelle. Détaillons les nouveautés.

- **Quantificateur universel** $\forall$**.** C'est un A inversé, introduit par Gerhard Gentzen en 1935, comme initiale de *all* ou *alle*. Il se lit « pour tout » ou « quel que soit ».
- **Quantificateur existentiel** $\exists$**.** C'est un E inversé, introduit par Giuseppe Peano à la fin du XIX^{e} siècle. Il se lit « il existe » ou « il y a au moins un ».
- **Variables.** Elles servent à parler des objets manipulés sans les nommer individuellement. Ainsi « $\forall x \; P(x)$ » ou « $\exists x \; P(x)$ » va s'appliquer à tous les x (ou à un x), mais il n'y a pas vraiment un

objet nommé x. On parle de ***variable liée*** (ou *variable muette*). Ainsi « $\forall y\ P(y)$ » aurait la même signification.
Note : pour les variables, on réserve les lettres de la fin de l'alphabet x, y, z, avec des indices si besoin.

2.3. Symboles non logiques

C'est aussi une nouveauté de la logique du premier ordre. On veut des constructions et des théories qui s'appliquent à de nombreuses situations. Les constantes, les prédicats et les fonctions sont des éléments comme des coquilles vides, qu'il faudra préciser à chaque application.

- **Constantes.** Notées $a, b, c, c_1, c_2, \ldots$ ce sont des symboles du langage qui vont servir à distinguer certains éléments. On peut par exemple définir un langage adapté à la notion de groupe avec une constante c, qui jouera le rôle de l'élément neutre e, 0, 1,... selon le groupe considéré. Lors de la définition du langage, on décide du nom et du nombre des constantes.
- **Prédicats.** Un prédicat est un peu comme une super-proposition de la logique Vrai/Faux, cela renvoie une valeur V/F, mais en fonction de la valeur d'un ou plusieurs paramètres. Leur symbole dans le langage sera souvent P, Q ou R.
  — Prédicat unaire P, noté $P(x)$: « est-ce que x a la propriété P ? » (cela peut être vrai ou faux selon la valeur affectée à x).
  Par exemple un tel prédicat pourra désigner « $x^2 \geqslant x$ » ou « $x^2 = 1$ » ou « $x^5 + x^3 + 1 = 0$ » ...
  Cela peut être vrai ou faux en fonction de x.
  — Prédicat binaire P, noté $P(x, y)$: « est-ce que (x, y) a la propriété P ? », mais qui peut aussi être compris comme « est-ce que x est en relation P avec y ? ».
  Par exemple un prédicat binaire peut désigner « $x < y$ » ou « $x^2 + y^2 = 1$ »...
  — L'***égalité*** « $x = y$ » est un prédicat binaire spécial. Sauf mention contraire (et rare) on supposera toujours que l'égalité est présente dans le langage.
  — Prédicat n-aire $P(x_1, \ldots, x_n)$ (on parle aussi de prédicat à n places).
  Encore une fois, ce qu'est concrètement chaque prédicat devra être précisé dans chaque structure. Au niveau du langage c'est juste un symbole P, mais on se permettra de le noter $P(x)$ ou $P(x_1, \ldots, x_n)$ afin de clarifier le nombre de places.
- **Fonctions.** À la fin du chapitre on ajoutera des fonctions au langage. Par exemple $f(x)$ (qui peut représenter $f(x) = 2x + 3$ ou $f(x) = 1/x, \ldots$), $f(x, y)$ (pour $f(x, y) = x + y$ ou $f(x, y) = x \times y, \ldots$).
  Les fonctions ajoutent une certaine technicité à la définition de la notion de formule, donc pour l'instant nos langages n'auront pas de fonctions.

3. Formules

3.1. Définitions

Comme le langage est riche, il y a un vocabulaire spécifique qu'il faut bien maîtriser.
Soit $\mathcal{L}$ un langage sans fonctions.

Définition.
On appelle ***terme*** d'un langage les variables ou les constantes.

Remarque : on modifiera cette définition pour les langages avec fonctions (voir en fin de chapitre).

Définition.
On appelle ***formule atomique*** un mot de la forme $P(t_1, \ldots, t_n)$ où P est un prédicat n-aire et $t_1, \ldots, t_n$ sont des termes.

Définition.
L'ensemble des ***formules*** $\mathrm{Form}(\mathcal{L})$ du langage est construit selon les règles suivantes :

- les formules atomiques sont des formules,
- si $\mathcal{P}$ et $\mathcal{Q}$ sont des formules alors $\neg\mathcal{P}$, $(\mathcal{P}\wedge\mathcal{Q})$, $(\mathcal{P}\vee\mathcal{Q})$, $(\mathcal{P}\to\mathcal{Q})$, $(\mathcal{P}\leftrightarrow\mathcal{Q})$ sont aussi des formules,
- si x est une variable et $\mathcal{P}$ est une formule alors
$$\forall x\ \mathcal{P} \qquad \text{et} \qquad \exists x\ \mathcal{P}$$
sont aussi des formules,
- aucun autre mot n'est une formule.

3.2. Remarques

- L'écriture « $\forall x\ \mathcal{P}$ » est un mot du langage, et dans ce langage x est une variable abstraite, pas un élément d'un ensemble. Lorsque l'on interprétera ultérieurement la formule, on définira un domaine E, par exemple $\mathbb{N}$, $\mathbb{R}$, G, … et alors la formule deviendra « $\forall x\in\mathbb{N}\ \ \mathcal{P}$ » ou « $\forall x\in\mathbb{R}\ \ \mathcal{P}$ », etc.
- Le cas intéressant est bien sûr celui où la formule $\mathcal{P}$ dépend effectivement de la variable x. On se permettra souvent de noter une formule $\mathcal{P}(x)$ pour montrer sa dépendance en la variable x. Ainsi on écrira plutôt « $\forall x\ \ \mathcal{P}(x)$ » ou « $\forall x\ \exists y\ \ \mathcal{P}(x,y)$ »…
- Comme d'habitude on s'autorisera à ajouter et supprimer des parenthèses pour clarifier les formules. Par exemple on écrira : « $(\forall x\ \mathcal{P})\to(\exists y\ \mathcal{Q})$ » au lieu de la formule officielle « $(\forall x\ \mathcal{P}\to\exists y\ \mathcal{Q})$ »

La définition de formule donnée ci-dessus était la définition inductive (« par le haut ») de $\mathrm{Form}(\mathcal{L})$. On peut aussi en donner une définition récursive (« par le bas »).
On définit $\mathrm{Form}(\mathcal{L})=\bigcup_{n\geqslant 0}\mathrm{Form}_n(\mathcal{L})$, avec :

$$\mathrm{Form}_0(\mathcal{L}) \quad \text{est l'ensemble des formules atomiques}$$

puis, pour $n\geqslant 0$:

$$\begin{aligned}\mathrm{Form}_{n+1}(\mathcal{L}) = \mathrm{Form}_n(\mathcal{L})&\\ \cup\ &\{\neg\mathcal{P}\mid \mathcal{P}\in\mathrm{Form}_n(\mathcal{L})\}\\ \cup\ &\{(\mathcal{P}\wedge\mathcal{Q})\mid \mathcal{P},\mathcal{Q}\in\mathrm{Form}_n(\mathcal{L})\}\\ \cup\ &\cdots\\ \cup\ &\{\forall x_k\ \mathcal{P}\mid x_k \text{ variable}, \mathcal{P}\in\mathrm{Form}_n(\mathcal{L})\}\\ \cup\ &\{\exists x_k\ \mathcal{P}\mid x_k \text{ variable}, \mathcal{P}\in\mathrm{Form}_n(\mathcal{L})\}\end{aligned}$$

où l'union porte bien sûr aussi sur les connecteurs $\vee$, $\to$, $\leftrightarrow$ et où on a noté $x_1,x_2,\ldots$ l'ensemble des variables de $\mathcal{L}$.

4. Exemples

Prenons un peu d'avance sur le prochain chapitre avec des exemples de structures afin de mieux comprendre les notions précédentes.

4.1. Avec des entiers

On souhaite modéliser les entiers naturels avec un zéro et une relation d'ordre. On définit un langage $\mathcal{L}=(c,P)$ avec une constante c et un prédicat binaire P, noté aussi $P(x,y)$ (il y a aussi par défaut le prédicat d'égalité $x=y$). En toute rigueur on devrait aussi préciser le nom des variables, mais on utilisera la convention de nommage $x,y,\ldots$ partout.

Voici une formule du langage :

$$\forall x \ \exists y \quad P(x,y)$$

Pour pouvoir utiliser ce langage, on précise un domaine, par exemple ici $E = \mathbb{N}$. Ensuite il faut définir tous les symboles non logiques :

- la constante : $c = 0$,
- le prédicat : $P(x,y)$: « $x < y$ ».

Les symboles logiques, eux, ne doivent pas être spécifiés : la variable x reste une variable, le connecteur $\wedge$ reste le ET, etc. On a ainsi défini une ***structure***, notée $\mathcal{S}$. Dans cette structure, la formule précédente s'interprète en :

$$\forall x \in \mathbb{N} \ \exists y \in \mathbb{N} \quad x < y,$$

c'est-à-dire « pour tout entier, il existe un entier encore plus grand ».
Pour le même langage $\mathcal{L}$, on peut faire d'autres choix de structure et la même formule s'interprète différemment.

Voici quelques exemples pour illustrer le vocabulaire lié aux formules avec à chaque fois un mot du langage et son interprétation dans la structure.
Commençons par les termes : il n'y a que la constante c et les variables $x, y, \ldots$ Dans la structure on remplace c par 0.

Terme dans $\mathcal{L}$	Interprétation dans $\mathcal{S}$
c	0
x	x

Les formules atomiques sont formées par un prédicat. Ici, seul le prédicat P est défini et est interprété par $x < y$ dans la structure. On a en plus toujours le prédicat d'égalité « $x = y$ ».

Formule atomique dans $\mathcal{L}$	Interprétation dans $\mathcal{S}$
$P(c,x)$	$0 < x$
$P(c,c)$	$0 < 0$
$P(x,y)$	$x < y$
$P(y,x)$	$y < x$
$x = c$	$x = 0$

Pour les formules, on autorise les connecteurs et les quantificateurs. Une formule « $\forall x \ \mathcal{P}$ » s'interprète dans la structure en « $\forall x \in \mathbb{N} \ \mathcal{P}$ ».

Formule dans $\mathcal{L}$	Interprétation dans $\mathcal{S}$
$\forall x \quad P(c,x)$	$\forall x \in \mathbb{N} \quad 0 < x$
$\exists y \quad x = y$	$\exists y \in \mathbb{N} \quad x = y$
$\forall x \quad P(c,x) \vee (x = c)$	$\forall x \in \mathbb{N} \quad (0 < x) \vee (x = 0)$
$\forall x \ \forall y \quad P(x,y) \to (\exists z \quad P(x,z) \wedge P(z,y))$	$\forall x \in \mathbb{N} \ \forall y \in \mathbb{N} \quad (x < y) \to (\exists z \in \mathbb{N} \quad (x < z) \wedge (z < y))$

La dernière formule s'interprète avec les entiers sous la forme « entre deux entiers distincts, on peut trouver un entier strictement compris entre les deux ». C'est clairement faux (par exemple, il n'y a aucun entier strictement compris entre 7 et 8). Ce serait vrai pour une autre structure où le domaine est l'ensemble des réels $\mathbb{R}$ (entre deux réels distincts, il existe même une infinité de réels).

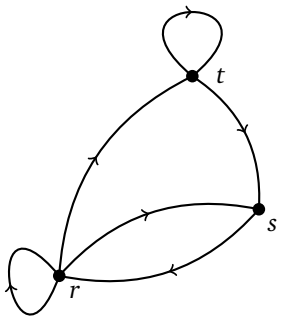
4.2. Avec des graphes

On veut modéliser des graphes (comme par exemple sur le dessin ci-dessous). On définit un langage $\mathcal{L}$ avec le prédicat binaire R (et aucune constante).

Dans ce langage, les seuls termes sont les variables, x et y par exemple. Les formules atomiques sont $R(x,x)$, $R(x,y)$, $R(y,x)$, $R(y,y)$ et aussi $x=x$, $x=y$, $y=x$, $y=y$. Voici un exemple de formule :

$$\forall x \ \exists y \quad R(x,y)$$

Nous avons notre langage. Comment modéliser un graphe ? Voici le graphe orienté que l'on souhaite modéliser :



Il y a trois sommets, on va donc définir une structure dont le domaine est l'ensemble des trois éléments $E=\{r,s,t\}$. Dans la structure, c'est le prédicat qui décrit les arêtes : on interprète le prédicat R en disant que $R(x,y)$ est vrai s'il existe une arête allant de x vers y (on dit que x est en relation avec y). Ainsi pour notre exemple, on définit le prédicat dans la structure par :

$$R(r,s)=\mathrm{V}, \quad R(s,t)=\mathrm{F}, \quad R(t,s)=\mathrm{V}, \ldots$$

La formule précédente du langage $\mathcal{L}$ s'interprète dans cette structure comme :

$$\forall x \in \{r,s,t\} \quad \exists y \in \{r,s,t\} \quad R(x,y)$$

Autrement dit, « pour tout sommet x, il existe un sommet y pour lequel on a une arête de x vers y », en d'autres termes « de tout sommet part au moins une arête », ce qui est vrai pour le graphe considéré. Voici des phrases du langage. À vous de déterminer ce qu'elles signifient pour les graphes et si, pour l'exemple de graphe ci-dessus, elles sont vraies ou fausses.

- $\forall x \ \forall y \quad R(x,y)$
- $\exists x \ \exists y \quad R(x,y)$
- $\exists x \ \forall y \quad R(x,y)$
- $\forall x \ \forall y \quad R(x,y) \vee R(y,x)$
- $\exists x \ \exists y \quad R(x,y) \wedge R(y,x)$

4.3. Il existe exactement un, il existe deux

- « $\exists x \ P(x)$ » demande l'existence d'au moins un x pour lequel $P(x)$ est vérifié. Il peut y en avoir un ou plusieurs.
- Voici comment exprimer qu'il existe exactement un élément qui vérifie $P(x)$:

  $$\big(\exists x \quad P(x)\big) \wedge \big(\forall x \ \forall y \quad (P(x) \wedge P(y) \to x=y)\big)$$

  On exige d'une part l'existence d'un élément, et d'autre part s'il en existe un autre, c'est le même. On n'utilisera pas la notation « $\exists !$ ».
- Voici comment exprimer qu'il existe au moins deux éléments :

  $$\exists x \ \exists y \quad (x \neq y) \wedge P(x) \wedge P(y)$$

  On note $\neg(x=y)$ par $x \neq y$.

5. Langage avec fonctions

5.1. Fonctions

Une ***fonction*** f prend en entrée un ou plusieurs objets et renvoie en sortie un objet. Il ne faut pas confondre les fonctions avec les prédicats : les prédicats prennent aussi des objets en entrée mais renvoient une valeur V/F.
Voici des exemples d'interprétation de fonctions :

- $f(x) = x^2$, fonction à une variable (ici pour les réels),
- $f(x, y) = x\%y$, reste de la division euclidienne de x par y (ici pour les entiers).

Par contre $P(x)$: « $x = 0$ » et $Q(x, y)$: « $x < y$ » sont des prédicats, pas des fonctions.
Dorénavant, on s'autorise l'ajout de fonctions n-aires (ou à n places) comme symboles non logiques du langage. Comme pour les prédicats, on notera souvent abusivement les fonctions par $f(x)$, $f(x, y)$, $f(x_1, \ldots, x_n)$, alors que le symbole du langage est bien seulement f.

5.2. Formules

Avec l'introduction des fonctions, il faut adapter notre définition de la notion de formule.
Soit $\mathcal{L}$ un langage (avec ou sans fonctions).

Définition.
On appelle ***terme*** d'un langage :

- les variables et les constantes,
- et si $t_1, \ldots, t_n$ sont des termes et f un symbole de fonction n-aire, alors $f(t_1, \ldots, t_n)$ est aussi un terme.

Noter que la définition est récursive.
La définition de ***formule atomique*** est inchangée : c'est une expression de la forme $P(t_1, \ldots, t_n)$ où P est un prédicat n-aire et $t_1, \ldots, t_n$ sont des termes (avec notre nouvelle définition).
La définition de ***formule*** est inchangée, c'est la même construction à l'aide des formules atomiques, des connecteurs, des quantificateurs (sauf que bien sûr les termes et les formules atomiques peuvent maintenant contenir des fonctions).

5.3. Exemples

Soit $\mathcal{L}$ le langage formé par :

- des constantes a, b,
- un prédicat binaire P, noté (abusivement) $x < y$,
- une fonction unaire f, une fonction binaire g.
- Exemples de termes :

$$a, \quad y, \quad f(x), \quad g(a, y), \quad g\big(f(x), b\big), \quad f\big(g(y, x)\big), \quad \ldots$$

- Exemples de formules atomiques :

$$x < a, \quad f(x) < f(y), \quad g(a, f(b)) < x, \quad \ldots$$

- Exemples de formules :

$$\neg(x < y), \quad (f(x) < a) \wedge (f(y) < b), \quad \forall x\ f(x) < b, \quad \exists x\ g(x, y) = a,$$
$$\forall x \quad (f(x) < a) \to (\exists y\ g(x, y) < b)$$

Comme auparavant, on anticipe sur la notion de structure afin de rendre concrètes ces fonctions. On peut par exemple se placer sur l'ensemble des réels $\mathbb{R}$ et considérer l'interprétation suivante des symboles non logiques :

- $a = 0$, $b = 1$,
- $P(x, y)$ désigne l'inégalité stricte usuelle $x < y$,
- $f(x) = x^2$ et $g(x, y) = x + y$.
- Exemples de termes dans cette interprétation :

$$x + 1, \quad y^2, \quad x^2 + y, \quad (x + y)^2, \quad x + (y^2 + 1)^2, \quad \dots$$

- Exemples de formules atomiques dans cette interprétation :

$$x < 0, \quad x + y < x^2, \quad x^2 + y^2 = 1, \quad \dots$$

- Exemples de formules dans cette interprétation :

$$(x^2 + y^2 < 1) \to ((x < 1) \wedge (y < 1)), \quad \forall x \;\; x^2 < x + y, \quad (y < 1) \to (\exists x \;\; x^2 = y), \quad \dots$$

Exercices

Langage, Formules, Exemples

Exercice 1. On considère le langage $\mathcal{L}$ formé des constantes a et b, d'un prédicat unaire P et d'un prédicat binaire Q (et en plus, les variables x, y, les connecteurs, les quantificateurs). Dire si les mots suivants sont des termes, des formules atomiques, des formules ?

$$\neg a, \quad xy, \quad (x \wedge a), \quad P(a), \quad \neg Q(x)$$

$$\forall x \; Q(x, y), \quad \exists y \; P(x), \quad \forall a \; \exists Q(a, b)$$

$$\forall x \; \exists y \; P(x \to y), \quad \exists x \; \big((\forall y \; Q(x, y)) \vee (\forall y \; \neg Q(x, y))\big)$$

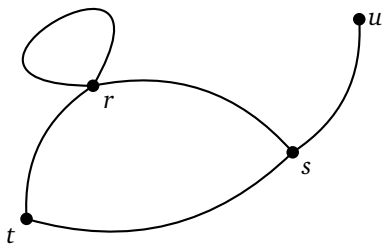
Exercice 2. On considère le langage $\mathcal{L}$ muni d'une constante c et d'un prédicat binaire $P(x, y)$. Dire si les mots suivants sont des termes, des formules atomiques, des formules ? Si oui, donner l'interprétation dans la structure de domaine $\mathbb{N}$, avec $c = 0$ et $P(x, y)$ défini par $x < y$. Faire ensuite l'interprétation dans la structure de domaine $\mathbb{R}$, avec $c = 1$ et $P(x, y)$ défini par $x \leqslant y$.

$$P, \quad c^2, \quad P(c, c), \quad \neg P(c, 0), \quad P(x, P(y, c)), \quad P(x = c) \to x = c$$

$$P(x, c), \quad \exists x, \quad x > c \to \forall y \; P(y, c), \quad \forall y \; \big(P(x, c) \to y = x\big)$$

$$\exists x \vee y \quad P(x, c) \vee P(y, c), \quad \forall x \quad \big((x = c) \leftrightarrow (\exists y \; P(y, c))\big)$$

Exercice 3. On considère un langage avec un prédicat binaire $R(x, y)$. On considère la structure de domaine $E = \{r, s, t, u\}$ et on interprète $R(x, y)$ comme vrai s'il existe une arête (non orientée) entre les sommets x et y d'un graphe G (par exemple comme ci-dessous).



Traduire dans le langage $\mathcal{L}$ les phrases suivantes :

1. Il existe un sommet relié à lui-même.
2. Il existe un sommet relié à tous les autres.
3. Tout sommet est relié à au moins un autre.

4. Il existe un sommet relié à aucun autre.
5. Tous les sommets ne sont pas reliés entre eux.
6. Il existe un seul sommet relié à lui même.

Exercice 4. On considère un langage $\mathcal{L}$ muni d'un prédicat $P(x)$ et de l'égalité $x = y$. Traduire par une formule de $\mathcal{L}$ les phrases suivantes :

1. Aucun élément ne vérifie P.
2. Il existe au plus un élément qui vérifie P.
3. Il existe exactement deux éléments qui vérifient P.
4. Il existe au moins trois éléments qui vérifient P.
5. Tous les éléments sauf un vérifient P.

Langage avec fonctions

Exercice 5. On considère un langage $\mathcal{L}$ ayant les constantes a, b, un prédicat $P(x)$ et deux fonctions $f(x)$ et $g(x, y)$. Dire si les mots suivants sont des termes, des formules atomiques, des formules de $\mathcal{L}$? Si oui, donner l'interprétation dans la structure de domaine $\mathbb{R}$, avec $a = 0$, $b = 1$, $P(x)$ défini par $x \geqslant 0$, $f(x) = \cos(x)$, $g(x, y) = x \times y$.

$$f(x), \quad g(a, b), \quad f(P(a)), \quad \neg P(f(a)), \quad P(g(y, x)), \quad P(g(f(x), f(f(b))))$$

$$P(x) \wedge P(y) \to g(x, y), \quad \exists x\ f(x) = b, \quad \forall y\ P(y) \leftrightarrow P(f(y))$$

$$\forall x \forall b\ P(g(x, b)) \to (P(x) \wedge P(b)), \quad \exists x\ \Big(P(x) \leftrightarrow (\forall y\ (P(y) \to P(g(x, y))))\Big)$$

Chapitre 6

Variables et structures

Notre but est d'utiliser un langage logique pour décrire des situations mathématiques concrètes. Pour cela, on introduit la notion de structure qui permet de relier les formules abstraites à des objets mathématiques réels. Avant de décider si, dans une structure donnée, une formule est vraie ou fausse, il est essentiel de distinguer les variables libres des variables liées.

1. Variables libres, variables liées

1.1. Idée

On considère un langage $\mathcal{L}$ avec des variables $x, y, z, x_1, x_2, \ldots$ Pour une formule donnée, une variable peut être libre ou liée. Comprenons la différence par des exemples avant de donner la définition précise.
Considérons une formule « $\forall x \;\; P(x)$ » ou « $\exists x \;\; P(x)$ », la variable « x » est ici *quantifiée*, c'est-à-dire *liée* à un quantificateur. De même, les deux variables x et y de « $\forall x \; \exists y \;\; Q(x, y)$ » sont quantifiées.
Par contre, dans la formule « $\forall x \;\; Q(x, y)$ » la variable y n'est pas quantifiée, on dit que cette variable est *libre* (mais la formule est quand même légitime).
Voici une formule (dans une interprétation) :

$$\forall x \in \mathbb{R} \quad \exists y \in \mathbb{R} \qquad y > x$$

Comme il n'y a pas de variables libres, on peut attribuer une valeur V/F à cette formule.
Par contre :

$$\forall x \in \mathbb{R} \qquad y > x$$

est une formule où y est une variable libre. Cela n'a pas de sens de se demander si cette formule est vraie ou fausse tant qu'une valeur n'est pas attribuée à y.

1.2. Définition

Portée

Définition.
Dans une formule contenant « $\forall x \;\; \mathcal{P}$ » ou « $\exists x \;\; \mathcal{P}$ », la ***portée*** de $\forall$ (ou $\exists$) est $\mathcal{P}$.

Il est bon de préciser que $\mathcal{P}$ peut être une formule arbitrairement complexe.
Exemple :

$$\underbrace{\forall x \quad \exists y \qquad \underbrace{P(x) \to Q(x, y)}_{\text{portée du } \exists}}_{\text{portée du } \forall}$$

Variable libre ou liée

Définition.
Dans une formule, une variable x est une ***variable liée*** si toute occurrence de x se trouve dans la portée d'un quantificateur $\forall x$ ou $\exists x$. Sinon on dit que x est une ***variable libre***.

Exemples :
- $\forall x \quad P(x) \vee Q(x,y)$: x est liée, y est libre.
- $P(x) \to Q(x,y)$: x et y sont libres.
- $\big(\forall x \quad P(x)\big) \leftrightarrow \big(\exists y \quad P(y)\big)$: x et y sont liées.
- $\big(\forall x \quad x = y\big) \vee \big(\exists y \quad x = y\big)$: x et y sont libres !

Il y a une subtilité à bien comprendre, illustrée par ce dernier exemple. Pour qu'une variable x soit libre, il suffit qu'il existe au moins une occurrence de x qui ne soit pas dans la portée d'un quantificateur. Dans notre exemple y est une variable libre pour la partie gauche et x libre pour la partie droite, donc pour la formule totale, x et y sont bien des variables libres.

Formule fermée

Définition.
Une formule est ***fermée*** si elle ne contient aucune variable libre.

On dit aussi que c'est une formule ***close***, ou bien que c'est une ***phrase***.
L'idée est qu'à une formule fermée on va pouvoir attribuer une valeur V/F alors que pour une formule non fermée il reste encore des variables libres auxquelles il faudra affecter une valeur.
Exemples :
- $\forall x \quad \big(P(x) \wedge (\exists y \quad P(y))\big)$ est une formule fermée.
- $\exists x \quad \Big(\big(\exists y \quad P(x) \wedge P(y)\big) \to \big(\forall y \quad Q(x,y)\big)\Big)$ est aussi une formule fermée.
- $\exists x \quad \big(Q(x,y) \to (\exists z \quad P(z))\big)$ n'est pas fermée car y est une variable libre.
- $\big(\forall x \quad P(x)\big) \vee \big(\exists x \quad Q(x)\big) \vee \big(\neg R(x)\big)$ n'est pas une formule fermée car x est une variable libre (dans la sous-formule de droite).

1.3. Substitution

Dans un langage $\mathcal{L}$, soit $\mathcal{P}$ une formule, x une variable et t un terme.

Définition.
$\mathcal{P}[t/x]$ désigne la formule où toutes les occurrences libres de la variable x ont été remplacées par le terme t.

Remarque : on rappelle qu'un terme désigne les constantes, les variables et si le langage contient un symbole de fonction f, $f(t_1, \ldots, t_n)$ est aussi un terme lorsque $t_1, \ldots, t_n$ sont des termes.

> **Exemple.**
> Soit $\mathcal{L}$ un langage contenant les constantes a, b, une fonction binaire notée $+$, un prédicat unaire P et un prédicat binaire Q.
> Étudions la formule :
>
> $$\mathcal{P} : \qquad \forall y \quad P(x) \to Q(x,y)$$
>
> - $t = a$, $\mathcal{P}[t/x] : \forall y \quad P(a) \to Q(a,y)$
> - $t = x + b$, $\mathcal{P}[t/x] : \forall y \quad P(x+b) \to Q(x+b,y)$
> - $t = b$, $\mathcal{P}[t/y] : \forall y \quad P(x) \to Q(x,y)$: la formule ne change pas car toutes les occurrences de y sont dans la portée d'un quantificateur.

Remarque : la définition formelle de la substitution se fait par induction en commençant par les formules atomiques, puis l'introduction des connecteurs logiques. Pour les formules avec quantificateurs « $\forall x \ \ \mathcal{P}$ » ou « $\exists x \ \ \mathcal{P}$ » il n'y a pas de substitution de la variable x à faire car la variable est quantifiée.

On peut aussi faire une ***substitution multiple*** $\mathcal{P}[t_1/x_1, t_2/x_2, \ldots, t_n/x_n]$. On la note $\mathcal{P}[s]$ où s désigne $(t_1/x_1, t_2/x_2, \ldots, t_n/x_n)$. Il faut bien comprendre que les substitutions sont simultanées : on remplace toutes les occurrences libres de x_1 par t_1 et *en même temps* toutes les occurrences libres de x_2 par t_2, etc.

> **Exemple.**
> Soit $\mathcal{P}$ la formule « $P(x, y)$ ». Alors :
> - $\mathcal{P}[a/x, b/y] : P(a, b)$
> - $\mathcal{P}[x + y/x, x/y] : P(x + y, x)$
> - $\mathcal{P}[y/x, x/y] : P(y, x)$

Remarque : la formule $\mathcal{P}$: « $P(x) \to (\forall x \ \ Q(x))$ » est une formule légitime du langage (c'est comme si on avait deux x différents : un rouge et un bleu). La substitution de x par a donne $\mathcal{P}[a/x]$: « $P(a) \to (\forall x \ \ Q(x))$ » car on ne remplace que les occurrences libres de x. On préférerait écrire comme formule initiale $\mathcal{P}'$: « $P(x) \to (\forall y \ \ Q(y))$ » qui est plus facile à comprendre.

Capture de variable.
Lors d'une substitution $\mathcal{P}[t/x]$, on suppose toujours qu'aucune variable apparaissant dans t ne tombe dans la portée d'un quantificateur de $\mathcal{P}$. Si le terme t contient des variables qui risqueraient d'être liées par des quantificateurs présents dans $\mathcal{P}$, la substitution pourrait modifier le sens logique de la formule : on parle alors de capture de variables.
Par exemple, soit :

$$\mathcal{P} : \qquad \forall y \quad P(x) \to Q(x, y)$$

La substitution de x par $t = y$ n'est pas légitime, car cela donnerait la formule :

$$\mathcal{P}[y/x] : \qquad \forall y \quad P(y) \to Q(y, y)$$

qui n'a pas du tout le même sens.
Afin d'éviter ce phénomène, on renomme préalablement les variables liées de $\mathcal{P}$ avant d'effectuer la substitution. Pour notre exemple on écrira d'abord $\mathcal{P}$ sous une nouvelle forme :

$$\mathcal{P} : \qquad \forall z \quad P(x) \to Q(x, z)$$

puis on effectue la substitution de x par y :

$$\mathcal{P}[y/x] : \qquad \forall z \quad P(y) \to Q(y, z)$$

2. Structures

Un langage reste une coquille abstraite, il sert de cadre à une grande variété d'objets mathématiques et à leurs propriétés : les entiers, les groupes, les espaces vectoriels, la théorie des ensembles…

2.1. Définition

Définition.
On considère un langage $\mathcal{L}$. Une ***structure*** $\mathcal{S}$ de ce langage, c'est la donnée :
- d'un ensemble non vide E, appelé ***domaine***,
- à chaque symbole de constante c de $\mathcal{L}$, on associe un élément $c^{\mathcal{S}} \in E$,

- à chaque symbole de prédicat P à n places, on associe un sous-ensemble $P^{\mathcal{S}} \subset E^n$, autrement dit on associe une fonction $\phi_P : E^n \to \{\mathrm{V}, \mathrm{F}\}$ telle que $\phi_P(a_1, \ldots, a_n)$ vaut vrai si et seulement si $(a_1, \ldots, a_n) \in P^{\mathcal{S}}$,
- à chaque symbole de fonction f à n places, on associe une fonction $f^{\mathcal{S}} : E^n \to E$.

On dit aussi que $\mathcal{S}$ est une ***interprétation*** du langage $\mathcal{L}$. À un même langage, on peut associer plusieurs structures différentes.
Une structure est donc la réalisation mathématique concrète d'un langage. Pour définir une structure, il faut définir tous les symboles non logiques : les constantes, les prédicats et les fonctions. On ne touche pas aux symboles logiques : une variable x reste une variable x, sauf que maintenant elle parcourt le domaine E.

2.2. Un premier exemple

On considère un langage $\mathcal{L} = (c, P, f)$ où c est une constante, $P(x, y)$ un prédicat binaire et $f(x)$ une fonction unaire. Comme d'habitude, on ne précise pas explicitement que les variables sont x, y, qu'il y a les connecteurs logiques et qu'il y a aussi le prédicat d'égalité « $x = y$ ».
On va définir une structure qui correspond aux entiers naturels et modélise le successeur. On considère la structure $\mathcal{S}$ de domaine $E = \mathbb{N}$ avec l'interprétation du langage $(0, <, x \mapsto x + 1)$, c'est-à-dire :

- pour c la constante de $\mathcal{L}$, on définit $c^{\mathcal{S}} = 0$, c'est l'interprétation du symbole c dans $\mathcal{S}$,
- pour P le prédicat de $\mathcal{L}$, on définit $P^{\mathcal{S}}$, par $P^{\mathcal{S}}(x, y) : x < y$,
- pour f la fonction de $\mathcal{L}$, on définit $f^{\mathcal{S}}$, par $f^{\mathcal{S}}(x)$: l'entier successeur de x, que l'on note $x + 1$.

Une fois définie la structure, toute formule de $\mathcal{L}$ s'interprète en un énoncé mathématique sur la structure. Voici des exemples :

Formule de $\mathcal{L}$	Formule interprétée dans $\mathcal{S}$	Vrai ou faux dans $\mathcal{S}$?
$\forall x \quad \exists y \quad P(x, y)$	$\forall x \in \mathbb{N} \quad \exists y \in \mathbb{N} \quad x < y$	C'est vrai
$\forall x \quad P(c, f(x))$	$\forall x \in \mathbb{N} \quad 0 < x + 1$	C'est aussi vrai
$\forall x \quad f(x) = y$	$\forall x \in \mathbb{N} \quad x + 1 = y$	La question n'a pas de sens car y est une variable libre

Pour définir cette structure $\mathcal{S}$, on écrira souvent abusivement $c = 0$, $P(x, y) : x < y$, $f(x) = x + 1$ (au lieu de $c^{\mathcal{S}}$, $P^{\mathcal{S}}$, $f^{\mathcal{S}}$).

Il y a évidemment beaucoup d'autres structures possibles pour ce même langage. On peut changer le domaine, la constante, le prédicat, la fonction :

- Structure $\mathcal{S}_1$ de domaine $\mathbb{R}$, avec toujours $(0, <, x \mapsto x + 1)$, sauf que cette fois la formule interprétée $\forall x \in \mathbb{R} \quad 0 < x + 1$ est devenue fausse.
- Structure $\mathcal{S}_2$ de domaine $\mathbb{R}$, avec $(0, \leqslant, x \mapsto x^2)$, la formule $\forall x \quad P(c, f(x))$ du langage devient $\forall x \in \mathbb{R} \quad 0 \leqslant x^2$, une formule vraie dans cette interprétation.

2.3. Commentaires

Voici des remarques importantes.

Domaine non vide

Le domaine d'une structure doit être un ensemble non vide. Cela peut sembler un détail, mais c'est assez important. Il semble naturel pour tout le monde que :

$$\text{Si } \ll \forall x \in E \quad P(x) \gg \text{ est vrai, alors } \ll \exists x \in E \quad P(x) \gg \text{ est aussi vrai.}$$

Mais imaginons un instant que E soit l'ensemble vide, alors on aurait toujours « $\forall x \in E \;\; P(x)$ » qui est vrai (puisqu'il n'y a rien à vérifier, car pas de x à considérer). Par contre, « $\exists x \in E \;\; P(x)$ » est toujours faux (car E est vide). Cette situation serait problématique.
On peut aussi argumenter en disant que si E était vide, alors « $\forall x \in E \;\; x \neq x$ » est vrai et « $\exists x \in E \;\; x = x$ » est faux, ce qui serait très embêtant.
Bilan : on suppose toujours le domaine E non vide.

Fonctions

Revenons un peu sur les fonctions. Dans le langage, f est juste un symbole de fonction qui doit être suivi de n termes (pour une fonction à n places). Dans la structure, il faut définir l'interprétation de cette fonction, par une bonne vieille fonction : $f^{\mathcal{S}} : E^n \to E$. C'est une fonction de n variables au sens usuel du terme. Comme déjà dit, dans la pratique on la note simplement $f : E^n \to E$.
Par exemple, soit $\mathcal{L}$ un langage avec une fonction unaire f et une fonction binaire g.

- Pour un domaine $\mathbb{N}$, on pourrait définir f par $f(x) = x + 1$ ou $f(x) = x^2$ et g par $g(x, y) = x + y$ ou $g(x, y) = xy$. Par contre, on ne peut pas définir f par $f(x) = x - 1$ car $f(0)$ n'est pas dans le domaine $\mathbb{N}$, ni définir g par $g(x, y) = x/y$ qui ne donne pas toujours des entiers.
- En revanche, sur le domaine $]-\infty, 0[$, on pourrait définir $f(x) = x - 1$ et $g(x, y) = -x/y$.

Prédicats

L'interprétation d'un prédicat est peut-être un peu plus difficile à comprendre. Dans le langage un prédicat n-aire est juste un symbole qui doit être suivi de n termes. Pour la structure il faut définir chaque prédicat, c'est-à-dire pour quels objets on obtient la valeur Vrai.
Prenons l'exemple d'un prédicat unaire P dans un langage $\mathcal{L}$. Pour définir une structure $\mathcal{S}$, on définit son domaine E et ensuite il faut définir l'interprétation du prédicat, c'est-à-dire quelles sont toutes les valeurs de E qui rendent $P(x)$ vraie. On peut le faire en définissant un sous-ensemble $P^{\mathcal{S}} \subset E$, qui est donc exactement l'ensemble des $d \in E$, tels que $P(d)$ soit vraie. On peut aussi le définir comme une fonction $\phi_P : E \to \{\mathrm{V}, \mathrm{F}\}$ qui renvoie Vrai ou Faux. Le lien entre les deux étant que $\phi_P(d) = \mathrm{V}$ si et seulement si $d \in P^{\mathcal{S}}$.
Ainsi, si on veut définir une interprétation de « être un réel plus grand que 7 », on pose $E = \mathbb{R}$ et pour le prédicat, soit on définit $P^{\mathcal{S}} = [7, +\infty[$ l'ensemble des valeurs qui vérifient la propriété, ou bien on définit une fonction :

$$\begin{array}{rcl} \phi_P : E & \longrightarrow & \{\mathrm{V}, \mathrm{F}\} \\ x & \longmapsto & \begin{cases} \mathrm{V} & \text{si } x \in [7, +\infty[\\ \mathrm{F} & \text{sinon.} \end{cases} \end{array}$$

Remarque : on a bien insisté sur l'exigence qu'un domaine doit être non vide. Par contre, on n'exige pas qu'un prédicat puisse être réalisé. C'est-à-dire qu'on peut avoir P qui s'évalue toujours à faux dans la structure. Ainsi sur $\mathbb{R}$ on a le droit de définir les prédicats « $x^2 < 0$ » ou bien « $x \neq x$ », même si aucun $x \in \mathbb{R}$ ne va les réaliser.
Pour définir un prédicat binaire $P(x, y)$ dans une structure de domaine E, il faut définir un ensemble $P^{\mathcal{S}} \subset E^2$, qui sera l'ensemble des couples (d_1, d_2) tels que $P(d_1, d_2)$ soit vrai.

Domaine

Lorsque l'on définit une structure, on fixe un domaine E. Les variables sont alors interprétées comme des objets de ce domaine. Par exemple la formule du langage « $\forall x \; \exists y \quad Q(x, y)$ » peut s'interpréter dans une structure où $E = \mathbb{N}$ par « $\forall x \in \mathbb{N} \; \exists y \in \mathbb{N} \quad x < y$ ». Mais alors comment écrire une formule dans laquelle x et y ne parcourent pas exactement le même ensemble, comme par exemple « $\forall \varepsilon > 0 \; \exists n \in \mathbb{N} \quad n > \frac{1}{\varepsilon}$ » ?

Il faut ruser un peu. Soient A et B deux sous-ensembles d'un même ensemble E. On souhaiterait écrire la phrase mathématique :

$$\forall x \in A \quad \exists y \in B \qquad Q(x,y)$$

Elle ne peut pas être issue d'une formule du langage, car x et y doivent parcourir le même ensemble. Définissons un prédicat P_A, de sorte que $P_A(x)$ soit vrai exactement si $x \in A$. De même on définit P_B avec $P_B(x)$ vrai si $x \in B$. Considérons la formule du langage :

$$\forall x \quad \big(P_A(x) \;\to\; \exists y\ (P_B(y) \wedge Q(x,y))\big)$$

Elle s'interprète dans notre structure de domaine E (avec P_A et P_B définis comme ci-dessus) par :

$$\forall x \in E \quad \big(x \in A \;\to\; \exists y \in E\ (y \in B) \wedge Q(x,y)\big)$$

Si $x \notin A$, alors cet énoncé est vrai (car l'implication est vraie par définition). Si $x \in A$, pour que cet énoncé soit vrai, il faut qu'il existe un $y \in B$ tel que $Q(x,y)$ soit vrai. C'est bien mathématiquement ce qu'on voulait.

2.4. D'autres exemples

Inverse

On souhaite exprimer l'existence d'un inverse (ou d'un opposé). C'est une notion très générale que l'on retrouve dans beaucoup de situations et qu'il est donc intéressant de conceptualiser. Soit $\mathcal{L}$ le langage muni d'une constante e et d'une fonction binaire $f(x,y)$ qu'on notera $x \circ y$. On peut écrire une formule qui exprime l'existence d'un inverse (à droite) :

$$\forall x \quad \exists y \qquad x \circ y = e$$

On dit qu'un tel y est un *inverse (à droite)* de x.
Concrètement, voici ce que cela recouvre :

- **Opposé.** On définit la structure $\mathcal{S}$ de domaine $\mathbb{Z}$, avec $e^{\mathcal{S}} = 0$ et $f^{\mathcal{S}}(x,y) = x + y$. La formule du langage devient :
$$\forall x \in \mathbb{Z} \quad \exists y \in \mathbb{Z} \qquad x + y = 0$$
Autrement dit, $y = -x$ est ici l'*opposé* de x.
- **Inverse multiplicatif.** Cette fois on définit la structure $\mathcal{S}$ de domaine $\mathbb{R}^*$, avec $e^{\mathcal{S}} = 1$ et $f^{\mathcal{S}}(x,y) = x \times y$. La formule du langage devient :
$$\forall x \in \mathbb{R}^* \quad \exists y \in \mathbb{R}^* \qquad x \times y = 1$$
Autrement dit, $y = \frac{1}{x}$ est ici l'*inverse* de x (au sens algébrique).
- **Inverse de matrice.** Soit l'ensemble des matrices inversibles de taille $n \times n$, muni de la multiplication de matrices et $e^{\mathcal{S}} = I$ est la matrice identité. Dans cette structure, la formule exprime $A \times B = I$ c'est-à-dire que $B = A^{-1}$ est l'inverse de la matrice A.
- **Etc.** On pourrait continuer les exemples avec un groupe $(G, \diamond)$ d'élément neutre e, pour exprimer $g \diamond h = e$; ou bien la notion de bijection réciproque d'une fonction pour exprimer $g \circ h = \mathrm{id}$, etc. Tout cela avec le même langage $\mathcal{L}$.

Il y a bien sûr d'autres formules du langage intéressantes pour toutes ces structures. L'associativité :

$$\forall x \quad \forall y \quad \forall z \qquad x \circ (y \circ z) = (x \circ y) \circ z$$

L'action de l'élément neutre qui ne change rien :

$$\forall x \qquad \big((e \circ x) = x\big) \wedge \big((x \circ e) = x\big)$$

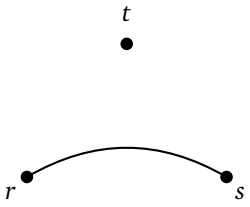
Graphe

Illustrons le fait qu'une même formule peut devenir vraie ou fausse en fonction de la structure choisie. Considérons le langage $\mathcal{L}$ ayant le prédicat $R(x,y)$ avec pour but de modéliser des graphes non orientés. Considérons la formule de $\mathcal{L}$:

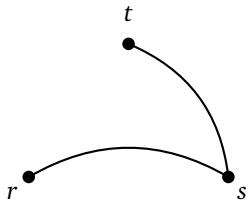
$$\forall x \quad \exists y \qquad (x \neq y) \wedge R(x,y)$$

qu'il faut comprendre comme « tout sommet est relié à au moins un autre sommet ».

- On définit $\mathcal{S}_1$ de domaine $E = \{r,s,t\}$ où $R^{\mathcal{S}_1}$ est défini par le graphe de gauche G_1 ci-dessous, c'est-à-dire que R est vrai seulement pour les couples (r,s) et (s,r) (qui sont reliés entre eux).
  Dans cette structure, la formule interprétée est fausse, car t n'est relié à personne.
- Si, en revanche, on définit $\mathcal{S}_2$ de domaine $E = \{r,s,t\}$ où cette fois le prédicat est défini par le graphe de droite G_2, alors la formule interprétée est vraie.



Graphe G_1



Graphe G_2

2.5. Exemples avancés

Anneaux

On considère un langage $\mathcal{L} = (0,1,+,-,\times)$, où 0, 1 sont des constantes, $+$, $-$, $\times$ sont des fonctions binaires (en toute rigueur on devrait écrire ces symboles a, b, f, g, h, car ici 0 n'est qu'un symbole du langage, noté ainsi par anticipation). Ce langage est adapté afin de modéliser la notion d'anneau (un ensemble avec addition, soustraction, multiplication, mais pas nécessairement d'inverse).
Voici des exemples de structures :

- **Entiers.** On considère la structure $\mathcal{S}$ de domaine $\mathbb{Z}$. Les constantes sont naturellement interprétées par les entiers 0 et 1 et les fonctions sont interprétées par les opérations usuelles sur les entiers.
- **Polynômes.** Pour cette structure $\mathcal{S}$, le domaine est l'ensemble des polynômes $\mathbb{R}[X]$; cette fois 0 désigne le polynôme nul, 1 le polynôme constant égal à 1, les opérations sont les additions, soustractions et multiplications de polynômes.
- **Matrices.** Pour cette nouvelle structure, le domaine est l'ensemble $M_n(\mathbb{R})$ des matrices de taille $n \times n$. Naturellement 0 désigne la matrice nulle, 1 désigne la matrice identité I, les opérations sont les opérations sur les matrices.

L'intérêt du langage est donc de pouvoir formaliser des propriétés qui s'appliquent à de nombreuses situations.
Pour un anneau, on exige de plus que les formules suivantes soient vraies, une fois interprétées dans la structure :

- $\forall x \quad x - x = 0$: la soustraction correspond à l'opposé,
- $\forall x \quad 1 \times x = x$: 1 est élément neutre pour la multiplication,
- $\forall x \ \ \forall y \ \ \forall z \quad x \times (y+z) = x \times y + x \times z$: distributivité,
- etc.

Théorie des ensembles

On considère un langage $\mathcal{L} = (\in)$ où le symbole d'appartenance « $\in$ » désigne un prédicat binaire $P(x,y)$, ici noté $x \in y$.

Avertissement : dans les structures suivantes, les objets vont être des ensembles, et pour le prédicat $x \in y$ il s'agira de savoir si l'ensemble x est bien un *élément* de l'ensemble y. Par exemple $\{1\} \in \{\varnothing, \{1\}, \{2\}, \{1,2\}\}$: c'est bien un ensemble qui est un élément d'un autre ensemble. Le symbole $\in$ ne désignera pas l'appartenance usuelle $1 \in \{1,2,3\}$ (puisque 1 et $\{1,2,3\}$ sont de nature différente, l'un est un entier, l'autre un ensemble d'entiers), ni l'inclusion $\{1,2\} \subset \{1,2,3\}$ qui désigne autre chose.
Construisons une première structure $\mathcal{S}_1$. Son domaine est $E = \{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\{\{\varnothing\}\}\}, \ldots\}$, chaque élément est un singleton, sauf l'ensemble vide. Le prédicat est défini par les relations suivantes :

$$\varnothing \in \{\varnothing\} \qquad \{\varnothing\} \in \{\{\varnothing\}\} \quad \ldots$$

Les autres appartenances ne sont pas vérifiées.
Formellement on définit $E = \bigcup_{n \geqslant 0} E_n$ où :

$$E_0 = \{\varnothing\} \quad \text{et} \quad E_{n+1} = E_n \cup \big\{\{x\} \mid x \in E_n\big\}$$

Ainsi E_1 est formé à partir de $\varnothing$ et $\{\varnothing\}$, donc $E_1 = \{\varnothing, \{\varnothing\}\}$, E_2 est formé par E_1 union $\{\{\varnothing\}, \{\{\varnothing\}\}\}$ donc $E_2 = \{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}\}$ (on ne compte pas $\{\varnothing\}$ deux fois).
Pour définir le prédicat dans cette structure, on donne l'ensemble des $(x, y) \in E^2$ tels que $x \in y$, c'est :

$$\in^{\mathcal{S}} : \qquad \big\{(x, \{x\}) \mid x \in E\big\}$$

Autrement dit, on a $x \in \{x\}$ et rien d'autre. Dans cette structure, si $x \in y$ alors $y = \{x\}$. On dit que $\{x\}$ est le *successeur* de x.
À quoi peut bien servir une construction si compliquée ? Si on note $\varnothing = 0$, $\{\varnothing\} = 1$, $\{\{\varnothing\}\} = 2$, alors 1 est le successeur de 0, 2 est le successeur de 1, etc. On a construit un modèle pour l'ensemble des entiers naturels $\mathbb{N}$, ayant ici pour seule propriété que tout entier admet un successeur.

Quand on construit une théorie des ensembles, on ajoute des axiomes ou des notions complémentaires. Voici donc des formules que l'on exige vraies quand on les interprète dans la structure :

- Égalité de deux ensembles. Deux ensembles sont égaux si, et seulement si, ils possèdent les mêmes éléments :
$$\forall x \quad \forall y \qquad (x = y) \leftrightarrow \big(\forall z \quad (z \in x \leftrightarrow z \in y)\big)$$
- Axiome de l'ensemble vide :
$$\exists x \quad \forall y \qquad \neg(y \in x)$$
C'est-à-dire qu'il existe un ensemble ne contenant aucun élément.
- On définit l'inclusion par :
$$x \subset y \quad \text{signifie par définition} \quad \forall z \quad \big(z \in x \to z \in y\big)$$
- Axiome de l'union d'ensembles :
$$\forall x \quad \exists u \quad \forall z \qquad \big(z \in u\big) \leftrightarrow \big(\exists y \quad (y \in x) \wedge (z \in y)\big)$$
Par exemple si $x = \big\{\{1,2\}, \{3\}, \{4,5\}\big\}$, alors $u = \bigcup_x = \{1,2,3,4,5\}$ (y pourrait par exemple être $\{1,2\}$ et alors z serait 1 ou 2).

On peut définir une autre structure liée à la théorie des ensembles. Cette fois on définit $E = \bigcup_{n \geqslant 0} E_n$ avec :

$$E_0 = \varnothing \quad \text{et} \quad E_{n+1} = \mathcal{P}(E_n)$$

où $\mathcal{P}(X)$ désigne l'ensemble des parties d'un ensemble X. Ainsi :

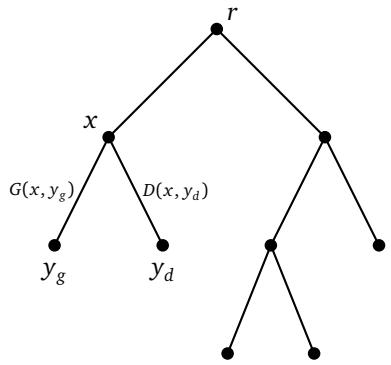
- $E_0 = \varnothing$
- $E_1 = \big\{\varnothing\big\}$, car le vide est la seule partie de E_0,
- $E_2 = \big\{\varnothing, \{\varnothing\}\big\}$, ce sont les parties de E_1,
- $E_3 = \big\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing, \{\varnothing\}\}\big\}$: le vide, deux singletons, une paire,
- … E_n contient un nombre d'éléments qui croît très rapidement.

L'appartenance est ici l'appartenance usuelle, par exemple $\{\varnothing\} \in \{\varnothing, \{\varnothing\}\}$, par contre, $\varnothing \notin \{\{\varnothing\}\}$.

Arbres binaires

On souhaite modéliser la notion d'arbre binaire (chaque sommet a 0 ou 2 enfants). Soit un langage $\mathcal{L} = (r, G, D)$ où r est une constante, $G(x, y)$ et $D(x, y)$ sont des prédicats binaires.

On définit une structure $\mathcal{S}$ pour ce langage :
- E est un ensemble de sommets (éventuellement infini),
- $r \in E$ est un sommet distingué, appelé la *racine*,
- $G \subset E^2$ est la relation *enfant gauche*, $G(x, y)$ vaut vrai si y est l'enfant gauche de x,
- $D \subset E^2$ est la relation *enfant droit*, $D(x, y)$ vaut vrai si y est l'enfant droit de x.



On définit une liste d'*axiomes*, c'est-à-dire des formules qui devront être vraies pour toute structure définissant un arbre binaire.
- Chaque sommet a au plus un enfant gauche et au plus un enfant droit :

$$\forall x \quad \forall y \quad \forall z \qquad G(x,y) \wedge G(x,z) \to y = z$$

$$\forall x \quad \forall y \quad \forall z \qquad D(x,y) \wedge D(x,z) \to y = z$$

- Les enfants gauche et droit sont distincts :

$$\forall x \quad \forall y \quad \forall z \qquad G(x,y) \wedge D(x,z) \to y \neq z$$

- Un sommet n'a qu'un seul parent :

$$\forall x \quad \forall y \quad \forall z \qquad G(x,z) \wedge G(y,z) \to x = y$$

$$\forall x \quad \forall y \quad \forall z \qquad D(x,z) \wedge D(y,z) \to x = y$$

$$\forall x \quad \forall y \quad \forall z \qquad G(x,z) \wedge D(y,z) \to x = y$$

- La racine n'est l'enfant de personne :

$$\forall x \qquad \neg G(x,r)$$

$$\forall x \qquad \neg D(x,r)$$

- Chaque sommet a soit zéro enfant, soit deux enfants :

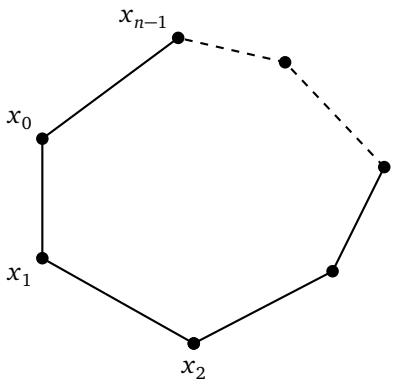
$$\forall x \qquad \big(\exists y \quad G(x,y)\big) \leftrightarrow \big(\exists z \quad D(x,z)\big)$$

- Axiome optionnel qui exige que l'arbre soit connexe (en un seul morceau), tout sommet autre que la racine doit avoir un parent :

$$\forall y \qquad \big(y \neq r\big) \to \big(\exists x \quad G(x,y) \vee D(x,y)\big)$$

  (Cela assure la connexité dans le cas des arbres finis.)

Il reste un point délicat à exprimer : un arbre n'a pas de cycle (en suivant un chemin, on ne peut pas revenir sur son point de départ).



Cycle dans un graphe

L'absence de cycle n'est pas exprimable par une seule formule, ni même par un nombre fini de formules. Pour tout entier $n \geqslant 1$ et pour toute suite de relations $R_1, \ldots, R_n \in \{G, D\}$, on impose que le chemin correspondant ne puisse pas revenir à son point de départ. Cela s'exprime par la formule :

$$\forall x_0 \quad \forall x_1 \quad \cdots \quad \forall x_{n-1} \qquad \neg \Big(R_1(x_0, x_1) \wedge R_2(x_1, x_2) \wedge \cdots \wedge R_n(x_{n-1}, x_0) \Big)$$

Exercices

Variables libres, variables liées

Exercice 1. Pour chacune des formules suivantes, indiquer la portée de chaque quantificateur, lister les variables libres et les variables liées parmi x, y, z. La formule est-elle une formule fermée ?

1. $\forall x \quad P(x) \to Q(a, x)$
2. $\exists z \quad Q(x, y) \wedge Q(y, z) \wedge Q(z, x)$
3. $\forall x \quad \big(T(x, y, c) \to (\forall z \quad T(a, y, z)) \big)$
4. $(\forall x \quad P(x)) \wedge (\exists y \quad P(y)) \wedge (x \neq y)$
5. $\big((\exists x \quad P(x)) \vee (\exists y \quad Q(x, y)) \big) \to (\forall z \quad P(z))$
6. $\big(\exists x \quad \exists y \quad P(x) \wedge P(y) \wedge (x \neq y) \big) \to P(z)$
7. $\forall x \quad \big(Q(x, x) \to (\exists x \quad P(x)) \big)$

Exercice 2. Pour chacune des formules suivantes données dans une structure, déterminer les variables libres et les variables liées (parmi x, y). Si la formule est fermée dire si la phrase est vraie ou fausse. Si la formule n'est pas fermée dire pour quelles valeurs des variables libres la formule devient vraie.

1. $\forall x \in \mathbb{N} \quad x^2 \geqslant y$
2. $\forall x \in \mathbb{N} \quad \exists y \in \mathbb{N} \quad x \geqslant y + 1$
3. $\forall y \in \mathbb{N} \quad \exists x \in \mathbb{N} \quad x \geqslant y + 1$
4. $\exists y \in \mathbb{R} \quad x \neq y$
5. $\forall y \in \mathbb{R} \quad x \neq y$
6. $\big(\forall x \in \mathbb{R} \quad x^2 \geqslant 0 \big) \to \big(y = x^2 \to y \geqslant 0 \big)$

Exercice 3. Réécrire les formules après chaque substitution indiquée.

1. $\mathcal{P}_1 : \forall x \quad \big(P(x) \wedge Q(x, y) \big)$; $\mathcal{P}_1[b/y]$, puis $\mathcal{P}_1[a/x]$, puis $\mathcal{P}_1[x/y]$.
2. $\mathcal{P}_2 : P(x) \to (\exists y \quad Q(x, y))$; $\mathcal{P}_2[x + a/x]$, puis $\mathcal{P}_2[y + b/y]$.
3. $\mathcal{P}_3 : \exists z \quad \big(x \neq y \to T(x, y, z) \big)$; $\mathcal{P}_3[a/x, b/y, c/z]$.
4. $\mathcal{P}_4 : \big(\forall x \quad P(x) \big) \wedge \big(\exists y \quad Q(x, y) \big) \wedge \big(\forall z \quad \exists x \quad T(x, y, z) \big)$; $\mathcal{P}_4[y/x, x/y, c/z]$.

Structures

Exercice 4. On considère le langage $\mathcal{L} = (a, f)$ avec une constante a et une fonction binaire $f(x, y)$. Pour chaque formule et pour chaque structure, donner l'interprétation de la formule et dire si cette formule interprétée est vraie ou fausse.
Formules :

- $\mathcal{P}_1 : \forall x \quad \exists y \quad f(x, y) = a$
- $\mathcal{P}_2 : \forall x \quad \exists y \quad f(x, x) = f\big(f(y, y), y\big)$
- $\mathcal{P}_3 : \forall x \quad \forall y \quad f(x, y) = a \to x = a$

Structures :

1. $\mathcal{S}_i : E = \mathbb{Z}$, $a = 0$, $f(x, y) = x + y$
2. $\mathcal{S}_{ii} : E = \mathbb{R}$, $a = 0$, $f(x, y) = x - y$
3. $\mathcal{S}_{iii} : E =]0, +\infty[$, $a = 1$, $f(x, y) = xy$

Exercice 5. On considère le langage $\mathcal{L} = (a, b, P, Q)$ où a, b sont des constantes et $P(x)$ et $Q(x, y)$ des prédicats (les variables sont x, y). Pour chaque formule et pour chaque structure, donner l'interprétation de la formule et dire si cette formule interprétée est vraie ou fausse.
Formules :

- $\mathcal{P}_1 : \forall x \quad Q(x, a)$
- $\mathcal{P}_2 : \exists x \quad \forall y \quad P(x) \wedge Q(x, y)$
- $\mathcal{P}_3 : \big(\forall x \quad P(x)\big) \to \big(\exists y \quad Q(a, y)\big)$

Structures :

1. $\mathcal{S}_i : E = \mathbb{N}$, $a = 0$, $b = 1$, $P(x) :$ « $x \neq 1$ », $Q(x, y) :$ « $x > y$ »
2. $\mathcal{S}_{ii} : E = M_2$ l'ensemble des matrices 2×2 à coefficients réels, a la matrice nulle, b la matrice identité, $P(x) :$ « $\det(x) \neq 0$ » (le déterminant est non nul), $Q(x, y) :$ « $xy = I$ » (le produit vaut l'identité)

Exercice 6. Soit $\mathcal{L}$ un langage avec un prédicat $R(x, y)$. On considère $E = \{r, s, t, u\}$. Pour chacune des formules suivantes, définir une interprétation du prédicat $R(x, y)$ sous la forme d'un graphe non orienté afin que la formule soit vraie dans cette structure. (Il peut y avoir plusieurs solutions possibles.)

1. $\forall x \quad \forall y \quad (x \neq y) \to R(x, y)$
2. $\big(\forall x \quad R(x, x)\big) \wedge \big(\forall x \quad \exists y \quad \neg R(x, y)\big)$
3. $\forall x \quad \forall y \quad \forall z \quad (R(x, y) \wedge R(x, z)) \to (y = z)$
4. $\neg \Big(\forall x \quad \forall y \quad \forall z \quad \big(R(x, z) \wedge R(z, y)\big) \to R(x, y)\Big)$

Chapitre 7

Satisfaction

Pour attribuer une valeur Vrai/Faux à une formule de la logique du premier ordre, il faut à la fois définir une structure mathématique et affecter une valeur aux variables. On pourra alors définir la conséquence logique.

1. Vrai/faux

Dans la logique Vrai/Faux, on attribuait une valeur V/F à une formule en affectant à chaque variable une valeur V/F. Pour une formule dans un langage du premier ordre, c'est un peu plus compliqué. Il faut d'abord définir une structure, afin d'interpréter la formule dans ce cadre mathématique, et si il y a des variables libres, il faut en plus affecter une valeur mathématique à chacune d'elles.

1.1. Pour tout & Il existe

La grande nouveauté de la logique du premier ordre est bien sûr l'introduction des quantificateurs. On considère un langage $\mathcal{L}$ et un prédicat $P(x)$. On fixe une structure $\mathcal{S}$ de ce langage de domaine E.

« $\forall x\ P(x)$ » est vrai ssi pour tout $x \in E$, $P^{\mathcal{S}}(x)$ est vrai

« $\exists x\ P(x)$ » est vrai ssi il existe (au moins) un $x \in E$ tel que $P^{\mathcal{S}}(x)$ est vrai

Vous remarquez que l'on est obligé de définir ces notions par des phrases en français, c'est-à-dire par un méta-langage. L'autre point important est que cette évaluation Vrai/Faux n'a pas de sens au niveau du langage lui-même, mais se fait au niveau d'une structure. On rappelle que $P^{\mathcal{S}}(x)$ est l'interprétation du prédicat $P(x)$ dans la structure $\mathcal{S}$. Bien sûr, le résultat peut dépendre du choix de la structure.

Exemple.

- La formule
$$\exists x \quad x^2 < x$$
est fausse si on considère une structure avec le domaine $E_1 = [1, +\infty[$, mais vraie si le domaine est $\mathbb{R}$ (par exemple $(\frac{1}{2})^2 < \frac{1}{2}$).
- La formule
$$\forall x \quad (\exists y \quad y^2 = x)$$
est vraie si on considère le domaine $E_1 = [1, +\infty[$, mais fausse si le domaine est $\mathbb{R}$ (par exemple -1 n'est pas le carré d'un nombre réel).

Remarque : si on a une formule $\mathcal{P}$ dont la seule variable libre est x, alors on peut aussi décider si les phrases « $\forall x\ \mathcal{P}$ » et « $\exists x\ \mathcal{P}$ » sont vraies ou fausses dans la structure. Comme d'habitude, on notera souvent, un peu abusivement, $\mathcal{P}(x)$ une formule $\mathcal{P}$ pour signifier qu'elle dépend de la variable x.

1.2. Vrai/Faux pour les formules fermées

On fixe un langage $\mathcal{L}$. On rappelle qu'une variable est ***libre*** dans une formule, si l'une de ses occurrences ne se trouve pas dans la portée d'un quantificateur portant sur cette variable. Une formule sans variable libre est une ***formule fermée***.

Définition.
Soit $\mathcal{S}$ une structure et soit $\mathcal{Q}$ une formule fermée. Si l'interprétation de $\mathcal{Q}$ dans $\mathcal{S}$ est vraie, alors on dit que $\mathcal{Q}$ est ***satisfaite*** dans $\mathcal{S}$ et on note :

$$\boxed{\vDash_{\mathcal{S}} \mathcal{Q}}$$

On dit aussi que $\mathcal{S}$ est un ***modèle*** pour $\mathcal{Q}$. On dit que $\mathcal{Q}$ est ***satisfaisable*** s'il existe une structure $\mathcal{S}$ telle que $\vDash_{\mathcal{S}} \mathcal{Q}$.

> **Exemple.**
> On considère un langage adapté et la formule :
> $$\mathcal{Q} : \qquad \forall x \quad (x \times x = a) \to x = a$$
> Pour la structure $\mathcal{S}_1$ de domaine $E = \mathbb{R}$ avec $a^{\mathcal{S}_1} = 0$ alors la formule est vraie, ainsi on a bien $\vDash_{\mathcal{S}_1} \mathcal{Q}$. Par contre, pour la structure $\mathcal{S}_2$ de domaine $E = \mathbb{R}$, mais avec cette fois $a^{\mathcal{S}_2} = 1$, la formule est fausse (en effet, si $x^2 = 1$ alors $x = +1$ ou $x = -1$).

1.3. Affectation

Considérons un langage $\mathcal{L}$ et une structure $\mathcal{S}$ de domaine E. Si x est une variable de $\mathcal{L}$ et d un élément de l'ensemble E, alors $x \mapsto d$ désigne l'***affectation*** de la valeur d à chaque occurrence libre de x dans les formules du langage. On note $\mathcal{Q}[x \mapsto d]$ la formule, interprétée dans $\mathcal{S}$, où les occurrences libres de x sont remplacées par la valeur d.
Par exemple, considérons la formule $\mathcal{Q}$: « $x^2 \geqslant 7$ » pour une structure de domaine $\mathbb{R}$. Comme la variable x est libre, cela n'a pas de sens de vouloir savoir si $\mathcal{Q}$ est vraie ou fausse. Par contre, après affectation, la formule $\mathcal{Q}[x \mapsto 3]$ devient « $3^2 \geqslant 7$ » et est donc vraie, alors que $\mathcal{Q}[x \mapsto 2]$ devient « $2^2 \geqslant 7$ » et est fausse.
Plus généralement, une ***affectation*** est une fonction $s : \text{Var} \to E$ qui à chaque variable x_i du langage associe une valeur d_i du domaine : $x_i \mapsto d_i$. On note $\mathcal{Q}[s]$ l'application d'une affectation s à une formule $\mathcal{Q}$. Insistons sur le fait que seules les occurrences des variables libres sont affectées.

1.4. Formules avec variables libres

Définition.
Soit $\mathcal{Q}$ une formule. On dit que $\mathcal{Q}$ est ***satisfaite*** dans $\mathcal{S}$ pour l'affectation s si l'interprétation $\mathcal{Q}[s]$ est vraie. On note alors $\vDash_{\mathcal{S}} \mathcal{Q}[s]$ ou aussi :

$$\boxed{\vDash_{\mathcal{S},s} \mathcal{Q}}$$

Bien évidemment, l'affectation n'intervient que s'il y a des variables libres. L'opération d'affectation est le pendant de l'évaluation en logique Vrai/Faux (on remplaçait chaque variable propositionnelle par une valeur V/F) : ici c'est plus général car x est remplacé par n'importe quelle valeur de l'ensemble E (le domaine).

Exemple.
Considérons un langage dans lequel on peut écrire la formule :

$$\mathcal{Q} : \qquad \forall x \quad y \leqslant x$$

La variable y est une variable libre. On va considérer une structure $\mathcal{S}_1$ de domaine $\mathbb{N}$ et une structure $\mathcal{S}_2$ de domaine $\mathbb{R}$ (et l'inégalité usuelle). Cela n'a pas de sens de se demander si $\mathcal{Q}$ est vraie, ni dans $\mathcal{S}_1$, ni dans $\mathcal{S}_2$.

- Dans $\mathcal{S}_1$, considérons l'affectation $s : y \mapsto 0$. On a bien
$$\mathcal{Q}[y \mapsto 0] : \qquad \forall x \in \mathbb{N} \quad 0 \leqslant x$$
et donc l'énoncé $\mathcal{Q}[y \mapsto 0]$ est vrai, donc on a bien $\vDash_{\mathcal{S}_1, y \mapsto 0} \mathcal{Q}$.
- Dans $\mathcal{S}_2$, l'affectation $s : y \mapsto 0$ conduit à :
$$\mathcal{Q}[y \mapsto 0] : \qquad \forall x \in \mathbb{R} \quad 0 \leqslant x$$
qui est bien sûr faux.
- L'affectation $y \mapsto 1$ conduit à des énoncés faux pour les deux structures.
- Enfin, soit s l'affectation telle que $x \mapsto 5$ et $y \mapsto 1$. Comme x n'est pas une variable libre, s n'affecte pas les x de la formule qui devient : $\mathcal{Q}[s]$: « $\forall x \quad 1 \leqslant x$ ».

Validité

Définition.
Soit $\mathcal{Q}$ une formule d'un langage $\mathcal{L}$. On dit que $\mathcal{Q}$ est une ***validité*** si quelle que soit la structure $\mathcal{S}$ et quelle que soit l'affectation s, l'interprétation de $\mathcal{Q}[s]$ dans $\mathcal{S}$ est vraie. On note alors :

$$\boxed{\vDash \mathcal{Q}}$$

Cela signifie donc qu'on a $\vDash_{\mathcal{S},s} \mathcal{Q}$, quelles que soient $\mathcal{S}$ et s.
Une validité est l'analogue d'une tautologie de la logique propositionnelle. Si on a $\vDash \neg\mathcal{Q}$ cela signifie que $\mathcal{Q}$ est toujours fausse (quelles que soient $\mathcal{S}$ et s), une telle formule $\mathcal{Q}$ s'appelle une ***contradiction*** ou est dite ***insatisfaisable***.

Exemple.
- « $\forall x \; x = x$ » est une validité.
- « $x = x$ » est aussi une validité.
- « $\forall x \quad P(x) \wedge Q(x) \to P(x)$ » est une validité (on n'a pas besoin de savoir comment les prédicats P et Q seront interprétés).
- « $\exists x \; \exists y \; x \neq y$ » n'est pas une validité. C'est très souvent vrai, mais pas si on choisit une structure dont le domaine n'a qu'un seul élément, par exemple $E = \{0\}$ pour lequel il ne peut pas exister deux éléments distincts.
- « $x < x$ » n'est pas une validité (c'est faux sur $\mathbb{R}$ si on choisit pour « $<$ » l'inégalité stricte, mais c'est vrai si on choisit l'inégalité large).
- « $\exists x \; P(x) \wedge \neg P(x)$ » est une contradiction. Même si on ne sait pas en quoi le prédicat P va être interprété, on ne pourra jamais avoir en même temps $P(x)$ et son contraire. (Noter que le domaine d'une structure est non vide, c'est important ici).

2. Conséquence logique

2.1. Définition

Nous généralisons la notion de satisfaction, sous la forme suivante : si les hypothèses $\mathcal{P}_1, \ldots, \mathcal{P}_l$ sont satisfaites, alors la conclusion $\mathcal{Q}$ est aussi satisfaite.

Définition.
On dit que $\mathcal{P}_1, \ldots, \mathcal{P}_l$ ont pour ***conséquence logique*** $\mathcal{Q}$ si, pour toute structure $\mathcal{S}$ et toute affectation s qui rendent $\mathcal{P}_1, \ldots, \mathcal{P}_l$ simultanément vraies, alors $\mathcal{Q}$ est aussi vraie.
On note alors :

$$\boxed{\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}}$$

De même, si Γ est un ensemble de formules, on note $\Gamma \vDash \mathcal{Q}$.

Voici un exemple important :

$$\boxed{\forall x \ \mathcal{P}(x) \quad \vDash \quad \exists x \ \mathcal{P}(x)}$$

Si une formule est vraie pour tous les éléments, alors elle est vraie pour au moins un élément. Noter encore une fois qu'il est important d'avoir exigé que les domaines des structures soient toujours non vides.
Il y a des conséquences logiques naturelles issues de la logique propositionnelle :

- $\mathcal{P} \wedge \mathcal{Q} \quad \vDash \quad \mathcal{P}$
- $\mathcal{P}, \ \mathcal{Q} \quad \vDash \quad \mathcal{P} \wedge \mathcal{Q}$
- $\mathcal{P}, \ \mathcal{P} \to \mathcal{Q} \quad \vDash \quad \mathcal{Q}$
- si $\mathcal{P} \vDash \mathcal{Q}$ et si $\mathcal{Q} \vDash \mathcal{R}$, alors $\mathcal{P} \vDash \mathcal{R}$

2.2. Équivalence logique

Définition.
Si $\mathcal{P} \vDash \mathcal{Q}$ et $\mathcal{Q} \vDash \mathcal{P}$, on dit que $\mathcal{P}$ et $\mathcal{Q}$ sont ***logiquement équivalentes*** et on note :

$$\boxed{\mathcal{P} \equiv \mathcal{Q}}$$

Lorsque c'est le cas, cela signifie que $\mathcal{P}$ et $\mathcal{Q}$ sont satisfaites en même temps. Une autre façon de définir l'équivalence logique serait $\vDash (\mathcal{P} \leftrightarrow \mathcal{Q})$.
On retrouve bien sûr des équivalences, issues de la logique propositionnelle, par exemple :

- $\neg(\mathcal{P} \vee \mathcal{Q}) \equiv \neg\mathcal{P} \wedge \neg\mathcal{Q}$ (loi de De Morgan),
- $\mathcal{P} \to \mathcal{Q} \equiv \neg\mathcal{P} \vee \mathcal{Q}$ (définition de l'implication),
- etc.

2.3. Équivalences avec quantificateurs

Intéressons-nous aux équivalences logiques faisant intervenir les quantificateurs.
Négation des quantificateurs.

$$\boxed{\neg\big(\forall x \ \mathcal{P}(x)\big) \quad \equiv \quad \exists x \ \neg\mathcal{P}(x)}$$

$$\boxed{\neg\big(\exists x \ \mathcal{P}(x)\big) \quad \equiv \quad \forall x \ \neg\mathcal{P}(x)}$$

On retient, un peu sommairement, que la négation de « pour tout » est « il existe » et réciproquement. Ces équivalences entraînent que si on a une formule $\mathcal{Q}$ qui contient des quantificateurs, alors on peut toujours en trouver une formule équivalente qui ne contient que le quantificateur « pour tout » (ou bien que uniquement le quantificateur « il existe »).

Exemple.

$$\begin{aligned} & \neg\big(\forall x \quad P(x) \wedge Q(x)\big) \\ \equiv \quad & \exists x \quad \neg(P(x) \wedge Q(x)) \\ \equiv \quad & \exists x \quad (\neg P(x)) \vee (\neg Q(x)) \end{aligned}$$

$$\begin{aligned} & \neg\big(\exists x \quad P(x) \to Q(x)\big) \\ \equiv \quad & \forall x \quad \neg\big(P(x) \to Q(x)\big) \\ \equiv \quad & \forall x \quad \neg\big((\neg P(x)) \vee Q(x)\big) \\ \equiv \quad & \forall x \quad P(x) \wedge (\neg Q(x)) \end{aligned}$$

Ordre des quantificateurs.
On peut intervertir deux « pour tout » consécutifs, ou deux « il existe » consécutifs :

$$\boxed{\forall x\ \forall y\ \mathcal{P}(x,y) \quad \equiv \quad \forall y\ \forall x\ \mathcal{P}(x,y)}$$

$$\boxed{\exists x\ \exists y\ \mathcal{P}(x,y) \quad \equiv \quad \exists y\ \exists x\ \mathcal{P}(x,y)}$$

Par contre on ne peut en général pas intervertir un « pour tout » et un « il existe » :

$$\boxed{\forall x\ \exists y\ \mathcal{P}(x,y) \text{ n'est pas équivalent à } \exists y\ \forall x\ \mathcal{P}(x,y)}$$

Par exemple :

$$\forall x \in \mathbb{R}\ \ \exists y \in \mathbb{R} \qquad x + y = 0 \qquad \text{est vrai,}$$

alors que

$$\exists y \in \mathbb{R}\ \ \forall x \in \mathbb{R} \qquad x + y = 0 \qquad \text{est faux.}$$

On peut aussi se convaincre avec des exemples de la vie courante : « tout étudiant a un numéro de téléphone », mais ce n'est pas vrai que « il existe un numéro de téléphone commun à tous les étudiants ».

Autres équivalences utiles.

$$\begin{aligned} \forall x \quad \big(\mathcal{P}(x) \wedge \mathcal{Q}(x)\big) \quad & \equiv \quad \big(\forall x \quad \mathcal{P}(x)\big) \wedge \big(\forall x \quad \mathcal{Q}(x)\big) \\ \exists x \quad \big(\mathcal{P}(x) \vee \mathcal{Q}(x)\big) \quad & \equiv \quad \big(\exists x \quad \mathcal{P}(x)\big) \vee \big(\exists x \quad \mathcal{Q}(x)\big) \\ \forall x \quad \big(\mathcal{P}(x) \to \mathcal{Q}\big) \quad & \equiv \quad \big(\exists x \quad \mathcal{P}(x)\big) \to \mathcal{Q} \qquad \text{si } x \text{ n'apparaît pas dans } \mathcal{Q} \end{aligned}$$

Enfin, on peut renommer une variable x en une nouvelle variable y (qui n'apparaît pas dans $\mathcal{P}(x)$) :

$$\forall x \quad \mathcal{P}(x) \quad \equiv \quad \forall y \quad \mathcal{P}[y/x]$$

2.4. Satisfaction des quantificateurs

Définissons de façon un peu plus formelle la satisfaction des quantificateurs :

$$\boxed{\vDash_{\mathcal{S},s} \forall x\ \mathcal{P}(x) \qquad \text{ssi} \qquad \text{pour tout } d \in E,\ \vDash_{\mathcal{S},s[x\mapsto d]}\ \mathcal{P}(x)}$$

La notation $s[x \mapsto d]$ est l'affectation qui envoie x sur d et coïncide avec s pour toutes les autres variables.

$$\boxed{\vDash_{\mathcal{S},s} \exists x\ \mathcal{P}(x) \qquad \text{ssi} \qquad \text{il existe un } d \in E \quad \text{tel que} \quad \vDash_{\mathcal{S},s[x\mapsto d]}\ \mathcal{P}(x)}$$

2.5. Affectation vs substitution

Cette section est plus technique et peut être passée lors d'une première lecture. Ce qu'il faut retenir, c'est que l'affectation et la substitution sont deux opérations différentes mais qu'elles sont compatibles. Considérons un langage $\mathcal{L}$, une structure $\mathcal{S}$ de domaine E. Rappel :

- la **substitution** $\mathcal{P}[t/x]$ remplace toute occurrence libre de x par le terme t. On part d'une formule $\mathcal{P}$ du langage $\mathcal{L}$ et on obtient une formule $\mathcal{P}[t/x]$ toujours dans ce même langage.
- l'**affectation** $\mathcal{P}[x \mapsto d]$ remplace toute occurrence libre de x par la valeur $d \in E$. La formule $\mathcal{P}$ est toujours une formule du langage $\mathcal{L}$, mais $\mathcal{P}[x \mapsto d]$ est une formule interprétée dans la structure $\mathcal{S}$.

Une fois qu'on a bien compris la différence, il est rassurant de savoir que faire une combinaison substitution puis affectation est équivalent à la combinaison dans l'autre sens :

$$\vDash_{\mathcal{S},s} \mathcal{P}[t/x] \qquad \text{ssi} \qquad \vDash_{\mathcal{S},s[x\mapsto t^{\mathcal{S},s}]} \mathcal{P}$$

Exemple.
Soit $\mathcal{L} = (0, 1, \leqslant, +)$ un langage. On note $2 = 1+1$, $3 = 2+1$. Considérons la formule :

$$\mathcal{P}(x) : \qquad \forall y \quad x \leqslant y + 3$$

Soit le terme $t = x + 1$.
Considérons la structure $\mathcal{S}$ de domaine $E = \mathbb{N}$ naturellement associée à $\mathcal{L}$.

- La substitution $\mathcal{P}[t/x]$ donne la formule suivante, qui reste dans le langage $\mathcal{L}$:

$$\mathcal{P}[t/x] : \qquad \forall y \quad x + 1 \leqslant y + 3$$

- On repart de $\mathcal{P}$. L'affectation $s : x \mapsto 1$ transforme une formule de $\mathcal{L}$ en une formule interprétée dans la structure :

$$\mathcal{P}[x \mapsto 1] : \qquad \forall y \in \mathbb{N} \quad 1 \leqslant y + 3$$

- Substitution puis affectation. On part de $\mathcal{P}$, on applique la substitution de x par t, pour obtenir $\mathcal{P}[t/x]$ comme ci-dessus, et ensuite on affecte 1 à la variable x. On obtient alors :

$$\mathcal{P}[t/x][x \mapsto 1] : \qquad \forall y \in \mathbb{N} \quad 2 \leqslant y + 3$$

- Affectation d'abord. On part du terme $t = x + 1$ et dans ce terme on affecte la valeur 1 à x. On obtient le terme $t^{\mathcal{S},s} = 2 \in E$. Maintenant dans $\mathcal{P}$, on remplace toute occurrence libre de x par $t^{\mathcal{S},s}$. On obtient la même formule que précédemment :

$$\mathcal{P}[x \mapsto 2] : \qquad \forall y \in \mathbb{N} \quad 2 \leqslant y + 3$$

2.6. Récapitulatif des notions

Notation	Terminologie	Description
$\vDash_{\mathcal{S}} \mathcal{Q}$	Satisfaction (formule fermée). Modèle	La formule fermée $\mathcal{Q}$ est vraie dans la structure $\mathcal{S}$. On dit que $\mathcal{S}$ est un modèle de $\mathcal{Q}$.
$\vDash_{\mathcal{S},s} \mathcal{Q}$	Satisfaction avec affectation	La formule $\mathcal{Q}$ est vraie dans $\mathcal{S}$ pour l'affectation s.
	Formule satisfaisable	Formule vraie dans au moins une structure (et une affectation si nécessaire).
$\vDash \mathcal{Q}$	Validité	Formule $\mathcal{Q}$ vraie dans toute structure et pour toute affectation.
$\vDash \neg\mathcal{Q}$	Contradiction	Formule $\mathcal{Q}$ toujours fausse, pour toute structure et toute affectation.
$\mathcal{P}_1, \ldots, \mathcal{P}_l \vDash \mathcal{Q}$	Conséquence logique	Toute structure et affectation satisfaisant simultanément les $\mathcal{P}_i$ satisfait aussi $\mathcal{Q}$.
$\mathcal{P} \equiv \mathcal{Q}$	Équivalence logique	$\mathcal{P} \vDash \mathcal{Q}$ et $\mathcal{Q} \vDash \mathcal{P}$. Pour toute structure $\mathcal{S}$ et toute affectation s, $\mathcal{P}$ et $\mathcal{Q}$ ont la même valeur de vérité.

Exercices

Vrai/faux

Exercice 1. Pour chacune des formules fermées suivantes, dire si elles sont vraies ou fausses lorsque l'on les interprète dans les différentes structures, c'est-à-dire a-t-on $\vDash_{\mathcal{S}} \mathcal{Q}$? On considère $\mathcal{L} = (a, f(x,y))$ et les formules :

$$\begin{aligned} \mathcal{Q}_1 &: \quad \forall x\ \exists y \quad f(x,y) = a \\ \mathcal{Q}_2 &: \quad \forall x\ \forall y \quad f(x,y) = a \to (x = a) \vee (y = a) \\ \mathcal{Q}_3 &: \quad \forall x \quad (x \neq a) \to (\exists y \quad f(x,y) \neq a) \end{aligned}$$

Voici les structures :

$$\begin{aligned} \mathcal{S}_i &: \quad \mathbb{N},\ a = 0,\ f(x,y) = x + y \\ \mathcal{S}_{ii} &: \quad \mathbb{N},\ a = 0,\ f(x,y) = xy \\ \mathcal{S}_{iii} &: \quad \mathbb{N},\ a = 1,\ f(x,y) = xy \end{aligned}$$

Exercice 2. On considère le langage $\mathcal{L} = (<, f(x))$ et la structure $\mathcal{S}$ de domaine $\mathbb{R}$ avec $f(x) = e^x$. Pour les formules suivantes et les affectations données, dire si l'interprétation est vraie ou fausse, c'est-à-dire a-t-on $\vDash_{\mathcal{S},s} \mathcal{Q}$?

1. $\mathcal{Q}_1$: $\forall x \quad y < f(x)$ $\qquad s_1 : y \mapsto 0$; puis $s_2 : y \mapsto e$
2. $\mathcal{Q}_2$: $\exists x \quad x < f(y)$ $\qquad s_1 : y \mapsto 0$; puis $s_2 : y \mapsto e$

3. $\mathcal{Q}_3$: $\neg\big(f(x) < f(y)\big)$ $s : x \mapsto 0, y \mapsto e$

Exercice 3. On considère le langage $\mathcal{L} = (0, 1, <, +, \times)$ et la structure $\mathcal{S}$ de domaine $\mathbb{R}$ naturellement associée. Pour chacune des formules suivantes, trouver toutes les affectations s des variables libres, qui rendent la formule $\mathcal{Q}$ vraie dans la structure :

1. $\mathcal{Q}_1$: $\forall x \quad (x \neq 0) \to x \times y \neq 0$
2. $\mathcal{Q}_2$: $\exists x \quad x \times x = y$
3. $\mathcal{Q}_3$: $\exists y \quad x < y \times y$

Pour chacune des formules suivantes, représenter dans le plan $\mathbb{R}^2$ l'ensemble des valeurs qui, affectées à (x, y), rendent la formule $\mathcal{Q}$ vraie :

4. $\mathcal{Q}_4$: $x \times x < y$
5. $\mathcal{Q}_5$: $\big(\exists x \quad x + y = 0\big) \vee \big(\forall y \quad x \times y = 0\big)$
6. $\mathcal{Q}_6$: $(x \times x = y \times y) \to (x \neq y)$

Exercice 4. Les formules suivantes sont-elles des validités (vraies quels que soient $\mathcal{S}$ et s), des contradictions (fausses quels que soient $\mathcal{S}$ et s), ou ni l'une ni l'autre (et dans ce cas justifier votre réponse) ?

1. $(\forall x\ P(x)) \to (\exists x\ P(x))$
2. $(\exists x\ P(x)) \to (\forall x\ P(x))$
3. $\forall x\ \exists y \quad (x = y) \wedge (x \neq y)$
4. $\forall x \quad Q(x, y) \to Q(y, x)$
5. $x = y$
6. $\exists y \quad (x = y) \vee (x \neq y)$

Conséquence logique

Exercice 5. Dire si les formules $\mathcal{Q}$ suivantes sont des conséquences logiques de $\mathcal{P}_1$,…,$\mathcal{P}_l$, c'est-à-dire a-t-on $\mathcal{P}_1, \ldots, \mathcal{P}_l \models \mathcal{Q}$?

1. $\mathcal{P}$: $\forall x \quad x < y$, $\mathcal{Q}$: $\exists x \quad x < y$
2. $\mathcal{P}$: $\neg(x = y)$, $\mathcal{Q}$: $x < y$
3. $\mathcal{P}_1$: $\forall x \quad P(x)$, $\mathcal{P}_2$: $\exists y \quad Q(y)$, $\mathcal{Q}$: $\exists x \quad P(x) \wedge Q(x)$
4. $\mathcal{P}_1$: $\exists x \quad P(x)$, $\mathcal{P}_2$: $\exists x \quad Q(x)$, $\mathcal{Q}$: $\exists x \quad P(x) \wedge Q(x)$

Exercice 6. Écrire une phrase en français qui explique ou justifie chaque propriété suivante :

1. $\mathcal{P} \models \mathcal{P}$
2. $\mathcal{P} \models \mathcal{Q}$ ssi $\models (\mathcal{P} \to \mathcal{Q})$
3. si $\mathcal{P} \models \mathcal{Q}$ alors $\mathcal{P}, \mathcal{R} \models \mathcal{Q}$
4. si $\mathcal{P} \models \mathcal{Q}$ et $\mathcal{Q} \models \mathcal{R}$ alors $\mathcal{P} \models \mathcal{R}$

Exercice 7. Pour chacune des formules suivantes (données dans une interprétation naturelle), écrire la négation de la phrase et dire si la phrase originale est vraie ou fausse.

1. $\exists x \in \mathbb{R} \quad x^2 = 2$
2. $\forall x \in \mathbb{N} \quad x > 2$
3. $\exists x \in]0, +\infty[\quad \forall y \in]0, +\infty[\quad x + y = 1$
4. $\forall z \in \mathbb{C} \quad z \neq 0 \to z^2 \neq 0$
5. $\forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad xy = 0 \to (x = 0) \vee (y = 0)$
6. $\forall x \in \mathbb{R} \quad x > 1 \to \big(\exists y \in \mathbb{R} \quad (y > 1) \wedge (x > y)\big)$

Exercice 8. Pour chaque paire de formules $\mathcal{P}$, $\mathcal{Q}$ dire si on a la conséquence logique $\mathcal{P} \vDash \mathcal{Q}$, ou $\mathcal{Q} \vDash \mathcal{P}$ ou l'équivalence logique $\mathcal{P} \equiv \mathcal{Q}$ ou rien du tout.

1. $\mathcal{A} \vee \mathcal{B}$ avec $\mathcal{B} \vee \mathcal{A}$
2. $\neg(\mathcal{A} \vee \mathcal{B})$ avec $(\neg\mathcal{A}) \vee (\neg\mathcal{B})$
3. $\exists x\ \neg\mathcal{A}$ avec $\forall x\ \mathcal{A}$
4. $\forall x\ \mathcal{A} \to \mathcal{B}$ avec $\exists x\ \mathcal{B} \wedge (\neg\mathcal{A})$
5. $\forall x\ \mathcal{A}$ avec $\mathcal{A}$
6. $\exists x\ \mathcal{A}$ avec $\mathcal{A}$

Exercice 9. Montrer qu'on n'a **pas** les équivalences logiques :

1. $\exists x\ \mathcal{P} \wedge \mathcal{Q}$ avec $(\exists x\ \mathcal{P}) \wedge (\exists x\ \mathcal{Q})$
2. $\forall x\ \mathcal{P} \vee \mathcal{Q}$ avec $(\forall x\ \mathcal{P}) \wedge (\forall x\ \mathcal{Q})$
3. $\forall x\ \exists y\ \mathcal{P}$ avec $\exists y\ \forall x\ \mathcal{P}$
4. $\forall x\ (\mathcal{P} \to \mathcal{Q})$ avec $(\exists x\ \mathcal{P}) \to \mathcal{Q}$

Exercice 10. Soit $\mathcal{L}$ un langage. Un *axiome* est une formule fermée $\mathcal{A}$ qu'on exige vraie pour les structures considérées, c'est-à-dire qu'on ne considère que les structures $\mathcal{S}$ telles qu'on ait $\vDash_{\mathcal{S}} \mathcal{A}$.

1. Que peut-on dire d'une structure $\mathcal{S}$ qui vérifie l'axiome
$$\mathcal{A} : \qquad \forall x\ \forall y \quad x = y$$
2. Même question avec :
$$\mathcal{A} : \qquad \exists x\ \exists y \quad \big(x \neq y\big) \wedge \big(\forall z \quad (z = x) \vee (z = y)\big)$$

Chapitre 8

Preuves

Nous adaptons la construction des preuves aux quantificateurs de la logique du premier ordre.

1. Preuves en logique du premier ordre

1.1. Rappels

On rappelle qu'une preuve consiste à partir d'un ensemble de formules $\mathcal{P}_1$, …, $\mathcal{P}_l$ jusqu'à arriver à une formule $\mathcal{Q}$ en un nombre fini d'étapes et en suivant des règles précises. Si on a obtenu une preuve, on note :

$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q}$$

Pour la logique du premier ordre, on reprend d'abord toutes les règles d'inférence des preuves de la logique V/F. On renvoie au chapitre « Preuves » de la première partie. On va ajouter de nouvelles règles liées aux quantificateurs. On rappelle qu'il n'y a encore aucun lien entre ce que l'on prouve et ce qui est vrai. Ce lien sera donné dans les chapitres suivants. En particulier dans le prochain chapitre on montrera que tout ce qui est prouvé est vrai.
Dans ce chapitre, on fixe un langage $\mathcal{L}$. Une preuve ne fait intervenir que ce langage. Il n'y a pas de structures ou d'affectations à considérer. Ainsi les formules d'une preuve restent dans $\mathcal{L}$, ce qui est un aspect où la notion de preuve ($\vdash$) est plus facile que la notion de satisfaction ($\models$).

1.2. Règles pour les quantificateurs

Nous présentons quatre nouvelles règles liées aux quantificateurs : il s'agit des règles d'introduction et d'élimination de $\forall$ et $\exists$. Une règle supplémentaire concerne l'égalité. Nous présentons chacune des règles succinctement avant d'y revenir en détails dans la section suivante. Il y a deux règles faciles et deux règles difficiles. La difficulté ne concerne pas la règle encadrée mais plutôt ses conditions d'application. Pour clarifier l'écriture des règles, lorsque qu'on a une formule $\mathcal{P}$ et qu'on s'intéresse à sa dépendance vis-à-vis de la variable x, alors on la note naturellement $\mathcal{P}(x)$. Attention, elle peut aussi dépendre d'autres variables. Pour une formule $\mathcal{P}(x)$, on notera $\mathcal{P}(t)$ au lieu de $\mathcal{P}[t/x]$, la substitution de la variable x par le terme t.

Règle I. Élimination de $\forall$.

$$\boxed{\forall x \;\; \mathcal{P}(x) \quad \vdash \quad \mathcal{P}(c)}$$

- *Conditions.* c peut être une constante quelconque du langage, c peut aussi être un *terme clos*, c'est-à-dire un terme sans variable libre.

- *Remarque.* Ainsi sous l'hypothèse $\forall x \;\; \mathcal{P}(x)$, on peut remplacer n'importe quelle occurrence libre de x par c et cela prouve $\mathcal{P}(c)$ (qui est en fait $\mathcal{P}[c/x]$). Ici cela ne pose de problème si la constante (ou le terme) c est déjà apparu auparavant dans la preuve.
- *Exemple.* Pour n'importe quelle constante c :

$$\forall x \;\; \exists y \;\; P(x) \wedge Q(x,y) \quad \vdash \quad \exists y \;\; P(c) \wedge Q(c,y)$$

Règle II. Introduction de $\forall$.

$$\boxed{\mathcal{P}(c) \quad \vdash \quad \forall x \;\; \mathcal{P}(x)}$$

- *Conditions.* c désigne une constante de $\mathcal{L}$, mais elle doit être *arbitraire*, c'est-à-dire qu'elle ne doit apparaître dans aucune hypothèse en cours d'utilisation.
- *Remarque.* C'est une règle piégeuse car ici c ne joue pas vraiment le rôle d'une constante mais plutôt d'une *variable libre*. Le mieux c'est d'utiliser un nom c qui n'apparaît pas du tout dans les lignes précédentes de la preuve.
- *Exemple.* Soit c arbitraire :

$$c^2 \geqslant 0 \quad \vdash \quad \forall x \;\; x^2 \geqslant 0$$

  En effet, si on a prouvé $c^2 \geqslant 0$ pour un c quelconque, alors pour tout x, $x^2 \geqslant 0$.

Règle III. Introduction de $\exists$.

$$\boxed{\mathcal{P}(c) \quad \vdash \quad \exists x \;\; \mathcal{P}(x)}$$

- *Conditions.* c est une constante du langage ou bien un terme sans variable libre.
- *Remarque.* c est appelé un *témoin* du quantificateur $\exists$. Ce symbole c a pu apparaître auparavant dans la preuve.
- *Exemple.*

$$c^2 \neq c \quad \vdash \quad \exists x \;\; x^2 \neq x$$

Règle IV. Élimination de $\exists$.

$$\boxed{\exists x \;\; \mathcal{P}(x), \left[\mathcal{P}(c) \vdash \mathcal{Q}\right] \quad \vdash \quad \mathcal{Q}}$$

La notation entre crochets correspond à une sous-preuve. C'est plus clair avec l'écriture sous forme d'un tableau :

$$\begin{array}{ll} & \exists x \;\; \mathcal{P}(x) \\ & \hline \\ & \quad \mathcal{P}(c) \qquad \text{Sous-hypothèse} \\ & \quad \hline \\ & \quad \vdots \\ & \quad \mathcal{Q} \\ \vdash & \mathcal{Q} \end{array}$$

- *Conditions.* c doit être arbitraire : elle ne doit apparaître ni dans $\mathcal{Q}$, ni dans $\exists x \;\; \mathcal{P}(x)$, ni dans aucune hypothèse en cours d'utilisation. On parle de *constante fraîche* (ou abusivement de *variable fraîche*).
- *Remarque.* La condition n'est pas évidente au premier abord, mais le principe est naturel : si l'on sait qu'il existe un élément vérifiant $\mathcal{P}$, et que le fait de supposer $\mathcal{P}(c)$ pour un c arbitraire conduit à $\mathcal{Q}$, alors la conclusion $\mathcal{Q}$ est prouvée.

- *Exemple.*
$$\exists x \;\; f(x)=0,\ \forall x \;\; \big(f(x)=0 \to g(x)=1\big) \quad \vdash \quad \exists x \;\; g(x)=1$$
C'est une application presque directe de la règle IV, voir la section suivante pour les détails.

Règle V. Égalité.

$$\boxed{t=s,\ \ \mathcal{P}(t) \quad \vdash \quad \mathcal{P}(s)}$$

- *Conditions.* t et s sont des termes du langage. Il faut bien sûr que les substitutions par t ou s soient légitimes (pas de capture).
- *Remarque.* Il n'y a pas de piège avec l'égalité. Deux choses égales sont considérées comme identiques et donc interchangeables à volonté.
- *Exemple.*
$$a=x+1,\ a>7 \quad \vdash \quad x+1>7$$

1.3. Exemples de preuves

Exemple.

$$\forall x \;\; \mathcal{P}(x) \quad \vdash \quad \exists x \;\; \mathcal{P}(x)$$

	$\forall x \;\; \mathcal{P}(x)$	Hypothèse
	$\mathcal{P}(c)$	Élimination du $\forall$
$\vdash$	$\exists x \; \mathcal{P}(x)$	Introduction du $\exists$

Les règles de preuve supposent implicitement que l'on travaille avec des ensembles non vides. C'est cohérent avec le domaine d'une structure qui doit être un ensemble non vide.

Exemple.
Soit une formule $\mathcal{P}$ dans laquelle la variable x n'apparaît pas.

$$\forall x \;\; \big(\mathcal{P} \to \mathcal{Q}(x)\big) \quad \vdash \quad \mathcal{P} \to \big(\forall x \;\; \mathcal{Q}(x)\big)$$

1	$\forall x \;\; \big(\mathcal{P} \to \mathcal{Q}(x)\big)$	
2	$\quad \mathcal{P}$	Sous-hypothèse
3	$\quad \mathcal{P} \to \mathcal{Q}(c)$	Élimination du $\forall$ de 1
4	$\quad \mathcal{Q}(c)$	
5	$\quad \forall x \;\; \mathcal{Q}(x)$	Introduction du $\forall$, c n'apparait dans aucune hypothèse
$\vdash$	$\mathcal{P} \to \big(\forall x \;\; \mathcal{Q}(x)\big)$	

Exemple.

$$\forall x\ \big(\mathcal{P}(x) \to \mathcal{Q}(x)\big),\ \exists x\ \mathcal{P}(x) \quad \vdash \quad \exists x\ \mathcal{Q}(x)$$

C'est assez naturel : si $\mathcal{P}(x)$ implique toujours $\mathcal{Q}(x)$ et que l'un des $\mathcal{P}(x)$ est vrai, alors l'un des $\mathcal{Q}(x)$ est vrai. C'est une variante du *modus ponens* de la logique propositionnelle : $\mathcal{P}, \mathcal{P} \to \mathcal{Q} \ \vdash \ \mathcal{Q}$.

1	$\forall x\ \big(\mathcal{P}(x) \to \mathcal{Q}(x)\big)$	
2	$\exists x\ \mathcal{P}(x)$	
3	$\quad \mathcal{P}(c)$	Sous-hypothèse
4	$\quad \mathcal{P}(c) \to \mathcal{Q}(c)$	Élimination du $\forall$ de 1
5	$\quad \mathcal{Q}(c)$	
6	$\quad \exists x\ \mathcal{Q}(x)$	Introduction du $\exists$
$\vdash$	$\exists x\ \mathcal{Q}(x)$	Élimination du $\exists$ de 2 par 3-6

Exemple.

$$\exists x\ \forall y\ \mathcal{P}(x,y) \quad \vdash \quad \forall y\ \exists x\ \mathcal{P}(x,y)$$

On peut donc intervertir un $\exists\ \forall$ vers un $\forall\ \exists$ (mais pas dans l'autre sens).

1	$\exists x\ \forall y\ \mathcal{P}(x,y)$	
2	$\quad \forall y\ \mathcal{P}(a,y)$	Avec a arbitraire
3	$\quad \mathcal{P}(a,b)$	Avec b arbitraire
4	$\quad \exists x\ \mathcal{P}(x,b)$	Introduction du $\exists$
5	$\quad \forall y\ \big(\exists x\ \mathcal{P}(x,y)\big)$	Introduction du $\forall$
$\vdash$	$\forall y\ \exists x\ \mathcal{P}(x,y)$	Élimination du $\exists$ de 1 par 2-5

2. Détails des règles

On reprend les règles une par une avec davantage d'explications et des exemples.

2.1. Règle I. Élimination de $\forall$

$$\boxed{\forall x\ \mathcal{P}(x) \quad \vdash \quad \mathcal{P}(c)}$$

Conditions.

- Si c est une constante du langage alors il n'y a pas de condition si la substitution de x par c est valide.
- Si c est un terme, aucune de ses variables ne doit être liée (capturée) par un quantificateur de $\mathcal{P}$ lors de la substitution.

Autres formulations.

$$\forall x \ \mathcal{P}(x) \quad \vdash \quad \mathcal{P}[c/x]$$

On trouve aussi parfois l'écriture un peu abusive : $\forall x \ \mathcal{P}(x) \vdash \mathcal{P}(x)$.
Et voici le schéma en déduction naturelle :

$$\begin{array}{l|l} & \forall x \ \mathcal{P}(x) \\ \hline \vdash & \mathcal{P}(c) \end{array}$$

Exemples.

- $\forall x \ P(x) \wedge Q(x,c) \ \vdash \ P(c) \wedge Q(c,c)$ est une preuve correcte. On a remplacé les occurrences de x par une constante c, même si cette constante apparaissait déjà dans la formule.
- $\forall x \ P(x) \ \vdash \ P(f(a,b))$ est correct si f est une fonction et a, b deux constantes.
- $\forall x \ P(x) \ \vdash \ P(y)$ est aussi correct, à condition que y ne soit pas capturée par un quantificateur dans $\mathcal{P}$.

Contre-exemples.

- $\big(\forall x \ P(x)\big) \to Q(a,b)$ ne prouve pas $P(c) \to Q(a,b)$, car le « pour tout » n'est pas le connecteur principal de la phrase. Par contre $\forall x \ \big(P(x) \to Q(a,b)\big)$ prouverait bien $P(c) \to Q(a,b)$.
- $\forall x \ \exists y \ Q(x,y)$ ne prouve pas $\exists y \ Q(y,y)$. En effet, dans la formule « $\exists y \ Q(x,y)$ », la variable x apparaît comme variable libre. La substitution de x par y est donc interdite car la variable y injectée lors du remplacement serait capturée par le quantificateur $\exists y$ de la formule.

Remarques. Comment parler d'une constante c, si le langage $\mathcal{L}$ n'a pas de constante à ce nom ou bien n'a pas de constante du tout ? On s'autorise à étendre le langage $\mathcal{L}$ en un langage $\mathcal{L}' = \mathcal{L} \cup \{c\}$ où c est un nouveau symbole de constante.

2.2. Règle II. Introduction de $\forall$

$$\boxed{\mathcal{P}(c) \quad \vdash \quad \forall x \ \mathcal{P}(x)}$$

Conditions. Ce qui est délicat dans de son usage, c'est que c doit être une constante arbitraire, c'est-à-dire une constante dont le symbole n'apparaît dans aucune hypothèse en cours d'utilisation.
Autres formulations.

$$\mathcal{P}(x) \quad \vdash \quad \forall x \ \mathcal{P}(x)$$

$$\begin{array}{l|l} & \mathcal{P}(c) \\ \hline \vdash & \forall x \ \mathcal{P}(x) \end{array}$$

Soit Γ un ensemble de formules et c une constante qui n'apparaît pas dans les formules de Γ.

$$\text{Si } \Gamma \ \vdash \ \mathcal{P}(c) \quad \text{alors} \quad \Gamma \ \vdash \ \forall x \ \mathcal{P}(x)$$

Exemples.

- $c > 0 \ \vdash \ \forall x \ x > 0$, à condition qu'aucune hypothèse ne porte sur c.
- $f(a,b) \neq 0 \ \vdash \ \forall x \ \forall y \ f(x,y) \neq 0$, à condition qu'aucune hypothèse ne porte ni sur a, ni sur b.

Contre-exemples. Beaucoup de choses sont interdites et il est facile de se tromper dans l'application de cette règle.

- $c > 0, \ P(c)$ ne prouve pas $\forall x \ P(x)$ car il y a la contrainte $c > 0$ qui fait que c n'est pas une constante arbitraire. Par contre, à partir des hypothèses $c > 0$ et $P(c)$, on pourrait prouver que $\forall x \ \big(x > 0 \to P(x)\big)$.

- Voici quasiment le même exemple avec l'écriture en déduction naturelle et un autre exemple :

$$\begin{array}{lll}
\left|\begin{array}{l} c>0 \\ \hline P(c) \end{array}\right. & \left|\begin{array}{l} \forall x\ \ Q(c,x) \\ \hline Q(c,c) \end{array}\right. & \text{Élimination de } \forall \text{ correcte} \\
\vdash\ \left|\begin{array}{l} \cancel{\forall x\ P(x)} \end{array}\right. & \vdash\ \left|\begin{array}{l} \cancel{\forall x\ Q(x,x)} \end{array}\right. & \text{Introduction de } \forall \text{ non valide}
\end{array}$$

- Il est interdit de ne remplacer que certaines occurrences de c. Par exemple, $c = c$ ne prouve pas $\forall x\ x = c$.

Remarque. Le mieux est que le symbole c utilisé dans la règle n'apparaisse pas du tout auparavant dans la preuve. Mais ce n'est pas toujours aussi simple. Voici un exemple valide :

$$\begin{array}{lll}
\left|\begin{array}{l} \vdots \\ \left|\begin{array}{l} P(c) \\ \hline Q(c) \end{array}\right. \\ P(c)\to Q(c) \end{array}\right. & & \begin{array}{l} c \text{ n'apparaît pas avant} \\ \text{Sous-hypothèse, condition porte sur } c \\ \text{Conclusion dans ce cas} \\ \text{Il n'y a plus aucune hypothèse en cours d'utilisation contenant } c \end{array} \\
\vdash\ \left|\begin{array}{l} \forall x\ \ \big(P(x)\to Q(x)\big) \end{array}\right. & & \text{Introduction de } \forall
\end{array}$$

2.3. Règle III. Introduction de $\exists$

$$\boxed{\mathcal{P}(c)\ \ \vdash\ \ \exists x\ \ \mathcal{P}(x)}$$

Conditions. c est une constante ou bien un terme sans variable libre.

Autres formulations.

$$\mathcal{P}[c/x]\ \ \vdash\ \ \exists x\ \ \mathcal{P}(x)$$

$$\vdash\ \left|\begin{array}{l} \mathcal{P}(c) \\ \hline \exists x\ \ \mathcal{P}(x) \end{array}\right.$$

Exemples.

- $c^2 \neq c\ \ \vdash\ \ \exists x\ x^2 \neq x$ (déjà vu).
- $c^2 \neq c\ \ \vdash\ \ \exists x\ x^2 \neq c$ (remplacement partiel autorisé)
- $c^2 \neq c\ \ \vdash\ \ \exists x\ c^2 \neq x$ (remplacement partiel)
- $a \times b = 12\ \ \vdash\ \ \exists x\ \exists y\ x \times y = 12$ (deux utilisations de la règle)

Remarque. Il n'y a pas de danger avec cette règle. Il faut cependant prendre soin d'introduire le « il existe » en tête de la formule obtenue. Par exemple, ceci est correct :

$$\forall x\ \ Q(x,b)\ \ \vdash\ \ \exists y\ \ \forall x\ \ Q(x,y)$$

Mais la même prémisse ne prouverait pas $\forall x\ \ \exists y\ \ Q(x,y)$.

Domaine non vide. En logique du premier ordre, on supposera toujours qu'on travaille avec un domaine non vide.

2.4. Règle IV. Élimination de ∃

$$\boxed{\exists x \;\; \mathcal{P}(x),\ \big[\mathcal{P}(c) \vdash \mathcal{Q}\big] \quad \vdash \quad \mathcal{Q}}$$

La notation entre crochets correspond à une sous-preuve.

Conditions. c doit être une ***constante fraîche*** : d'une part elle ne doit apparaître dans aucune hypothèse en cours d'utilisation, d'autre part elle ne doit pas apparaître dans la formule $\mathcal{Q}$ (ni bien sûr dans $\mathcal{P}(x)$). Le mieux est que la constante n'apparaisse ni avant dans la preuve, ni dans $\mathcal{Q}$.
Autres formulations.

$$\exists x \;\; \mathcal{P}(x),\ \big[\mathcal{P}[c/x] \vdash \mathcal{Q}\big] \quad \vdash \quad \mathcal{Q}$$
$$\exists x \;\; \mathcal{P}(x),\ \big[\mathcal{P}(x) \vdash \mathcal{Q}\big] \quad \vdash \quad \mathcal{Q}$$

Soit Γ un ensemble de formules, c une constante du langage qui n'apparaît ni dans $\mathcal{Q}$, ni dans les formules de Γ :

$$\text{Si} \quad \Gamma \vdash \exists x \;\; \mathcal{P}(x) \quad \text{et} \quad \mathcal{P}(c) \vdash \mathcal{Q}, \quad \text{alors } \Gamma \vdash \mathcal{Q}$$

Le plus parlant est l'écriture en déduction naturelle.

$$\begin{array}{lll}
 & \exists x \;\; \mathcal{P}(x) & \\
 & \hline
 & \quad \mathcal{P}(c) & \text{Sous-hypothèse} \\
 & \quad \hline
 & \quad \vdots & \\
 & \quad \mathcal{Q} & \\
\vdash & \mathcal{Q} &
\end{array}$$

Exemple. Voyons comment utiliser cette règle dans la pratique :

$$\exists x \;\; f(x) = 0,\ \forall x \;\; \big(f(x) = 0 \to g(x) = 1\big) \quad \vdash \quad \exists x \;\; g(x) = 1$$

Ce n'est pas une application directe de la règle, mais presque :

$$\begin{array}{lll}
 & \exists x \;\; f(x) = 0 & \\
 & \forall x \;\; \big(f(x) = 0 \to g(x) = 1\big) & \\
 & \hline
 & \quad f(c) = 0 & \text{Sous-hypothèse} \\
 & \quad \hline
 & \quad f(c) = 0 \to g(c) = 1 & \text{Élimination de } \forall \\
 & \quad g(c) = 1 & \text{Par l'implication} \\
 & \quad \exists x \;\; g(x) = 1 & \text{C'est la formule } \mathcal{Q} \text{ de la règle} \\
\vdash & \exists x \;\; g(x) = 1 & \text{Élimination de } \exists
\end{array}$$

Remarque. Si on enchaîne plusieurs « il existe », on doit prendre une nouvelle constante fraîche pour chacun. Par exemple, si on a :

$$\exists x \;\; P(x), \;\; \exists y \;\; Q(y)$$

alors dans une preuve on ne peut pas choisir $P(c)$ et $Q(c)$ avec le même c. On doit utiliser deux constantes fraîches c et d afin d'écrire $P(c)$ et $Q(d)$ dans les sous-hypothèses.

2.5. Règle V. Égalité

$$\boxed{t = s,\ \mathcal{P}(t) \quad \vdash \quad \mathcal{P}(s)}$$

Conditions. t et s sont des termes du langage. La substitution de x par t et s doit être valide. Il n'y a pas d'autres conditions.
Autres formulations.

$$\begin{array}{rl} & t = s \\ & \mathcal{P}(t) \\ \hline \vdash & \mathcal{P}(s) \end{array}$$

$$x = y,\ \mathcal{P}(x) \quad \vdash \quad \mathcal{P}(y)$$

Exemple. $t = x + 1,\ t^2 \neq 0 \ \vdash \ (x+1)^2 \neq 0$.

Remarque. Cette règle peut être remplacée par les axiomes suivants pour le symbole égalité :

- $\forall x \ \ x = x$
- $\forall x \ \ \forall y \ \ \big((x = y) \to (\mathcal{P}(x) \leftrightarrow \mathcal{P}(y))\big)$

3. Négation des quantificateurs et règles dérivées

3.1. Négation des quantificateurs

Règle dérivée. Négation des quantificateurs.

$$\boxed{\neg\big(\forall x \ \ \mathcal{P}(x)\big) \quad \vdash \quad \exists x \ \ \big(\neg\mathcal{P}(x)\big)}$$

$$\boxed{\neg\big(\exists x \ \ \mathcal{P}(x)\big) \quad \vdash \quad \forall x \ \ \big(\neg\mathcal{P}(x)\big)}$$

Nous ne sommes pas surpris : la négation d'un « pour tout » est un « il existe », et réciproquement.
Exemples :

1. $\neg\big(\forall x \ \ f(x) = g(x)\big) \quad \vdash \quad \exists x \ \ \big(f(x) \neq g(x)\big)$
2. $\neg\big(\exists x \ \ P(x) \wedge Q(x)\big) \quad \vdash \quad \forall x \ \ \big(\neg P(x)\big) \vee \big(\neg Q(x)\big)$
3. $\neg\big(\forall x \ \ P(x) \to Q(x)\big) \quad \vdash \quad \exists x \ \ P(x) \wedge \big(\neg Q(x)\big)$

3.2. Autres règles dérivées

Règle dérivée. Renommage.

$$\boxed{\forall x \ \ \mathcal{P}(x) \quad \vdash \quad \forall y \ \ \mathcal{P}(y) \qquad \text{et} \qquad \exists x \ \ \mathcal{P}(x) \quad \vdash \quad \exists y \ \ \mathcal{P}(y)}$$

La variable y ne doit pas apparaître comme variable libre dans la formule $\forall x \ \mathcal{P}(x)$ ni dans $\exists x \ \mathcal{P}(x)$.

Règle dérivée. Interversion de deux $\forall$ ou deux $\exists$.

$$\boxed{\forall x \ \ \forall y \ \ \mathcal{P}(x, y) \quad \vdash \quad \forall y \ \ \forall x \ \ \mathcal{P}(x, y)}$$

$$\boxed{\exists x\ \exists y\ \mathcal{P}(x,y) \quad \vdash \quad \exists y\ \exists x\ \mathcal{P}(x,y)}$$

Règle dérivée. Interversion de $\exists\ \forall$ vers $\forall\ \exists$.

$$\boxed{\exists x\ \forall y\ \mathcal{P}(x,y) \quad \vdash \quad \forall y\ \exists x\ \mathcal{P}(x,y)}$$

On l'a prouvée dans un exemple précédent.

Règle dérivée. Commutation de $\forall$ et $\wedge$.

$$\boxed{\forall x\ \big(\mathcal{P}(x) \wedge \mathcal{Q}(x)\big) \quad \vdash \quad \big(\forall x\ \mathcal{P}(x)\big) \wedge \big(\forall x\ \mathcal{Q}(x)\big)}$$

$$\boxed{\big(\forall x\ \mathcal{P}(x)\big) \wedge \big(\forall x\ \mathcal{Q}(x)\big) \quad \vdash \quad \forall x\ \big(\mathcal{P}(x) \wedge \mathcal{Q}(x)\big)}$$

Ainsi on peut passer d'un sens à l'autre. Les formules sont interdérivables.

Règle dérivée. Commutation de $\exists$ et $\vee$.

$$\boxed{\exists x\ \big(\mathcal{P}(x) \vee \mathcal{Q}(x)\big) \quad \vdash \quad \big(\exists x\ \mathcal{P}(x)\big) \vee \big(\exists x\ \mathcal{Q}(x)\big)}$$

$$\boxed{\big(\exists x\ \mathcal{P}(x)\big) \vee \big(\exists x\ \mathcal{Q}(x)\big) \quad \vdash \quad \exists x\ \big(\mathcal{P}(x) \vee \mathcal{Q}(x)\big)}$$

3.3. Preuve des règles dérivées

Voyons quelques preuves des règles dérivées.

Exemple.
Les règles de renommage sont naturelles. Les preuves sont de simples jeux d'écriture.

$$\forall x\ \mathcal{P}(x) \quad \vdash \quad \forall y\ \mathcal{P}(y)$$

	$\forall x\ \mathcal{P}(x)$	
	$\mathcal{P}(c)$	Élimination du $\forall$, c arbitraire
$\vdash$	$\forall y\ \mathcal{P}(y)$	Introduction du $\forall$

$$\exists x\ \mathcal{P}(x) \quad \vdash \quad \exists y\ \mathcal{P}(y)$$

	$\exists x\ \mathcal{P}(x)$	
	$\mathcal{P}(c)$	c constante
	$\exists y\ \mathcal{P}(y)$	Introduction du $\exists$
$\vdash$	$\exists y\ \mathcal{P}(y)$	Élimination du $\exists$ initial

Exemple.

$$\forall x \ \forall y \ \mathcal{P}(x,y) \quad \vdash \quad \forall y \ \forall x \ \mathcal{P}(x,y)$$

	$\forall x \ \forall y \ \mathcal{P}(x,y)$	
	$\forall y \ \mathcal{P}(a,y)$	Élimination du premier $\forall$, a arbitraire
	$\mathcal{P}(a,b)$	Élimination du second $\forall$, b arbitraire
	$\forall x \ \mathcal{P}(x,b)$	Introduction d'un $\forall$
$\vdash$	$\forall y \ \forall x \ \mathcal{P}(x,y)$	Introduction d'un autre $\forall$

Exemple.

$$\big(\forall x \ \mathcal{P}(x)\big) \wedge \big(\forall x \ \mathcal{Q}(x)\big) \quad \vdash \quad \forall x \ \big(\mathcal{P}(x) \wedge \mathcal{Q}(x)\big)$$

Voici la preuve, à vous d'expliciter les règles utilisées.

	$\big(\forall x \ \mathcal{P}(x)\big) \wedge \big(\forall x \ \mathcal{Q}(x)\big)$
	$\forall x \ \mathcal{P}(x)$
	$\mathcal{P}(c)$
	$\forall x \ \mathcal{Q}(x)$
	$\mathcal{Q}(c)$
	$\mathcal{P}(c) \wedge \mathcal{Q}(c)$
$\vdash$	$\forall x \ \big(\mathcal{P}(x) \wedge \mathcal{Q}(x)\big)$

Exemple.

$$\neg \big(\forall x \ \mathcal{P}(x)\big) \quad \vdash \quad \exists x \ \big(\neg \mathcal{P}(x)\big)$$

1	$\neg\big(\forall x\ \mathcal{P}(x)\big)$	
2	$\quad\neg\big(\exists x\ \neg\mathcal{P}(x)\big)$	Raisonnement par l'absurde
3	$\quad\quad\neg\mathcal{P}(c)$	Hypothèse, avec c arbitraire
4	$\quad\quad\exists x\ \neg\mathcal{P}(x)$	Introduction de $\exists$
5	$\quad\quad\bot$	Contradiction 2 avec 4
6	$\quad\mathcal{P}(c)$	Par 2-5
7	$\quad\forall x\ \mathcal{P}(x)$	Introduction de $\forall$
8	$\quad\bot$	Contradiction 1 avec 7
$\vdash$	$\exists x\ \neg\mathcal{P}(x)$	Par contradiction de 2-8

Exercices

Preuves en logique du premier ordre

Exercice 1. Écrire les preuves des théorèmes suivants. P, Q, R désignent des prédicats, f, g des fonctions.

1. $\forall x\ P(x),\ \ P(a) \to Q(a)\ \ \vdash\ \ Q(a)$
2. $\forall x\ P(x) \wedge Q(x)\ \ \vdash\ \ \exists x\ P(x)$
3. $\forall x\ \forall y\ R(x,y)\ \ \vdash\ \ \forall x\ R(x,x)$
4. $\exists x\ f(x) = g(x)\ \ \vdash\ \ \exists x\ \exists y\ f(x) = g(y)$

Exercice 2. Écrire les preuves des théorèmes suivants. P, Q, R désignent des prédicats, f une fonction.

1. $\vdash \forall x\ \ \forall y\ \ \big(f(x) \neq f(y)\ \to\ x \neq y\big)$
2. $\forall x\ P(x),\ \forall y\ Q(y)\ \ \vdash\ \ \exists z\ P(z) \wedge Q(z)$
3. $\forall x\ P(x) \to Q(x),\ \ \forall y\ Q(y) \to R(y)\ \ \vdash\ \ \forall z\ P(z) \to R(z)$
4. $\forall x\ \big(P(x) \to Q\big),\ \ \exists x\ P(x)\ \ \vdash\ \ Q$, où Q ne dépend pas de x.

Détails des règles

Exercice 3. Expliquer l'erreur faite en appliquant la règle d'élimination de $\forall$.

1. $\exists x\ \forall y\ Q(x,y)$ prouve directement $\exists x\ Q(x,x)$. Bonus : donner une preuve correcte.
2. $\forall x\ \big(P(x) \to Q(c)\big)$ prouve $P(c)$.
3. $\forall x\ \exists y\ Q(x,y)$ prouve $\exists y\ Q(y,y)$.

Exercice 4. Expliquer l'erreur faite en appliquant la règle d'introduction de $\forall$.

1. $c = 1,\ \ c > 0$ prouve $\forall x\ x > 0$.
2. $\exists x\ P(x)$ prouve $\forall x\ P(x)$.
3. $(x \neq 0) \to (x^2 \neq 0)$ prouve $\forall x\ x^2 \neq 0$.
4. $(x = 0) \vee (x \neq 0)$ prouve $\big(\forall x\ x = 0\big) \vee \big(\forall x\ x \neq 0\big)$.
5. $f(c) = g(c)$ prouve $\forall x\ f(x) = g(c)$.

Exercice 5. Expliquer l'erreur faite en appliquant la règle d'introduction de $\exists$.

1. $\forall x\ P(x)$ prouve directement $\exists x\ P(x)$.
2. $P(a) \wedge Q(a)$ prouve $\exists x\ P(x) \wedge Q(y)$.
3. $c = 0,\ \ \exists x\ P(x)$ prouve $\exists x\ (x = 0) \wedge P(x)$.
4. $\forall x\ Q(y, y)$ prouve $\exists x\ \forall y\ Q(x, y)$.
5. $\forall x\ Q(y, y)$ prouve $\forall y\ \exists x\ Q(x, y)$.

Exercice 6. Expliquer l'erreur faite en appliquant la règle d'élimination de $\exists$.

1. $\exists x\ P(x),\ \ P(0) \to Q$ prouve Q.
2. $\exists x\ P(x)$ prouve $P(c)$.
3. $\exists x\ \big(P(x) \to Q\big),\ \ P(c)$ prouve Q.
4. $\exists x\ P(x),\ \ \exists y\ Q(y)$ prouve $\exists z\ P(z) \wedge Q(z)$.

Négation des quantificateurs et règles dérivées

Exercice 7. Écrire ce que prouvent les formules suivantes. La formule prouvée ne doit plus contenir aucune négation à gauche d'un quantificateur.

1. $\neg\Big(\forall x\ f(x) = g(x)\Big)$
2. $\neg\Big(\exists x\ P(x) \leftrightarrow Q(x)\Big)$
3. $\neg\big[\forall x\ \big(P(x) \to \exists y\ Q(y)\big)\big]$

Exercice 8. Prouver toutes les règles dérivées à l'aide des règles de base.

Chapitre 9

Théorème de validité

*Nous faisons le lien entre « preuve » et « vérité » grâce au théorème de validité (*Soundness Theorem*) qui affirme que toute formule prouvable est une formule vraie.*

1. Le théorème de validité

1.1. Énoncé

Soit $\mathcal{L}$ un langage fixé pour la logique du premier ordre.

Théorème 1 (Théorème de validité (*Soundness Theorem*))**.**

$$\boxed{\textit{Si} \quad \mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q} \quad \textit{alors} \quad \mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}.}$$

Et plus généralement, si Γ est un ensemble de formules (éventuellement infini) :

$$\text{Si} \quad \Gamma \quad \vdash \quad \mathcal{Q} \quad \text{alors} \quad \Gamma \quad \vDash \quad \mathcal{Q}.$$

Il s'agit d'un théorème de bon sens : une formule prouvée est une formule vraie ! C'est un résultat implicite dans la thèse de Gödel de 1930 où il se concentre sur la réciproque (le théorème de complétude). L'idée est que les règles d'inférence pour les preuves ont été bien choisies car chacune des règles respecte le caractère V/F de la logique du premier ordre.

1.2. Rappels

Rappelons aussi que,

$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q}$$

signifie que l'on peut déduire $\mathcal{Q}$ des formules $\mathcal{P}_1, \ldots, \mathcal{P}_l$ par un nombre fini d'étapes, où chaque étape consiste à appliquer une règle d'inférence (parmi un choix fini bien délimité).

Rappelons aussi que,

$$\mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vDash \quad \mathcal{Q}$$

signifie que, pour toute structure $\mathcal{S}$ et toute affectation s, si $\mathcal{P}_1, \ldots, \mathcal{P}_l$ sont vraies dans cette interprétation, alors $\mathcal{Q}$ est aussi vraie dans cette interprétation.

Ainsi le théorème de validité affirme qu'une formule prouvable est vraie dans toute interprétation, donc indépendamment de la structure choisie.

On peut le reformuler ainsi : si $\mathcal{P} \vdash \bot$ alors $\mathcal{P}$ est insatisfaisable (c'est-à-dire que $\mathcal{P}$ est une contradiction), autrement dit quelles que soient $\mathcal{S}$ et s, l'interprétation de $\mathcal{P}$ est fausse. On rappelle que $\bot$ désigne la formule toujours fausse. Voici la preuve de cette reformulation : supposons qu'il existe $\mathcal{S}$ et s pour lesquelles $\mathcal{P}$ soit vraie. Comme dans n'importe quelle interprétation $\bot$ est fausse, on obtient une contradiction avec $\mathcal{P} \vDash_{\mathcal{S},s} \bot$. Ainsi $\mathcal{P}$ n'est pas satisfaisable.

1.3. Applications

Expliquons l'utilité du théorème de validité.

1. En un sens, $\mathcal{P} \vdash \mathcal{Q}$ est quelque chose de facile. En effet, il s'agit de construire une preuve. Même si trouver une preuve n'est pas toujours une tâche aisée, loin de là, cela reste un processus avec un nombre fini d'étapes et un nombre fini de règles possibles.
2. Par contre, montrer qu'il n'existe pas de preuve que $\mathcal{P}$ entraîne $\mathcal{Q}$, c'est-à-dire $\mathcal{P} \nvdash \mathcal{Q}$ est difficile. Ce n'est pas parce qu'on n'a pas réussi à obtenir une preuve qu'il n'en existe pas.
3. D'un autre côté, montrer que $\mathcal{P} \models \mathcal{Q}$ n'est pas évident, car pour cela il faut tester une infinité de structures et d'affectations imaginables.
4. En revanche, obtenir $\mathcal{P} \not\models \mathcal{Q}$ est facile. Il s'agit de fournir un contre-exemple en explicitant une structure et une affectation telles que $\mathcal{P}$ soit vraie et $\mathcal{Q}$ soit fausse dans cette interprétation.

Bilan : le théorème de validité peut aider pour les points difficiles, dans son sens direct ou sa contraposée :

$$\begin{array}{llllll} \text{Si} & \mathcal{P} & \vdash & \mathcal{Q} & \text{alors} & \mathcal{P} \models \mathcal{Q}. \\ \text{Si} & \mathcal{P} & \not\models & \mathcal{Q} & \text{alors} & \mathcal{P} \nvdash \mathcal{Q}. \end{array}$$

2. Preuves du théorème de validité

Nous allons prouver que chacune des règles d'inférence respecte la logique vrai/faux et la satisfaction des formules. Comme une preuve est une succession finie de ces règles, on en déduira par récurrence le théorème de validité.

2.1. Règles de la logique V/F

On renvoie au chapitre « Preuves » de la première partie pour la liste complète de toutes ces règles. On se contente d'en donner quelques exemples.

- On a la règle « $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$ » et on a bien la satisfaction « $\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$ ».
- On a la règle « $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$ » et on a bien la satisfaction « $\mathcal{P}, \mathcal{Q} \models \mathcal{P} \wedge \mathcal{Q}$ ».
- On a la règle « $\neg\neg\mathcal{P} \vdash \mathcal{P}$ » et on a bien la satisfaction « $\neg\neg\mathcal{P} \models \mathcal{P}$ ».
- On a la règle « $\bot \vdash \mathcal{P}$ » et on a bien la satisfaction « $\bot \models \mathcal{P}$ ».
- On a la règle « $\mathcal{P}, \mathcal{P} \to \mathcal{Q} \vdash \mathcal{Q}$ » et on a bien la satisfaction « $\mathcal{P}, \mathcal{P} \to \mathcal{Q} \models \mathcal{Q}$ ».

Il n'y a aucune difficulté pour justifier ces relations de satisfaction à l'aide de tables de vérité. Par exemple, pour prouver $\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$, on fixe une structure $\mathcal{S}$ et une affectation s et on dresse la table de vérité :

$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{P} \wedge \mathcal{Q}$	$\mathcal{P}$
V	V	V	V
V	F	F	–
F	V	F	–
F	F	F	–

On constate que chaque fois que $\mathcal{P} \wedge \mathcal{Q}$ est vraie, alors $\mathcal{P}$ est aussi vraie. Ainsi

$$\mathcal{P} \wedge \mathcal{Q} \models_{\mathcal{S},s} \mathcal{P}$$

et comme c'est vrai quelles que soient $\mathcal{S}$ et s alors :

$$\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$$

On peut faire de même pour le *modus ponens* :

$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{P} \to \mathcal{Q}$	$\mathcal{Q}$
V	V	V	V
V	F	F	—
F	V	V	—
F	F	V	—

On constate que chaque fois que $\mathcal{P}$ est vraie et en même temps que $\mathcal{P} \to \mathcal{Q}$ est vraie, alors $\mathcal{Q}$ est aussi vraie.

2.2. Règles des quantificateurs

Les règles avec les quantificateurs sont aussi compatibles avec la satisfaction. C'est encore dû au fait que les règles d'inférence ont été choisies avec bon sens. Par contre cette validité est un jeu d'écriture pas toujours simple à détailler. En première lecture, vous pouvez passer cette étape et vous rendre directement à la fin de la preuve. Cependant, étudier les preuves aide à bien comprendre les conditions d'application des règles d'inférence.
Rappelons les définitions et notions du chapitre « Satisfaction ».

$$\vDash_{\mathcal{S},s} \forall x\ \mathcal{P}(x) \qquad \text{ssi} \qquad \text{pour tout } d \in E,\ \vDash_{\mathcal{S},s[x\mapsto d]}\ \mathcal{P}(x)$$

La notation $s[x \mapsto d]$ est l'affectation qui envoie x sur d et coïncide avec s pour toutes les autres variables.

$$\vDash_{\mathcal{S},s} \exists x\ \mathcal{P}(x) \qquad \text{ssi} \qquad \text{il existe un } d \in E,\ \vDash_{\mathcal{S},s[x\mapsto d]}\ \mathcal{P}(x)$$

Rappelons le lien entre affectation et substitution :

$$\vDash_{\mathcal{S},s} \mathcal{P}[c/x] \qquad \text{ssi} \qquad \vDash_{\mathcal{S},s[x\mapsto c^{\mathcal{S}}]} \mathcal{P}(x)$$

Enfin, pour une formule $\mathcal{P}$, on la note $\mathcal{P}(x)$ pour insister sur sa dépendance vis-à-vis de la variable x, et on note :

$$\mathcal{P}(c) = \mathcal{P}[c/x]$$

Lorsque l'on note $\mathcal{P}(c)$, cela sous-entend que la substitution de x par c est valide.
Nous commençons par les deux règles simples.

Règle I. On a la règle

$$\forall x\ \ \mathcal{P}(x) \quad \vdash \quad \mathcal{P}(c)$$

pour c une constante. Et on a bien la satisfaction :

$$\forall x\ \ \mathcal{P}(x) \quad \vDash \quad \mathcal{P}(c)$$

Démonstration. Considérons une structure $\mathcal{S}$ de domaine E et une affectation s telles que « $\forall x\ \ \mathcal{P}(x)$ » est une formule vraie. Par définition, cela signifie que, quel que soit d dans le domaine E, $\mathcal{P}(d)$ est vraie. Donc en particulier $\mathcal{P}(c)$ est satisfaite.
Reprenons formellement la preuve. Soit c une constante du langage et $c^{\mathcal{S}}$ son interprétation dans E.

$$\begin{aligned}
& \vDash_{\mathcal{S},s} \quad \forall x\ \ \mathcal{P}(x) \\
\text{donc} \quad & \vDash_{\mathcal{S},s[x\mapsto d]}\ \mathcal{P}(x) \quad \text{pour tout } d \in E \\
\text{en particulier} \quad & \vDash_{\mathcal{S},s[x\mapsto c^{\mathcal{S}}]}\ \mathcal{P}(x) \\
\text{donc} \quad & \vDash_{\mathcal{S},s} \mathcal{P}[c/x] \\
\text{d'où} \quad & \vDash_{\mathcal{S},s} \mathcal{P}(c)
\end{aligned}$$

□

Règle III. On a la règle

$$\mathcal{P}(c) \quad \vdash \quad \exists x \;\; \mathcal{P}(x)$$

pour c une constante, et on a bien la satisfaction :

$$\mathcal{P}(c) \quad \vDash \quad \exists x \;\; \mathcal{P}(x)$$

Démonstration. Fixons $\mathcal{S}$ de domaine E et s. Si $\mathcal{P}(c)$ est vraie, alors il existe bien un $d \in E$ (en l'occurrence $d = c^{\mathcal{S}}$) tel que $\mathcal{P}(d)$ soit vraie et donc « $\exists x \;\; \mathcal{P}(x)$ » est une formule vraie.
Reprenons formellement la preuve. Soit c une constante du langage et $c^{\mathcal{S}}$ son interprétation dans E.

$$\begin{array}{rl} & \vDash_{\mathcal{S},s} \mathcal{P}(c) \\ \text{donc} & \vDash_{\mathcal{S},s} \mathcal{P}[c/x] \\ \text{donc} & \vDash_{\mathcal{S},s[x\mapsto c^{\mathcal{S}}]} \mathcal{P}(x) \\ \text{d'où} & \vDash_{\mathcal{S},s} \;\; \exists x \;\; \mathcal{P}(x) \quad \text{puisque la formule est vraie pour } d = c^{\mathcal{S}} \end{array}$$

□

Règle II. On reformule cette règle de la façon suivante. Soit Γ un ensemble de formules et c une constante qui n'apparaît pas dans les formules de Γ. La règle II s'écrit :

$$\text{Si} \;\; \Gamma \;\; \vdash \;\; \mathcal{P}(c) \quad \text{alors} \quad \Gamma \;\; \vdash \;\; \forall x \;\; \mathcal{P}(x)$$

Et on a bien un énoncé correspondant en termes de satisfaction :

$$\text{Si} \;\; \Gamma \;\; \vDash \;\; \mathcal{P}(c) \quad \text{alors} \quad \Gamma \;\; \vDash \;\; \forall x \;\; \mathcal{P}(x)$$

Démonstration. Nous avons un langage $\mathcal{L}$ avec une constante c. Supposons $\Gamma \;\; \vDash \;\; \mathcal{P}(c)$. Fixons $\mathcal{S}$ une structure de domaine E et s une affectation telles que Γ soit satisfait. Intuitivement, il s'agit de montrer que, quel que soit $d \in E$, $\mathcal{P}(d)$ est satisfaite. Formellement, il faut prouver que, quel que soit $d \in E$, on a $\vDash_{\mathcal{S},s[x\mapsto d]} \;\; \mathcal{P}(x)$.
Pour l'instant nous avons une structure $\mathcal{S}$ de domaine E ; dans cette structure, la constante c du langage s'interprète en une valeur $c^{\mathcal{S}} \in E$. Les hypothèses impliquent alors que $\mathcal{P}(c^{\mathcal{S}})$ est satisfaite. On a donc vérifié $\mathcal{P}(x)$ pour une valeur du domaine, mais on veut le prouver pour toutes les valeurs.
Fixons $d \in E$ une valeur quelconque du domaine. On va considérer une autre structure $\mathcal{S}'$ de même domaine E. $\mathcal{S}'$ est égale à $\mathcal{S}$, la seule différence avec $\mathcal{S}$ est que dans cette structure l'interprétation de c est notre valeur fixée d : $c^{\mathcal{S}'} = d$. On ne change pas l'affectation s. Comme les formules de Γ ne dépendent pas de la constante c et que Γ est satisfait pour $\mathcal{S}$ et s alors Γ est aussi satisfait pour $\mathcal{S}'$ et s.
L'hypothèse $\Gamma \;\; \vDash \;\; \mathcal{P}(c)$ exprime la satisfaction pour n'importe quelle structure, donc en particulier, on a

$$\vDash_{\mathcal{S}',s} \;\; \mathcal{P}(c)$$

Mais comme Γ est satisfait pour $\mathcal{S}'$ et s, alors on a :

$$\begin{array}{rl} & \vDash_{\mathcal{S}',s} \;\; \mathcal{P}(c) \\ \text{donc} & \vDash_{\mathcal{S}',s} \;\; \mathcal{P}[c/x] \\ \text{donc} & \vDash_{\mathcal{S}',s[x\mapsto c^{\mathcal{S}'}]} \;\; \mathcal{P}(x) \\ \text{donc} & \vDash_{\mathcal{S}',s[x\mapsto d]} \;\; \mathcal{P}(x) \\ \text{d'où} & \vDash_{\mathcal{S},s[x\mapsto d]} \;\; \mathcal{P}(x) \end{array}$$

Noter que dans la dernière ligne on est revenu à la structure $\mathcal{S}$, ce qui est possible car dans la formule $\mathcal{P}(x)$ la constante c n'apparaît pas (en effet dès qu'on écrit $\mathcal{P}(c)$, il faut que la substitution de x par c soit possible, donc que c n'apparaisse pas dans $\mathcal{P}(x)$).
Ainsi, pour n'importe quel $d \in E$, nous avons obtenu :

$$\vDash_{\mathcal{S},s[x\mapsto d]} \;\; \mathcal{P}(x)$$

Ce qui est exactement la définition de :

$$\vDash_{\mathcal{S},s} \quad \forall x \;\; \mathcal{P}(x)$$

□

Règle IV. On reformule cette règle de la façon suivante. $\mathcal{Q}$ désigne une formule, Γ un ensemble de formules, c une constante du langage qui n'apparaît ni dans $\mathcal{Q}$, ni dans les formules de Γ. La règle IV s'écrit :

$$\text{Si} \quad \Gamma \vdash \exists x \;\; \mathcal{P}(x) \quad \text{et} \quad \mathcal{P}(c) \vdash \mathcal{Q} \quad \text{alors} \;\; \Gamma \vdash \mathcal{Q}$$

Et on a bien un énoncé correspondant en termes de satisfaction :

$$\text{Si} \quad \Gamma \vDash \exists x \;\; \mathcal{P}(x) \quad \text{et} \quad \mathcal{P}(c) \vDash \mathcal{Q} \quad \text{alors} \;\; \Gamma \vDash \mathcal{Q}$$

Démonstration. Considérons d'abord une structure $\mathcal{S}$ de domaine E et une affectation s telles que Γ soit satisfait. Alors par hypothèse, « $\exists x \;\; \mathcal{P}(x)$ » est satisfaite, donc il existe $d \in E$ tel que $\mathcal{P}(x)$ soit satisfaite pour l'affectation $s[x \mapsto d]$.
Changeons la structure $\mathcal{S}$ en une structure $\mathcal{S}'$, où seule l'interprétation de la constante c est changée : dans $\mathcal{S}'$ on définit $c^{\mathcal{S}'} = d$. Comme $\mathcal{Q}$ et les formules de Γ ne dépendent pas de c, alors leurs satisfactions dans $\mathcal{S}$ et dans $\mathcal{S}'$ sont équivalentes. Ainsi Γ est satisfait dans $\mathcal{S}'$, mais en plus $\mathcal{P}(c)$ le sera aussi :

$$\begin{array}{ll}
 & \vDash_{\mathcal{S}',s} \quad \exists x \;\; \mathcal{P}(x) \\
\text{donc} & \vDash_{\mathcal{S}',s[x\mapsto d]} \quad \mathcal{P}(x) \quad \text{pour le même } d \in E \text{ que ci-dessus} \\
\text{donc} & \vDash_{\mathcal{S}',s[x\mapsto c^{\mathcal{S}'}]} \quad \mathcal{P}(x) \quad \text{car } c^{\mathcal{S}'} = d \\
\text{donc} & \vDash_{\mathcal{S}',s} \quad \mathcal{P}[c/x] \\
\text{donc} & \vDash_{\mathcal{S}',s} \quad \mathcal{P}(c) \\
\text{donc} & \vDash_{\mathcal{S}',s} \quad \mathcal{Q} \quad \text{d'après l'hypothèse } \mathcal{P}(c) \vDash \mathcal{Q} \\
\text{d'où} & \vDash_{\mathcal{S},s} \quad \mathcal{Q} \quad \text{car } c \text{ n'apparaît pas dans } \mathcal{Q}
\end{array}$$

□

2.3. Preuve du théorème

On note $\Gamma = \{\mathcal{P}_1, \ldots, \mathcal{P}_l\}$ l'ensemble des formules jouant le rôle des hypothèses et $\mathcal{Q}$ la formule de conclusion. On suppose $\Gamma \vdash \mathcal{Q}$. Il s'agit de démontrer que $\Gamma \vDash \mathcal{Q}$.
Le fait que $\Gamma \vdash \mathcal{Q}$ signifie qu'il existe une preuve de $\mathcal{Q}$ à partir des formules de Γ. Supposons que cette preuve soit écrite en déduction naturelle comme la figure ci-dessous. $\mathcal{P}_1, \ldots, \mathcal{P}_l$ sont les hypothèses. Chaque formule $\mathcal{P}_k$, avec $l+1 \leqslant k \leqslant q$, est une formule obtenue en appliquant une des règles d'inférence aux lignes précédentes $\mathcal{P}_1, \ldots, \mathcal{P}_{k-1}$. La dernière formule $\mathcal{P}_q$ obtenue est la formule $\mathcal{Q}$.

	$\mathcal{P}_1$	
	$\vdots$	Hypothèses
	$\mathcal{P}_l$	
	$\mathcal{P}_{l+1}$	
	$\vdots$	
	$\mathcal{P}_k$	Étapes de la preuve
	$\vdots$	
	$\mathcal{P}_{q-1}$	
$\vdash$	$\mathcal{P}_q = \mathcal{Q}$	Conclusion

Fixons une structure $\mathcal{S}$ et une affectation s telles que Γ soit satisfait. Il s'agit de démontrer qu'alors $\mathcal{Q}$ est aussi satisfaite (pour $\mathcal{S}$ et s). Nous allons le prouver par récurrence, sur k (pour $1 \leqslant k \leqslant q$) :

$$\mathcal{H}_k : \quad \text{la formule } \mathcal{P}_k \text{ est satisfaite}$$

- *Initialisation.* Pour $k = 1, \ldots, l$, la formule $\mathcal{P}_k$ est une des formules de Γ, donc est satisfaite par hypothèse.
- *Hérédité.* Soit k tel que $1 \leqslant k < q$. Supposons $\mathcal{P}_1, \ldots, \mathcal{P}_k$ satisfaites (pour $\mathcal{S}$ et s) et justifions que $\mathcal{P}_{k+1}$ est encore satisfaite. En effet, $\mathcal{P}_{k+1}$ est obtenue à partir des formules précédentes en suivant l'une des règles d'inférence des preuves. Nous avons déjà démontré que chacune de ces règles respecte la satisfaction des formules, donc $\mathcal{P}_{k+1}$ est satisfaite (pour $\mathcal{S}$ et s).
- *Conclusion.* Par le principe de récurrence, la dernière formule $\mathcal{P}_q$, qui est en fait $\mathcal{Q}$, est satisfaite.

Ainsi $\vDash_{\mathcal{S},s} \mathcal{Q}$ et comme $\mathcal{S}$ et s sont quelconques, alors $\Gamma \vDash \mathcal{Q}$.

Exercices

Le théorème de validité

Exercice 1. Soit Γ un ensemble de formules tel que $\Gamma \vdash \bot$. Montrer que Γ est une contradiction, c'est-à-dire qu'il n'existe aucune structure $\mathcal{S}$ et aucune affectation s, telles que toutes les formules de Γ soient satisfaites.

Exercice 2. Démontrer l'improuvabilité des énoncés suivants :

1. $\exists x\ P(x) \quad \nvdash \quad \forall x\ P(x)$
2. $\forall x\ \exists y\ Q(x,y) \quad \nvdash \quad \exists y\ \forall x\ Q(x,y)$
3. $\exists x\ \big(P(x) \to Q(x)\big) \quad \nvdash \quad \exists x\ \big(Q(x) \to P(x)\big)$
4. $\forall x\ \big(P(x) \vee Q(x)\big) \quad \nvdash \quad \big(\forall x\ P(x)\big) \vee \big(\forall x\ Q(x)\big)$

Exercice 3. Montrer les satisfactions suivantes :

1. $\forall x\ \big(P(x) \to Q(x)\big),\ P(c) \quad \vDash \quad Q(c)$
2. $\forall x\ \big(P(x) \to \bot\big) \quad \vDash \quad \neg\big(\exists x\ P(x)\big)$
3. $\forall x\ \big(P(x) \leftrightarrow Q(x)\big),\ \ \forall x\ \big(Q(x) \leftrightarrow R(x)\big) \quad \vDash \quad \forall x\ \big(P(x) \leftrightarrow R(x)\big)$
4. $\forall x\ \big(P(x) \wedge Q(x)\big) \quad \vDash \quad \forall x\ P(x)$

Preuves du théorème de validité

Exercice 4. Vérifier que toutes les règles de base de la logique V/F sont compatibles avec la satisfaction.

Chapitre 10

Théorème de complétude

Le théorème de complétude dit que, en logique du premier ordre, tout ce qui est vrai est prouvable. Nous énonçons ce théorème, ses variantes et ses applications.

1. Théorème de complétude

1.1. Énoncé

Soit $\mathcal{L}$ un langage du premier ordre.

Théorème 1 (Théorème de complétude)**.**

$$\boxed{\textit{Si} \quad \mathcal{P}_1, \ldots, \mathcal{P}_l \quad \models \quad \mathcal{Q} \quad \textit{alors} \quad \mathcal{P}_1, \ldots, \mathcal{P}_l \quad \vdash \quad \mathcal{Q}}$$

Ce que l'on peut lire : si, lorsque les hypothèses $\mathcal{P}_1, \ldots, \mathcal{P}_l$ sont satisfaites, $\mathcal{Q}$ l'est aussi, alors il existe une preuve de $\mathcal{Q}$ à partir de $\mathcal{P}_1, \ldots, \mathcal{P}_l$.
Voici la variante pour un ensemble Γ de formules, éventuellement infini dénombrable :

Théorème 2 (Théorème de complétude)**.**

$$\textit{Si} \quad \Gamma \quad \models \quad \mathcal{Q} \quad \textit{alors} \quad \Gamma \quad \vdash \quad \mathcal{Q}.$$

La conclusion $\Gamma \vdash \mathcal{Q}$ est un résultat d'existence : « il existe une preuve de $\mathcal{Q}$ à partir des formules de Γ ». Cependant, le théorème n'est pas constructif et ne donne aucune indication concrète sur cette preuve.
La démonstration du théorème de complétude est assez exigeante. Elle partage des points communs avec le théorème de compacité de la logique V/F mais avec un niveau de difficulté supplémentaire. Cette démonstration sera faite au prochain chapitre. On va maintenant se concentrer sur les différentes reformulations et les applications.

1.2. Équivalence

On a vu au chapitre précédent le théorème de validité qui est exactement la réciproque du théorème de complétude. Ainsi, on obtient une équivalence qu'on appelle encore théorème de complétude :

Théorème 3 (Théorème de complétude)**.**

$$\boxed{\Gamma \quad \models \quad \mathcal{Q} \quad \textit{ssi} \quad \Gamma \quad \vdash \quad \mathcal{Q}}$$

Ainsi, pour la logique du premier ordre, la conséquence logique et la prouvabilité sont des notions équivalentes.

1.3. Et l'incomplétude ?

N'y a-t-il pas une contradiction à obtenir le théorème de complétude alors qu'à la fin du livre on obtiendra le théorème d'incomplétude, dont le nom semble suggérer le contraire ? En fait, le théorème de complétude s'applique à la logique du premier ordre qui est une logique assez rudimentaire mais avec l'hypothèse $\Gamma \models \mathcal{Q}$ qui est assez forte car la conséquence logique exige que la formule soit vraie dans toutes les structures possibles.
Le théorème d'incomplétude ne concerne pas l'ensemble des structures, mais concerne des théories plus spécifiques, typiquement qui contiennent les entiers naturels $\mathbb{N}$ avec les axiomes de l'arithmétique (les axiomes de Peano). On verra que, pour l'arithmétique des entiers, il existe des formules $\mathcal{Q}$ telles que ni $\mathcal{Q}$ ni $\neg\mathcal{Q}$ ne sont prouvables, autrement dit, il existe des formules vraies mais non prouvables, mais tout ceci uniquement pour la structure associée à l'arithmétique des entiers. Ceci n'est pas en contradiction avec le théorème de complétude car l'hypothèse $\Gamma \models_{\mathcal{S}} \mathcal{Q}$ n'est pas vérifiée pour toutes les structures, mais seulement pour certaines structures spécifiques ; les conditions du théorème de complétude ne sont donc pas réunies.
Ainsi, les théorèmes de complétude et d'incomplétude sont tous les deux vrais, mais ne s'appliquent pas aux mêmes situations et ne se contredisent donc pas.

2. Applications aux théories

On va rappeler des définitions et introduire de nouvelles notions. Soit $\mathcal{L}$ un langage du premier ordre.

2.1. Théorie cohérente, modèle

- Une ***formule fermée*** (ou ***formule close***, ou ***phrase***) est une formule sans variable libre, c'est-à-dire que toutes les occurrences de variables sont dans la portée d'un quantificateur.
- Une ***théorie*** $\mathcal{T}$ est un ensemble de formules fermées. (Remarque : certains appellent « théorie » un ensemble quelconque de formules.)
- Une structure $\mathcal{S}$ est un ***modèle*** pour $\mathcal{T}$ si $\mathcal{T}$ est satisfaite dans $\mathcal{S}$, noté $\models_{\mathcal{S}} \mathcal{T}$, c'est-à-dire que l'interprétation de chaque formule de $\mathcal{T}$ est vraie dans $\mathcal{S}$. (Comme les formules sont fermées, il n'y a pas d'affectation à considérer.)

Voici une nouvelle notion très importante.

Définition.
Une théorie $\mathcal{T}$ est ***non cohérente*** s'il existe une formule $\mathcal{Q}$ telle que :

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{et} \quad \mathcal{T} \vdash \neg\mathcal{Q}.$$

Dans le cas contraire, on dit que $\mathcal{T}$ est ***cohérente***.

Il est équivalent de dire que $\mathcal{T}$ est non cohérente si $\mathcal{T} \vdash \bot$, où $\bot$ désigne une contradiction.
Ainsi, d'une théorie cohérente on ne pourra jamais démontrer une contradiction (ce qui est rassurant).

2.2. Lemme de déduction naturelle

Γ désigne un ensemble de formules, et $\mathcal{P}$, $\mathcal{Q}$ des formules.

Lemme 1 (Lemme de déduction naturelle)**.**

$$\boxed{\Gamma, \mathcal{P} \vdash \mathcal{Q} \quad \textit{ssi} \quad \Gamma \vdash (\mathcal{P} \to \mathcal{Q})}$$

Démonstration. Le sens réciproque $\Leftarrow$ est en fait la règle du *modus ponens* $\mathcal{P}, \mathcal{P} \to \mathcal{Q} \vdash \mathcal{Q}$.
Le sens direct $\Rightarrow$ est immédiat quand on a une preuve en déduction naturelle (d'où le nom du lemme), il s'agit juste d'une reformulation dans l'écriture de la preuve :

Preuve de $\Gamma, \mathcal{P} \vdash \mathcal{Q}$

	Γ	
	$\mathcal{P}$	Hypothèses
	$\vdots$	Étapes
$\vdash$	$\mathcal{Q}$	Conclusion

Preuve de $\Gamma \vdash (\mathcal{P} \to \mathcal{Q})$

	Γ	
	$\mathcal{P}$	Sous-hypothèse
	$\vdots$	Mêmes étapes
	$\mathcal{Q}$	Sous-conclusion
$\vdash$	$\mathcal{P} \to \mathcal{Q}$	Conclusion

□

Soit $\mathcal{T}$ une théorie et $\mathcal{Q}$ une formule fermée.

Corollaire 1.

$$\boxed{\mathcal{T} \vdash \mathcal{Q} \quad \textit{ssi} \quad \mathcal{T} \cup \{\neg\mathcal{Q}\} \ \textit{non cohérente}}$$

Démonstration.

$\mathcal{T} \cup \{\neg\mathcal{Q}\}$ non cohérente
ssi $\mathcal{T} \cup \{\neg\mathcal{Q}\} \vdash \bot$ par définition
ssi $\mathcal{T} \vdash (\neg\mathcal{Q} \to \bot)$ par le lemme de déduction naturelle
ssi $\mathcal{T} \vdash \neg(\neg\mathcal{Q})$ car par définition $\neg\mathcal{P}$ signifie $\mathcal{P} \to \bot$
ssi $\mathcal{T} \vdash \mathcal{Q}$ par double négation

□

2.3. Différentes formulations du théorème de complétude

Nous allons énoncer le théorème de complétude ainsi que quelques reformulations.
Cadre : $\mathcal{T}$ est une théorie, $\mathcal{Q}$ une formule fermée.
Formulation classique :

Théorème 4 (Théorème A)**.**

$$\boxed{\mathcal{T} \vDash \mathcal{Q} \quad \textit{ssi} \quad \mathcal{T} \vdash \mathcal{Q}}$$

Reformulation :

Théorème 5 (Théorème B)**.**

$$\boxed{\mathcal{T} \ \textit{cohérente} \quad \textit{ssi} \quad \mathcal{T} \ \textit{admet un modèle}}$$

On pourrait énoncer un résultat similaire pour un ensemble quelconque de formules :

Γ cohérent ssi Γ satisfaisable.

2.4. Démonstration du théorème B à partir du théorème A

$\mathcal{T}$ non cohérente
ssi $\mathcal{T} \vdash \bot$ par définition
ssi $\mathcal{T} \models \bot$ par le théorème A
ssi $\mathcal{T}$ n'est satisfaite par aucune structure $\mathcal{S}$
ssi $\mathcal{T}$ n'a pas de modèle

On verra en exercice qu'on pourrait prouver le théorème A à partir du théorème B.

3. Application à la finitude

Nous nous intéressons au cas où l'ensemble des hypothèses est infini (mais dénombrable). Nous allons voir que seul un nombre fini d'hypothèses est nécessaire. Nous aurons deux résultats, un sur les preuves (le théorème de finitude) et un sur la satisfaction (le théorème de compacité en logique du premier ordre, analogue à celui du chapitre « Théorème de compacité » de la logique V/F).

3.1. Théorème de finitude

Soit Γ un ensemble de formules (éventuellement infini), soit $\mathcal{Q}$ une formule d'un langage $\mathcal{L}$.

Théorème 6 (Théorème de finitude)**.**
Si $\Gamma \vdash \mathcal{Q}$ *alors il existe une partie finie* $\Gamma_0 \subset \Gamma$ *telle que* $\Gamma_0 \vdash \mathcal{Q}$.

Remarque : la réciproque est bien sûr vraie.

Démonstration. La démonstration est fondée sur la nature même de ce qu'est une preuve. $\Gamma \vdash \mathcal{Q}$ signifie qu'il existe une preuve de $\mathcal{Q}$ à partir des formules de Γ. Considérons une telle preuve. Cette preuve consiste en un nombre fini d'étapes et à chaque étape on applique une règle. Chaque règle ne fait intervenir qu'un nombre fini de formules de Γ. On note Γ_0 l'ensemble des formules de Γ qui apparaissent dans cette preuve. Alors on a bien obtenu une preuve de $\mathcal{Q}$ à partir de Γ_0, donc $\Gamma_0 \vdash \mathcal{Q}$ avec Γ_0 fini. □

3.2. Théorème de compacité

La version correspondante pour la satisfaction est le théorème de compacité pour la logique du premier ordre, similaire à celui rencontré pour la logique V/F dans le chapitre « Théorème de compacité » de la première partie.

Théorème 7 (Théorème de compacité)**.**
Γ *satisfaisable ssi toute partie finie* $\Gamma_0 \subset \Gamma$ *est satisfaisable.*

Reformulation : $\Gamma \models \mathcal{Q}$ ssi pour toute partie finie $\Gamma_0 \subset \Gamma$ on a $\Gamma_0 \models \mathcal{Q}$.
Remarque : à chaque fois le sens direct $\Rightarrow$ est évident.
Pour les théories, une reformulation du sens intéressant est :

Théorème 8.
Si toute partie finie d'une théorie $\mathcal{T}$ *admet un modèle, alors* $\mathcal{T}$ *elle-même admet un modèle.*

Il est intéressant de prendre le temps de réfléchir à la profondeur d'un tel théorème. Chaque partie finie a un modèle, mais bien sûr ce modèle peut être différent pour chaque Γ_0. Cependant, on trouve à la fin un modèle commun à toutes les formules.

Démonstration du théorème de compacité. On a déjà dit que le sens direct $\Rightarrow$ est évident. On va démontrer le sens réciproque $\Leftarrow$ par contraposition : « si Γ non satisfaisable, alors il existe Γ_0 fini tel que Γ_0 non satisfaisable. »

Γ non satisfaisable
donc $\Gamma \models \bot$ car de toute façon Γ n'est jamais satisfait
donc $\Gamma \vdash \bot$ par le théorème de complétude, donc Γ non cohérente
donc $\Gamma_0 \vdash \bot$ pour un certain $\Gamma_0 \subset \Gamma$ fini, par le théorème de finitude
donc $\Gamma_0 \models \bot$ par le théorème de validité
donc Γ_0 non satisfaisable
donc il existe Γ_0 fini, non satisfaisable
donc non (tout Γ_0 fini est satisfaisable)

□

4. Théorème de clôture

Le théorème de clôture explique que l'on peut passer d'un énoncé du théorème de complétude avec des formules fermées à un énoncé pour des formules quelconques.

4.1. Énoncé

Théorème 9 (Théorème A)**.**
Soit $\mathcal{T}$ une théorie, $\mathcal{Q}$ une formule fermée.

$$\textit{Si } \mathcal{T} \models \mathcal{Q} \quad \textit{alors} \quad \mathcal{T} \vdash \mathcal{Q}.$$

Théorème 10 (Théorème C)**.**
Soit maintenant Γ un ensemble de formules quelconques, et $\mathcal{Q}$ une formule quelconque.

$$\textit{Si } \Gamma \models \mathcal{Q} \quad \textit{alors} \quad \Gamma \vdash \mathcal{Q}.$$

Théorème 11 (Théorème de clôture)**.**
Le théorème A entraîne le théorème C.

Autrement dit, pour démontrer le théorème de complétude en toute généralité, il suffit de démontrer le théorème de complétude pour les formules fermées.

4.2. Clôture

Avant de donner la démonstration du théorème C, introduisons une notion de clôture par introduction de nouvelles constantes.

Définition.
Soit $\mathcal{P}$ une formule ayant des variables libres, notées $x_1, \ldots, x_n$, définie sur un langage $\mathcal{L}$. On note $\mathcal{L}^c$ le langage obtenu à partir de $\mathcal{L}$ en ajoutant n nouvelles constantes $c_1, \ldots, c_n$ (ces constantes ne doivent pas apparaître dans $\mathcal{L}$). On note $\mathcal{P}^c$ ***substitution par les constantes nouvelles*** de $\mathcal{P}$, la formule obtenue à partir de $\mathcal{P}$ en remplaçant chaque variable libre x_i par la nouvelle constante c_i.

Par construction, $\mathcal{P}^c$ est une formule fermée (du langage $\mathcal{L}^c$).
Pour un ensemble de formules Γ et une formule $\mathcal{Q}$, on note $x_1, \ldots, x_n$ l'ensemble des variables libres dans $\Gamma \cup \{\mathcal{Q}\}$. On construit ainsi un langage $\mathcal{L}^c$, et des formules fermées Γ^c (la clôture par les constantes nouvelles des formules de Γ) et $\mathcal{Q}^c$.

Voici quelques propriétés utiles pour la preuve, qui sont essentiellement des jeux d'écriture.

- Point a.

$$\Gamma \models \mathcal{Q} \qquad \text{implique} \qquad \Gamma^c \models \mathcal{Q}^c$$

Justifions en détail. Considérons une structure $\mathcal{S}^c$ (pour le langage $\mathcal{L}^c$) qui satisfait Γ^c. On note $d_i = c_i^{\mathcal{S}^c}$ l'interprétation de la constante c_i dans $\mathcal{S}^c$.
On peut construire une structure $\mathcal{S}$ (pour le langage $\mathcal{L}$) en oubliant les constantes nouvelles $c_1, \ldots, c_n$ et en même temps on construit une affectation s en posant $s(x_i) = d_i$ pour chaque variable libre x_i.
Pour une formule $\mathcal{P}$ (de Γ ou $\mathcal{Q}$), on a alors :

$$\models_{\mathcal{S}^c} \mathcal{P}^c \qquad \text{ssi} \qquad \models_{\mathcal{S},s} \mathcal{P}$$

(Ceci découle directement de la substitution de chaque variable libre x_i par la constante c_i.)
Comme $\models_{\mathcal{S}^c} \Gamma^c$, alors on a $\models_{\mathcal{S},s} \Gamma$ (par la substitution précédente). Par hypothèse, on a $\Gamma \models \mathcal{Q}$, donc $\models_{\mathcal{S},s} \mathcal{Q}$, et donc en retour $\models_{\mathcal{S}^c} \mathcal{Q}^c$.

- Point b. Le lemme des constantes nouvelles :

$$\Gamma^c \vdash \mathcal{Q}^c \qquad \text{implique} \qquad \Gamma \vdash \mathcal{Q}$$

En effet imaginons qu'on a une preuve écrite en déduction naturelle de $\mathcal{Q}^c$ à partir de Γ^c. On recopie cette preuve, en remplaçant dans toutes les lignes les occurrences des constantes nouvelles c_i par les variables x_i. On obtient une preuve de $\mathcal{Q}$ à partir de Γ.

Preuve de $\Gamma^c \vdash \mathcal{Q}^c$

	$\mathcal{P}_1^c$	Hypothèses Γ^c
	$\vdots$	
	$\mathcal{P}_l^c$	
	$\vdots$	Preuve
	$\ldots c_i \ldots$	Les constantes nouvelles
	$\vdots$	
$\vdash$	$\mathcal{Q}^c$	Conclusion

Preuve de $\Gamma \vdash \mathcal{Q}$

	$\mathcal{P}_1$	Hypothèses Γ
	$\vdots$	
	$\mathcal{P}_l$	
	$\vdots$	Preuve
	$\ldots x_i \ldots$	Retour aux variables
	$\vdots$	
$\vdash$	$\mathcal{Q}$	Conclusion

4.3. Démonstration du théorème C à partir du théorème A

	$\Gamma \models \mathcal{Q}$	
donc	$\Gamma^c \models \mathcal{Q}^c$	par le point a
donc	$\Gamma^c \vdash \mathcal{Q}^c$	par le théorème A pour les formules fermées
donc	$\Gamma \vdash \mathcal{Q}$	par le point b

4.4. Clôture universelle

Il existe une autre notion de clôture, appelée *clôture universelle*.

Définition.
Soit $\mathcal{P}$ une formule ayant des variables libres, notées $x_1, \ldots, x_n$. On note $\overline{\mathcal{P}}$ (ou aussi $\forall \mathcal{P}$) la ***clôture universelle*** de $\mathcal{P}$:

$$\overline{\mathcal{P}} : \qquad \forall x_1 \, \forall x_2 \, \ldots \, \forall x_n \quad \mathcal{P}$$

Exercices

Théorème de complétude

Exercice 1. On considère Γ un ensemble de formules et $\mathcal{P}$, $\mathcal{Q}$ des formules telles que :

$$\Gamma \vdash \mathcal{P} \qquad \text{et} \qquad \mathcal{P} \vdash \mathcal{Q} \tag{$\star$}$$

Le but de l'exercice est d'obtenir $\Gamma \vdash \mathcal{Q}$.

1. *Première méthode.* À partir d'une preuve en déduction naturelle de $\Gamma \vdash \mathcal{P}$ et d'une preuve en déduction naturelle de $\mathcal{P} \vdash \mathcal{Q}$, écrire une preuve de $\Gamma \vdash \mathcal{Q}$.
2. *Deuxième méthode.*
   (a) Quels énoncés obtient-on à partir de nos hypothèses ($\star$) en appliquant le théorème de validité ?
   (b) Sous les hypothèses ($\star$), montrer qu'on a $\Gamma \vDash \mathcal{Q}$.
   (c) Conclure.

Exercice 2. Soit $\mathcal{L}$ un langage muni d'un prédicat binaire $R(x,y)$. On considère les trois formule suivantes :

1. symétrie $\mathcal{S} : \forall x\ \forall y\ \big(R(x,y) \to R(y,x)\big)$
2. transitivité $\mathcal{T} : \forall x\ \forall y\ \forall z\ \big(R(x,y) \wedge R(y,z) \to R(x,z)\big)$
3. réflexivité $\mathcal{Q} : \forall x\ R(x,x)$

Une relation R pour laquelle ces trois formules sont vraies, s'appelle une *relation d'équivalence*. On se demande si les deux seuls axiomes $\mathcal{S}$ et $\mathcal{T}$ suffisent, c'est-à-dire a-t-on $\mathcal{S}, \mathcal{T} \vdash \mathcal{Q}$?

1. Un étudiant propose la preuve suivante :

   > *Soit x un élément quelconque. Prenons un élément y tel que $R(x,y)$. Par l'axiome $\mathcal{S}$ (symétrie), on a aussi $R(y,x)$. En appliquant l'axiome $\mathcal{T}$ (transitivité) sur $R(x,y)$ et $R(y,x)$, on en déduit $R(x,x)$. La relation est donc réflexive et $\mathcal{Q}$ est prouvée.*

   Quelle est l'erreur logique fondamentale dans ce raisonnement ?
2. Trouver une structure $\mathcal{S}$, c'est-à-dire définir un domaine E et y expliciter la relation R, telle que les formules $\mathcal{S}$ et $\mathcal{T}$ y soient vraies, mais telle que $\mathcal{Q}$ ne le soit pas.
3. A-t-on $\mathcal{S}, \mathcal{T} \vdash \mathcal{Q}$?

Applications aux théories

Exercice 3. Montrer l'équivalence entre ces deux définitions d'une théorie $\mathcal{T}$ *non cohérente* :
(i) il existe $\mathcal{Q}$, telle que $\mathcal{T} \vdash \mathcal{Q}$ et $\mathcal{T} \vdash \neg\mathcal{Q}$,
(ii) $\mathcal{T} \vdash \bot$.

Exercice 4. À l'aide du théorème de complétude « si $\mathcal{T} \vDash \mathcal{Q}$ alors $\mathcal{T} \vdash \mathcal{Q}$ » prouver que :
« si $\mathcal{T}$ est insatisfaisable alors $\mathcal{T}$ est non cohérente ».
Démontrer ensuite la réciproque.

Exercice 5. On rappelle les théorèmes A et B :

- Théorème A.
$$\mathcal{T} \vDash \mathcal{Q} \qquad \text{ssi} \qquad \mathcal{T} \vdash \mathcal{Q}.$$
- Théorème B.
$$\mathcal{T} \text{ cohérente} \qquad \text{ssi} \qquad \mathcal{T} \text{ admet un modèle.}$$

On admet le théorème B. En déduire le théorème A.

Application à la finitude

Exercice 6. Soit $\mathcal{L}$ un langage (contenant le symbole d'égalité $=$). Pour tout entier $n \geqslant 1$, on note $\mathcal{P}_n$ la formule exprimant « il existe au moins n éléments distincts ».

1. Écrire explicitement les formules $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$.
2. Soit $\mathcal{T}$ une théorie ayant des modèles finis de taille arbitrairement grande, c'est-à-dire que pour tout entier k, $\mathcal{T}$ admet un modèle ayant au moins k éléments.
   On pose $\Gamma = \mathcal{T} \cup \{\mathcal{P}_n \mid n \geqslant 1\}$. Montrer que tout sous-ensemble fini de Γ admet un modèle.
3. En utilisant le théorème de compacité, en déduire que $\mathcal{T}$ admet un modèle infini.
4. Conclusion : existe-t-il une théorie du premier ordre $\mathcal{T}_{\text{fini}}$ dont les modèles seraient **exactement** toutes les structures finies ?

Théorème de clôture

Exercice 7. Pour une formule $\mathcal{P}$ dont on note $x_1, \ldots, x_n$ les variables libres, sa *clôture existentielle* est la formule $\mathcal{P}_\exists$ (notée aussi $\exists\mathcal{P}$) :

$$\mathcal{P}_\exists : \qquad \exists x_1 \ \ldots \ \exists x_n \quad \mathcal{P}$$

1. Montrer que $\mathcal{P}$ est satisfiable si et seulement si $\mathcal{P}_\exists$ admet un modèle.
2. Montrer l'équivalence $\neg\big(\overline{\mathcal{P}}\big) \equiv (\neg\mathcal{P})_\exists$.
   On rappelle que $\overline{\mathcal{P}}$ désigne la clôture universelle et que $\mathcal{P} \equiv \mathcal{Q}$ signifie ($\mathcal{P} \vDash \mathcal{Q}$ et $\mathcal{Q} \vDash \mathcal{P}$), voir le chapitre « Satisfaction ».
3. Montrer que si $\Gamma \vdash \mathcal{Q}$, alors $\Gamma \vdash \mathcal{Q}_\exists$. La réciproque est-elle vraie ?

Chapitre 11
Preuve du théorème de complétude

Nous prouvons le théorème de complétude.

1. Le théorème de complétude

1.1. Énoncé

On rappelle l'énoncé qu'on a déjà bien étudié dans le chapitre « Théorème de complétude » mais que l'on n'a pas encore prouvé.

Théorème 1 (Théorème de complétude)**.**
Si $\Gamma \vDash \mathcal{Q}$ *alors* $\Gamma \vdash \mathcal{Q}$.

Autrement dit, tout ce qui est vrai est prouvable.

1.2. À prouver : « Si $\mathcal{T}$ est cohérente, alors $\mathcal{T}$ admet un modèle. »

Grâce au théorème de clôture du chapitre précédent, on ramène la preuve du théorème de complétude dans le cadre général au cadre des formules fermées. Soit $\mathcal{T}$ une théorie, $\mathcal{Q}$ une formule fermée.

Théorème 2 (Théorème A)**.**
Si $\mathcal{T} \vDash \mathcal{Q}$*, alors* $\mathcal{T} \vdash \mathcal{Q}$.

On va donc étudier le cas des théories. Ce théorème A sera obtenu à partir du théorème B suivant qui lui est équivalent :

Théorème 3 (Théorème B)**.**
Si $\mathcal{T}$ *est cohérente alors* $\mathcal{T}$ *admet un modèle.*

Ainsi on aura l'enchaînement :

$$\text{Théorème B} \implies \text{Théorème A} \implies \text{Théorème de complétude}$$

C'est donc ce théorème B que l'on va démontrer dans ce chapitre, après avoir justifié comment en déduire le théorème A.

1.3. Rappel du vocabulaire

- ***Théorie*** : un ensemble $\mathcal{T}$ de formules fermées.
- $\mathcal{T}$ est ***non cohérente*** si $\mathcal{T} \vdash \bot$; sinon $\mathcal{T}$ est ***cohérente***.
- $\mathcal{T}$ admet un ***modèle*** s'il existe une structure $\mathcal{S}$ telle que $\vDash_{\mathcal{S}} \mathcal{T}$, c'est-à-dire une structure dans laquelle toutes les formules de $\mathcal{T}$ deviennent vraies.

On a vu au chapitre précédent la proposition suivante :

Proposition 1.
$\mathcal{T} \vdash \mathcal{Q}$ ssi $\mathcal{T} \cup \{\neg\mathcal{Q}\}$ non cohérente.

Voici comment prouver le théorème A à partir du théorème B (on a vu la réciproque dans le chapitre précédent).

Démonstration. On va donc prouver le théorème A sous sa forme contraposée : si $\mathcal{T} \nvdash \mathcal{Q}$, alors $\mathcal{T} \nvDash \mathcal{Q}$.
On utilisera avec modération les négations d'opérateurs : $\mathcal{T} \nvdash \mathcal{Q}$ signifie non($\mathcal{T} \vdash \mathcal{Q}$) et $\mathcal{T} \nvDash \mathcal{Q}$ signifie non($\mathcal{T} \vDash \mathcal{Q}$).
Hypothèse : non($\mathcal{T} \vdash \mathcal{Q}$). But : non($\mathcal{T} \vDash \mathcal{Q}$).

	non($\mathcal{T} \vdash \mathcal{Q}$)	
donc	$\mathcal{T} \cup \{\neg\mathcal{Q}\}$ est cohérente	par la proposition 1 ci-dessus
donc	$\mathcal{T} \cup \{\neg\mathcal{Q}\}$ admet un modèle	par le théorème B
donc	$\vDash_{\mathcal{S}} \mathcal{T} \cup \{\neg\mathcal{Q}\}$	pour une certaine structure $\mathcal{S}$
donc	$\vDash_{\mathcal{S}} \mathcal{T}$ et $\vDash_{\mathcal{S}} \neg\mathcal{Q}$	car dans $\mathcal{S}$, $\mathcal{T}$ est vraie et $\neg\mathcal{Q}$ aussi
donc	non($\mathcal{T} \vDash \mathcal{Q}$)	car pour $\mathcal{S}$, $\mathcal{T}$ est vraie et $\mathcal{Q}$ fausse

□

2. Théorie complète

2.1. Définition

Définition.
Une théorie $\mathcal{T}$ est dite ***complète*** si :

- $\mathcal{T}$ est cohérente,
- et pour toute formule fermée $\mathcal{Q}$, on a : $\mathcal{T} \vdash \mathcal{Q}$ ou $\mathcal{T} \vdash \neg\mathcal{Q}$.

Ainsi dans une théorie complète, quelle que soit la formule fermée $\mathcal{Q}$, on peut la prouver ou bien prouver son contraire, mais pas les deux en même temps. La dénomination « complète » est donc bien choisie.

2.2. Satisfaction

Voici une conséquence importante quand on se place dans le cadre d'une théorie complète : les formules sont vraies ou fausses indépendamment de la structure.

Théorème 4.
Soient $\mathcal{T}$ une théorie complète, $\mathcal{Q}$ une formule fermée. Soient $\mathcal{S}_1$ et $\mathcal{S}_2$ deux modèles de $\mathcal{T}$. Alors :

$$\vDash_{\mathcal{S}_1} \mathcal{Q} \quad \textit{ssi} \quad \vDash_{\mathcal{S}_2} \mathcal{Q}.$$

Démonstration. Comme $\mathcal{T}$ est complète, on considère les deux cas : soit $\mathcal{T} \vdash \mathcal{Q}$, soit $\mathcal{T} \vdash \neg\mathcal{Q}$.

- Si $\mathcal{T} \vdash \mathcal{Q}$, alors $\mathcal{T} \vDash \mathcal{Q}$ par le théorème de validité, donc $\vDash_{\mathcal{S}_1} \mathcal{Q}$ et $\vDash_{\mathcal{S}_2} \mathcal{Q}$ car $\mathcal{S}_1$ et $\mathcal{S}_2$ sont des modèles de $\mathcal{T}$, c'est-à-dire que $\mathcal{Q}$ est vraie pour $\mathcal{S}_1$ et $\mathcal{S}_2$.
- Si $\mathcal{T} \vdash \neg\mathcal{Q}$, le même raisonnement donne $\vDash_{\mathcal{S}_1} \neg\mathcal{Q}$ et $\vDash_{\mathcal{S}_2} \neg\mathcal{Q}$, c'est-à-dire que $\mathcal{Q}$ est fausse dans $\mathcal{S}_1$ et $\mathcal{S}_2$.

□

2.3. Un lemme

Dans une théorie complète, il n'y a pas de piège. On va prouver les énoncés suivants qui pourraient sembler évidents mais nécessitent cette notion.

Lemme 1.
Soit $\mathcal{T}$ une théorie complète.

1. $\mathcal{T} \vdash \neg\mathcal{P}$ *ssi* *non*$(\mathcal{T} \vdash \mathcal{P})$
2. $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$ *ssi* $(\mathcal{T} \vdash \mathcal{P}$ *et* $\mathcal{T} \vdash \mathcal{Q})$
3. $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$ *ssi* $(\mathcal{T} \vdash \mathcal{P}$ *ou* $\mathcal{T} \vdash \mathcal{Q})$

Démonstration.

1. - Supposons $\mathcal{T} \vdash \neg\mathcal{P}$. Par l'absurde, si $\mathcal{T} \vdash \mathcal{P}$, alors $\mathcal{T} \vdash (\mathcal{P} \wedge \neg\mathcal{P})$, donc $\mathcal{T} \vdash \bot$. Contradiction avec $\mathcal{T}$ cohérente (car $\mathcal{T}$ complète). Donc non$(\mathcal{T} \vdash \mathcal{P})$.
   - Supposons non$(\mathcal{T} \vdash \mathcal{P})$. Comme $\mathcal{T}$ est complète ($\mathcal{T} \vdash \mathcal{P}$ ou $\mathcal{T} \vdash \neg\mathcal{P}$), on a donc $\mathcal{T} \vdash \neg\mathcal{P}$.
2. - Si $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$, alors $\mathcal{T} \vdash \mathcal{P}$ et $\mathcal{T} \vdash \mathcal{Q}$ car $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$ et $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}$.
   - Si $\mathcal{T} \vdash \mathcal{P}$ et $\mathcal{T} \vdash \mathcal{Q}$, alors $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$, car $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$.

   Remarque : ce point n'utilise pas l'hypothèse que $\mathcal{T}$ est complète.
3. - Si ($\mathcal{T} \vdash \mathcal{P}$ ou $\mathcal{T} \vdash \mathcal{Q}$), alors $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$ car $\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q}$ et $\mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}$. (Pas besoin de $\mathcal{T}$ complète).
   - Supposons $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$. Supposons par l'absurde que $\mathcal{T} \nvdash \mathcal{P}$ et $\mathcal{T} \nvdash \mathcal{Q}$. Comme $\mathcal{T} \nvdash \mathcal{P}$, on a $\mathcal{T} \vdash \neg\mathcal{P}$ car $\mathcal{T}$ est complète (par le point 1), et de même comme $\mathcal{T} \nvdash \mathcal{Q}$, alors $\mathcal{T} \vdash \neg\mathcal{Q}$. Ainsi par le point 2, $\mathcal{T} \vdash \neg\mathcal{P} \wedge \neg\mathcal{Q}$, donc $\mathcal{T} \vdash \neg(\mathcal{P} \vee \mathcal{Q})$. Mais alors, on a à la fois $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$ (par l'hypothèse) et $\mathcal{T} \vdash \neg(\mathcal{P} \vee \mathcal{Q})$, donc $\mathcal{T}$ est non cohérente. C'est une contradiction car $\mathcal{T}$ complète, donc cohérente.

□

3. Construction d'un modèle à partir d'une théorie complète

On commence la preuve du théorème de complétude dans sa forme du théorème B : « si $\mathcal{T}$ est complète, alors $\mathcal{T}$ admet un modèle ». On débute cette démonstration un peu par la fin. On va partir d'une théorie complète, avec des hypothèses supplémentaires, et on va construire un modèle. On expliquera plus tard comment obtenir ces hypothèses supplémentaires. Il existe plusieurs preuves du théorème de complétude, nous adoptons ici celle basée sur les témoins de Henkin.

3.1. Hypothèses renforcées

Soit $\mathcal{L}$ un langage et $\mathcal{T}$ une théorie. On va supposer les propriétés suivantes pour $\mathcal{T}$:
(i) $\mathcal{T}$ est cohérente.
(ii) $\mathcal{T}$ est complète.
(iii) $\mathcal{T}$ admet des témoins de Henkin.
Remarque : (ii) $\Longrightarrow$ (i) mais on insiste sur le fait qu'à la fin on aimerait avoir la seule hypothèse (i).

Définition.
$\mathcal{T}$ admet des ***témoins de Henkin*** si pour toute formule $\mathcal{P}$ de $\mathcal{L}$ ayant une unique variable libre x, il existe une constante $c_{\mathcal{P}}$ de $\mathcal{L}$ telle que la formule suivante :

$$\exists x\ \mathcal{P}(x) \quad \rightarrow \quad \mathcal{P}(c_{\mathcal{P}})$$

soit une formule de $\mathcal{T}$.

Remarques.

- On pourrait aussi écrire : il existe $c \in \mathcal{L}$ tel que :
$$\text{« } (\exists x\ \mathcal{P}(x)) \ \rightarrow\ \mathcal{P}(c) \text{ » } \in\ \mathcal{T},$$
si on ne souhaite pas insister sur la dépendance de c vis-à-vis de $\mathcal{P}$.
- Avec des témoins de Henkin, une formule « $\exists x\ \mathcal{P}(x)$ » n'est plus une formule abstraite car on peut en obtenir une réalisation concrète « $\mathcal{P}(c_{\mathcal{P}})$ », d'où le nom de « témoin ».

Voici encore deux hypothèses supplémentaires, cette fois concernant le langage $\mathcal{L}$:

(iv) $\mathcal{L}$ est un langage dénombrable (l'ensemble des symboles de $\mathcal{L}$ est dénombrable). Cela entraîne que l'ensemble des formules de $\mathcal{L}$ est aussi dénombrable (voir le chapitre « Théorème de compacité »).

(v) $\mathcal{L}$ ne contient pas le prédicat d'égalité « $=$ ». Sans cette hypothèse, on augmenterait la technicité de la preuve car on devrait travailler avec des classes d'équivalence de la relation définie par $t_1 \sim t_2$ lorsque $t_1 = t_2$ est vérifié.

Malgré ces hypothèses simplificatrices, la preuve reste ardue.

3.2. Construction du modèle

Pour une théorie $\mathcal{T}$ qui vérifie les hypothèses (i), (ii), (iii) dans un langage $\mathcal{L}$ qui vérifie les hypothèses (iv), (v), nous allons construire un modèle $\mathcal{S}$ pour $\mathcal{T}$, c'est-à-dire une structure $\mathcal{S}$ telle que $\vDash_{\mathcal{S}} \mathcal{T}$.
Construire la structure $\mathcal{S}$, c'est donner :

- un domaine E,
- une interprétation $c^{\mathcal{S}}$ de chaque constante c de $\mathcal{L}$,
- une interprétation $f^{\mathcal{S}}$ de chaque fonction f de $\mathcal{L}$,
- une interprétation $P^{\mathcal{S}}$ de chaque prédicat P de $\mathcal{L}$.

Ensuite, il faudra démontrer que toute formule $\mathcal{P}$ de $\mathcal{T}$ est vraie lorsqu'elle est interprétée dans $\mathcal{S}$.

C'est parti ! La construction est plus ou moins automatique. On définit les objets afin d'avoir les propriétés voulues. Voici la structure $\mathcal{S}$:

- Le domaine E est l'ensemble des termes fermés de $\mathcal{L}$. On rappelle que les termes sont formés à partir des constantes et des variables, auxquelles on peut appliquer des fonctions de manière récursive. Ici, les termes fermés sont donc uniquement les constantes et les itérations de fonctions appliquées à des constantes. Exemples de termes fermés : c_1, c_2, $f(c_1)$, $g(c_1, c_2)$, $f(f(c_1))$, $g(c_1, f(c_2))$, etc.
  Remarque : si on avait l'égalité comme élément du langage, il faudrait prendre pour E l'ensemble des classes d'équivalence des termes fermés modulo la relation d'équivalence $t_1 \sim t_2$ ssi $\mathcal{T} \vdash (t_1 = t_2)$.
- Si c est un symbole de constante de $\mathcal{L}$, alors c est un terme fermé donc un élément de E. On définit simplement $c^{\mathcal{S}} = c$.
- Soit f un symbole de fonction n-aire de $\mathcal{L}$. Définir $f^{\mathcal{S}}$, c'est définir une « vraie » fonction $f^{\mathcal{S}} : E^n \rightarrow E$. Soient $t_1, \ldots, t_n$ des éléments de E. Ce sont des termes fermés de $\mathcal{L}$, donc $f(t_1, \ldots, t_n)$ est aussi un terme fermé, donc un élément de E. On pose $f^{\mathcal{S}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.
- Soit P un prédicat n-aire de $\mathcal{L}$. Définir l'interprétation $P^{\mathcal{S}}$, c'est définir une fonction $P^{\mathcal{S}} : E^n \rightarrow \{V, F\}$. Voici comment attribuer une valeur vrai/faux à chaque élément $(t_1, \ldots, t_n) \in E^n$: on pose $P^{\mathcal{S}}(t_1, \ldots, t_n) = V$ si $\mathcal{T} \vdash P(t_1, \ldots, t_n)$ et on pose $P^{\mathcal{S}}(t_1, \ldots, t_n) = F$ sinon.
  On a ainsi par construction : $\vDash_{\mathcal{S}} P(t_1, \ldots, t_n)$ ssi $\mathcal{T} \vdash P(t_1, \ldots, t_n)$. C'est une assertion que l'on généralisera plus tard à une formule quelconque.

3.3. Satisfaction

Il reste à montrer que notre structure $\mathcal{S}$ est bien un modèle pour $\mathcal{T}$, c'est-à-dire $\vDash_{\mathcal{S}} \mathcal{T}$. La preuve repose sur le résultat suivant :

Lemme 2.
Soit $\mathcal{Q}$ une formule fermée.

$$\mathcal{T} \vdash \mathcal{Q} \quad \textit{ssi} \quad \vDash_{\mathcal{S}} \mathcal{Q}$$

Admettons un instant ce lemme et justifions que $\mathcal{S}$ est un modèle de $\mathcal{T}$. En effet soit $\mathcal{P}$ une formule de $\mathcal{T}$, alors bien sûr $\mathcal{T} \vdash \mathcal{P}$ donc par le lemme $\vDash_{\mathcal{S}} \mathcal{P}$. Ceci étant vrai pour toute formule de $\mathcal{T}$, on a bien $\vDash_{\mathcal{S}} \mathcal{T}$.

Démonstration du lemme. Pour simplifier, et sans perte de généralité, on suppose que la formule $\mathcal{Q}$ ne contient que les connecteurs $\neg$ et $\wedge$ ainsi que le quantificateur $\exists$. Pour les connecteurs, on renvoie au chapitre « Quels connecteurs sont utiles ? » afin de se ramener à ces deux connecteurs. Si on a une formule « $\forall x\ \mathcal{P}(x)$ », alors elle équivaut à « $\neg(\exists x\ \neg\mathcal{P}(x))$ » ce qui permet de transformer tous les quantificateurs $\forall$ en des quantifications $\exists$ (avec des négations en plus).
Démontrons le résultat par induction sur la construction de la formule $\mathcal{Q}$.

- Cas $\mathcal{Q}$ est une formule atomique.
  Alors $\mathcal{Q} = P(t_1, \ldots, t_n)$ où P est un prédicat et $t_1, \ldots, t_n$ des termes fermés. Par construction de $\mathcal{S}$,
  $$\mathcal{T} \vdash P(t_1, \ldots, t_n) \quad \text{ssi} \quad \vDash_{\mathcal{S}} P(t_1, \ldots, t_n),$$
  donc le lemme est vrai dans ce cas.
- Cas $\mathcal{Q} = \neg\mathcal{P}$.
  Par récurrence on suppose le lemme vrai pour $\mathcal{P}$ et on veut le prouver pour $\mathcal{Q} = \neg\mathcal{P}$.
  $$\begin{array}{lll} & \mathcal{T} \vdash \mathcal{Q} & \\ \text{ssi} & \mathcal{T} \vdash \neg\mathcal{P} & \\ \text{ssi} & \text{non}(\mathcal{T} \vdash \mathcal{P}) & \text{par le Lemme 1, car } \mathcal{T} \text{ est complète} \\ \text{ssi} & \text{non}(\vDash_{\mathcal{S}} \mathcal{P}) & \text{par l'hypothèse de récurrence} \\ \text{ssi} & \vDash_{\mathcal{S}} \neg\mathcal{P} & \\ \text{ssi} & \vDash_{\mathcal{S}} \mathcal{Q} & \end{array}$$
  Le point clé est le passage de la deuxième à la troisième ligne qui utilise l'hypothèse (ii) $\mathcal{T}$ complète via le Lemme 1.
- Cas $\mathcal{Q} = \mathcal{P}_1 \wedge \mathcal{P}_2$.
  Par récurrence on suppose le lemme valide pour $\mathcal{P}_1$ et $\mathcal{P}_2$.
  $$\begin{array}{lll} & \mathcal{T} \vdash \mathcal{Q} & \\ \text{ssi} & \mathcal{T} \vdash \mathcal{P}_1 \wedge \mathcal{P}_2 & \\ \text{ssi} & (\mathcal{T} \vdash \mathcal{P}_1 \ \text{ et } \ \mathcal{T} \vdash \mathcal{P}_2) & \text{par le Lemme 1} \\ \text{ssi} & (\vDash_{\mathcal{S}} \mathcal{P}_1 \ \text{ et } \ \vDash_{\mathcal{S}} \mathcal{P}_2) & \text{par l'hypothèse de récurrence} \\ \text{ssi} & \vDash_{\mathcal{S}} \mathcal{P}_1 \wedge \mathcal{P}_2 & \\ \text{ssi} & \vDash_{\mathcal{S}} \mathcal{Q} & \end{array}$$
- Cas $\mathcal{Q} = \exists x\ \mathcal{P}(x)$.
  On suppose le lemme vrai pour $\mathcal{P}$.
  — Si $\mathcal{T} \vdash \mathcal{Q}$, c'est-à-dire $\mathcal{T} \vdash \exists x\ \mathcal{P}(x)$. Par l'hypothèse (iii) sur les témoins de Henkin, on sait qu'il existe $c_{\mathcal{P}} \in \mathcal{L}$ tel que « $(\exists x\ \mathcal{P}(x)) \to \mathcal{P}(c_{\mathcal{P}})$ » soit une formule de $\mathcal{T}$. Comme $\mathcal{T} \vdash \exists x\ \mathcal{P}(x)$ et $\mathcal{T} \vdash \big[(\exists x\ \mathcal{P}(x)) \to \mathcal{P}(c_{\mathcal{P}})\big]$, alors $\mathcal{T} \vdash \mathcal{P}(c_{\mathcal{P}})$. Donc par hypothèse de récurrence on a $\vDash_{\mathcal{S}} \mathcal{P}(c_{\mathcal{P}})$, et par définition de la satisfaction pour $\exists$, on en déduit $\vDash_{\mathcal{S}} \exists x\ \mathcal{P}(x)$, c'est-à-dire $\vDash_{\mathcal{S}} \mathcal{Q}$.
  — Si $\vDash_{\mathcal{S}} \mathcal{Q}$, alors $\vDash_{\mathcal{S}} \exists x\ \mathcal{P}(x)$, donc il existe un terme fermé t, dont l'interprétation est $t^{\mathcal{S}} \in E$, tel que $\vDash_{\mathcal{S}} \mathcal{P}(t)$. Par hypothèse de récurrence on a $\mathcal{T} \vdash \mathcal{P}(t)$. Ainsi par la règle d'introduction de $\exists$ on a bien $\mathcal{T} \vdash \exists x\ \mathcal{P}(x)$ donc $\mathcal{T} \vdash \mathcal{Q}$.

□

4. Construction d'une théorie complète

Expliquons comment rendre une théorie complète en ajoutant des formules.

4.1. Construction

Soit $\mathcal{T}$ une théorie cohérente sur un langage $\mathcal{L}$. Nous allons construire une théorie $\mathcal{T}'$, sur le même langage $\mathcal{L}$, telle que $\mathcal{T} \subset \mathcal{T}'$ et $\mathcal{T}'$ est complète. On va supposer que le langage $\mathcal{L}$ est dénombrable, donc l'ensemble des formules l'est aussi. Considérons une énumération $\mathcal{P}_0, \mathcal{P}_1, \ldots$ de toutes les formules fermées de $\mathcal{L}$. On construit $\mathcal{T}'$ comme la limite de $(\mathcal{T}_n)_{n\geqslant 0}$ définie par récurrence :

- $\mathcal{T}_0 = \mathcal{T}$
- — $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$ si cette union est cohérente,
  — $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg\mathcal{P}_n\}$ sinon.

Enfin on pose $\mathcal{T}' = \bigcup_{n\geqslant 0} \mathcal{T}_n$.

Proposition 2.
Chaque $\mathcal{T}_n$ est cohérente et $\mathcal{T}'$ est complète.

4.2. Démonstration

Démonstration.

- Montrons par récurrence que les $\mathcal{T}_n$ sont cohérentes. C'est vrai pour $\mathcal{T}_0 = \mathcal{T}$ par hypothèse. Supposons que ce soit vrai pour $\mathcal{T}_n$. Dans le premier cas, $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$ est cohérente par construction. Dans le second cas, $\mathcal{T}_n \cup \{\mathcal{P}_n\}$ n'est pas cohérente, donc $\mathcal{T}_n \vdash \neg\mathcal{P}_n$ (Proposition 1). Si $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg\mathcal{P}_n\}$ n'était pas cohérente non plus, alors $\mathcal{T}_n \vdash \neg(\neg\mathcal{P}_n)$ donc $\mathcal{T}_n \vdash \mathcal{P}_n$. On aurait alors $\mathcal{T}_n \vdash \neg\mathcal{P}_n$ et $\mathcal{T}_n \vdash \mathcal{P}_n$, ce qui contredit l'hypothèse de récurrence que $\mathcal{T}_n$ est cohérente. Ainsi, dans tous les cas, $\mathcal{T}_{n+1}$ est cohérente.
- $\mathcal{T}'$ est cohérente. En effet, sinon on aurait $\mathcal{T}' \vdash \bot$. Comme une preuve est constituée d'un nombre fini d'étapes, donc d'un nombre fini de formules, ces formules sont incluses dans un certain $\mathcal{T}_n$, pour n assez grand. On aurait alors $\mathcal{T}_n \vdash \bot$, ce qui contredit la cohérence de $\mathcal{T}_n$.
- $\mathcal{T}'$ est complète. En effet, soit $\mathcal{Q}$ une formule fermée de $\mathcal{L}$. C'est l'une des $\mathcal{P}_n$. Donc, $\mathcal{P}_n \in \mathcal{T}_{n+1}$ ou $\neg\mathcal{P}_n \in \mathcal{T}_{n+1}$. Ainsi $\mathcal{Q}$ ou $\neg\mathcal{Q}$ est un élément de $\mathcal{T}'$. Par conséquent, $\mathcal{T}' \vdash \mathcal{Q}$ ou $\mathcal{T}' \vdash \neg\mathcal{Q}$.

□

Remarque utile pour la suite : si $\mathcal{T}$ admet des témoins de Henkin, alors $\mathcal{T}'$ aussi. En effet « $(\exists x\ \mathcal{P}(x)) \to \mathcal{P}(c_\mathcal{P})$ » est dans $\mathcal{T}$ donc dans $\mathcal{T}'$.

5. Construction des témoins

Partons d'une théorie $\mathcal{T}$ d'un langage $\mathcal{L}$. Nous allons étendre le langage afin d'obtenir une nouvelle théorie $\mathcal{T}'$ sur un langage $\mathcal{L}'$ qui admet des témoins de Henkin, c'est-à-dire :

$$\text{« } \exists x\ \mathcal{P}(x) \quad \to \quad \mathcal{P}(c_\mathcal{P}) \text{ »} \quad \text{ est une formule de } \mathcal{T}'.$$

En plus $\mathcal{T}'$ doit être cohérente, comme $\mathcal{T}$.

5.1. Extension du langage et de la théorie

$\mathcal{L}'$ sera construit à partir de $\mathcal{L}$ en ajoutant de nouveaux symboles de constantes c_0, c_1, … Avant de donner la preuve entière, voici à quoi correspond l'étape fondamentale.
Soit $\mathcal{T}$ une théorie cohérente d'un langage $\mathcal{L}$. Soit $\mathcal{P}$ une formule de $\mathcal{L}$ ayant x pour seule variable libre. Soit $c_\mathcal{P}$ un nouveau symbole de constante (qui n'est pas dans $\mathcal{L}$). On note $\mathcal{L}' = \mathcal{L} \cup \{c_\mathcal{P}\}$.

Affirmation. $\mathcal{T}' = \mathcal{T} \cup \{\exists x\ \mathcal{P}(x) \to \mathcal{P}(c_\mathcal{P})\}$ est encore cohérente.
On a par construction un témoin de Henkin (juste pour la formule $\mathcal{P}$) et on a conservé la cohérence.

Démonstration. Par l'absurde, si $\mathcal{T}'$ n'est pas cohérente :

$$\begin{aligned}
& \mathcal{T} \cup \{\exists x\ \mathcal{P}(x) \to \mathcal{P}(c_\mathcal{P})\} \vdash \bot \\
\text{donc}\quad & \mathcal{T} \vdash \neg(\exists x\ \mathcal{P}(x) \to \mathcal{P}(c_\mathcal{P})) \\
\text{donc}\quad & \mathcal{T} \vdash \exists x\ \mathcal{P}(x) \ \wedge\ \neg\mathcal{P}(c_\mathcal{P}) \qquad \text{car } \neg(\mathcal{A} \to \mathcal{B}) \equiv \mathcal{A} \wedge \neg\mathcal{B} \\
\text{donc}\quad & \mathcal{T} \vdash \exists x\ \mathcal{P}(x) \ \text{ et } \ \mathcal{T} \vdash \neg\mathcal{P}(c_\mathcal{P}) \\
\text{donc}\quad & \mathcal{T} \vdash \exists x\ \mathcal{P}(x) \ \text{ et } \ \mathcal{T} \vdash \forall x\ \neg\mathcal{P}(x) \qquad \text{par la règle d'introduction de } \forall \\
\text{donc}\quad & \mathcal{T} \vdash \exists x\ \mathcal{P}(x) \ \text{ et } \ \mathcal{T} \vdash \neg\big(\exists x\ \mathcal{P}(x)\big) \qquad \text{par la négation de } \forall \\
\text{donc}\quad & \mathcal{T} \vdash \mathcal{Q} \ \text{ et } \ \mathcal{T} \vdash \neg\mathcal{Q} \qquad \text{pour } \mathcal{Q} = \exists x\ \mathcal{P}(x)
\end{aligned}$$

Ce qui contredit $\mathcal{T}$ cohérente. Conclusion : $\mathcal{T}'$ est cohérente. Noter l'introduction de $\forall$, afin de passer de « $\neg\mathcal{P}(c_\mathcal{P})$ » à « $\forall x \neg\mathcal{P}(x)$ » qui est légitime car la constante $c_\mathcal{P}$ n'apparaît pas dans $\mathcal{T}$. En effet, $\mathcal{T}$ sont des formules de $\mathcal{L}$, qui ne contient pas la constante $c_\mathcal{P}$. □

On pourrait penser faire ceci à chaque formule $\mathcal{P}_0$, $\mathcal{P}_1$,… de $\mathcal{L}$, mais ce sera un peu plus délicat que cela. En effet à chaque étape on augmente le langage, donc il y a de plus en plus de formules ce qui complique la récurrence.

5.2. Construction

Soit $\mathcal{T}$ une théorie cohérente d'un langage $\mathcal{L}$. On va construire par récurrence une suite de théories $(\mathcal{T}_n)_{n\geqslant 0}$ et une suite de langages $(\mathcal{L}_n)_{n\geqslant 0}$ de la façon suivante :

- $\mathcal{L}_0 = \mathcal{L}$ et $\mathcal{T}_0 = \mathcal{T}$,
- puis pour $n \geqslant 0$:
  - $\mathcal{L}_{n+1} = \mathcal{L}_n \cup \{c_\mathcal{P}\}_\mathcal{P}$ où l'union porte sur toutes les formules $\mathcal{P}$ de $\mathcal{L}_n$ ayant une seule variable libre,
  - $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\exists x\ \mathcal{P}(x) \to \mathcal{P}(c_\mathcal{P})\}_\mathcal{P}$.

Enfin, on pose $\mathcal{L}' = \bigcup_{n\geqslant 0} \mathcal{L}_n$ et $\mathcal{T}' = \bigcup_{n\geqslant 0} \mathcal{T}_n$.

Affirmation. $\mathcal{T}'$ admet des témoins de Henkin.

Démonstration. Soit $\mathcal{P}$ une formule de $\mathcal{L}'$ ayant une seule variable libre x. Elle contient un nombre fini de symboles de $\mathcal{L}'$, donc pour un n suffisamment grand, $\mathcal{P}$ est une formule de $\mathcal{L}_n$. Donc $\exists x\ \mathcal{P}(x) \to \mathcal{P}(c_\mathcal{P})$ est une formule de $\mathcal{T}_{n+1}$ donc de $\mathcal{T}'$. □

Affirmation. $\mathcal{T}'$ est cohérente.

Démonstration. On va montrer que chaque $\mathcal{T}_n$ est cohérente. Cela prouvera que $\mathcal{T}'$ est cohérente. En effet si on avait $\mathcal{T}' \vdash \bot$, alors une preuve de ceci ne fait intervenir qu'un nombre fini de formules de $\mathcal{T}'$, qui sont donc toutes incluses dans un $\mathcal{T}_n$ pour un n assez grand. Pour ce n, on a $\mathcal{T}_n \vdash \bot$ ce qui contredit $\mathcal{T}_n$ cohérente.
La preuve que les $\mathcal{T}_n$ sont cohérentes se fait par récurrence, sur le même principe que l'étape fondamentale expliquée pour une seule formule au début de cette section.

Montrons par récurrence que $\mathcal{T}_n$ (pour $n \geqslant 0$) est cohérente. Pour $n = 0$, $\mathcal{T}_0 = \mathcal{T}$, alors $\mathcal{T}_0$ est cohérente par hypothèse.

Supposons $\mathcal{T}_n$ cohérente et par l'absurde que $\mathcal{T}_{n+1}$ ne l'est pas :

$$\mathcal{T}_n \cup \bigcup_{i=1}^{k} \{\exists x\ \mathcal{P}_i(x) \to \mathcal{P}_i(c_{\mathcal{P}_i})\} \vdash \bot \qquad \text{(avec un nombre fini de } \mathcal{P}_i \text{ impliquées dans la preuve)}$$

donc $\mathcal{T}_n \cup \Big\{ \bigwedge_{i=1}^{k} \big(\exists x\ \mathcal{P}_i(x) \to \mathcal{P}_i(c_{\mathcal{P}_i})\big)\Big\} \vdash \bot$

donc $\mathcal{T}_n \vdash \neg \bigwedge_{i=1}^{k} \big(\exists x\ \mathcal{P}_i(x) \to \mathcal{P}_i(c_{\mathcal{P}_i})\big)$

donc $\mathcal{T}_n \vdash \bigvee_{i=1}^{k} \neg\big(\exists x\ \mathcal{P}_i(x) \to \mathcal{P}_i(c_{\mathcal{P}_i})\big)$

donc $\mathcal{T}_n \vdash \neg\big(\exists x\ \mathcal{P}_j(x) \to \mathcal{P}_j(c_{\mathcal{P}_j})\big)$ pour un certain j (par le Lemme 1)

donc $\mathcal{T}_n \vdash \exists x\ \mathcal{P}_j(x)$ et $\mathcal{T}_n \vdash \neg\mathcal{P}_j(c_{\mathcal{P}_j})$

donc $\mathcal{T}_n \vdash \exists x\ \mathcal{P}_j(x)$ et $\mathcal{T}_n \vdash \forall x\ \neg\mathcal{P}_j(x)$ par la règle d'introduction de $\forall$

donc $\mathcal{T}_n \vdash \exists x\ \mathcal{P}_j(x)$ et $\mathcal{T}_n \vdash \neg\big(\exists x\ \mathcal{P}_j(x)\big)$

Donc $\mathcal{T}_n$ est non cohérente. C'est contradictoire avec l'hypothèse de récurrence. Conclusion : $\mathcal{T}_{n+1}$ est cohérente.
Noter encore une fois l'introduction de $\forall$, afin de passer de « $\neg\mathcal{P}_j(c_{\mathcal{P}_j})$ » à « $\forall x\ \neg\mathcal{P}_j(x)$ » qui est légitime car la constante $c_{\mathcal{P}_j}$ n'apparaît pas dans $\mathcal{T}_n$. □

5.3. Preuve complète du théorème

Nous avons donné les détails de la preuve du théorème de complétude mais un peu dans le désordre. Voici le plan de la preuve remis dans le bon sens.

- **Hypothèses.** $\mathcal{T}$ théorie cohérente, $\mathcal{L}$ langage dénombrable.
- **Étape 1 : Ajouter les témoins de Henkin.** On renvoie au début de cette section 5. On obtient une théorie $\mathcal{T}'$ sur un nouveau langage $\mathcal{L}'$ avec $\mathcal{T} \subset \mathcal{T}'$ et $\mathcal{L} \subset \mathcal{L}'$. $\mathcal{T}'$ admet des témoins de Henkin et reste cohérente. $\mathcal{L}'$ est obtenu à partir de $\mathcal{L}$ en ajoutant un nombre dénombrable de constantes $c_{\mathcal{P}}$, ainsi $\mathcal{L}'$ reste dénombrable.
- **Étape 2 : Rendre la théorie complète.** Voir la section 4. À partir de $\mathcal{T}'$ et $\mathcal{L}'$, on obtient une théorie $\mathcal{T}''$ complète avec $\mathcal{T}' \subset \mathcal{T}''$. De plus on préserve les témoins de Henkin (remarque à la fin de la section 4). Le langage est inchangé, on note $\mathcal{L}'' = \mathcal{L}'$.
- **Étape 3 : Construction du modèle.** C'est la section 3 appliquée à $\mathcal{T}''$ et $\mathcal{L}''$. On obtient un modèle $\mathcal{S}''$ de $\mathcal{T}''$ sur $\mathcal{L}''$. Ce modèle se restreint naturellement en un modèle $\mathcal{S}$ de $\mathcal{T}$ sur $\mathcal{L}$. En effet, $\mathcal{L}''$ est obtenu à partir de $\mathcal{L}$ en ajoutant des constantes $c_{\mathcal{P}}$. On considère la structure $\mathcal{S}$ identique à $\mathcal{S}''$ sauf que l'on ne donne pas d'interprétation de ces constantes (qui n'existent pas dans $\mathcal{L}$). Soit $\mathcal{P}$ une formule de $\mathcal{T}$ sur $\mathcal{L}$. C'est donc aussi une formule de $\mathcal{T}''$ sur $\mathcal{L}''$. Comme $\mathcal{S}''$ est un modèle de $\mathcal{T}''$, alors on a $\vDash_{\mathcal{S}''} \mathcal{P}$. Mais comme $\mathcal{P}$ est une formule de $\mathcal{L}$, elle ne contient pas de symboles de ces constantes supplémentaires, donc on a aussi $\vDash_{\mathcal{S}} \mathcal{P}$. Ainsi, $\mathcal{S}$ est un modèle pour $\mathcal{T}$.

Ainsi se termine la preuve du théorème de complétude de Gödel. Ce théorème était le sujet de sa thèse, terminée en 1929 à l'âge de 23 ans et publiée l'année d'après. Ainsi se termine aussi cette partie sur la logique du premier ordre. On continue dans la partie suivante avec ce qui a rendu Gödel célèbre dans le monde entier bien au-delà des mathématiques : le théorème d'incomplétude.

Exercices

Le théorème de complétude

Exercice 1. On rappelle les théorèmes A et B :
- Théorème A. Si $\mathcal{T} \models \mathcal{Q}$, alors $\mathcal{T} \vdash \mathcal{Q}$.
- Théorème B. Si $\mathcal{T}$ est cohérente, alors $\mathcal{T}$ admet un modèle.

On admet le théorème A. En déduire le théorème B.

Exercice 2. Une théorie $\mathcal{T}$ (qui peut contenir une infinité de formules) est dite *finiment cohérente* si pour tout sous-ensemble fini $\Gamma \subset \mathcal{T}$, Γ est cohérente.
Montrer qu'une théorie $\mathcal{T}$ est cohérente si et seulement si elle est finiment cohérente.

Théorie complète

Exercice 3. Soit $\mathcal{L}$ un langage du premier ordre et soit $\mathcal{S}$ une structure pour ce langage. On définit la *théorie* de $\mathcal{S}$, notée $\mathcal{T}(\mathcal{S})$, comme l'ensemble de toutes les formules fermées $\mathcal{P}$ sur $\mathcal{L}$ qui sont vraies dans $\mathcal{S}$:

$$\mathcal{T}(\mathcal{S}) = \left\{\mathcal{P} \text{ formule fermée} \mid \models_{\mathcal{S}} \mathcal{P}\right\}$$

Montrer que $\mathcal{T}(\mathcal{S})$ est une théorie complète.

Exercice 4. Soit $\mathcal{T}$ une théorie cohérente. Montrer l'équivalence entre :
(i) $\mathcal{T}$ est complète,
(ii) pour toute formule fermée $\mathcal{Q}$, si $\mathcal{T} \cup \{\mathcal{Q}\}$ est cohérente, alors $\mathcal{T} \vdash \mathcal{Q}$.

Construction d'un modèle à partir d'une théorie complète

Exercice 5. On considère un langage $\mathcal{L}$ sans symbole d'égalité, contenant une constante 0, un symbole de fonction unaire s, et un prédicat unaire P. Soit $\mathcal{T}$ une théorie complète sur $\mathcal{L}$ qui contient notamment les deux axiomes (fomules) suivants :
- $\mathcal{A}_1 : P(0)$
- $\mathcal{A}_2 : \forall x \ \big(P(x) \leftrightarrow \neg P(s(x))\big)$

1. Construire le modèle $\mathcal{S}$ canonique associé à $\mathcal{T}$ (comme dans la Section 3.2).
2. Construire un modèle $\mathcal{S}'$ avec la structure de domaine $\mathbb{N}$, où $s(x)$ correspond à $x+1$ (le successeur) et expliquer à quoi correspond $P(x)$.

Exercice 6. Compléter la preuve du Lemme 2 pour les connecteurs manquants, ainsi que pour le quantificateur $\exists$, c'est-à-dire pour les formules : $\mathcal{Q} = \mathcal{P}_1 \vee \mathcal{P}_2$, $\mathcal{Q} = \mathcal{P}_1 \to \mathcal{P}_2$, $\mathcal{Q} = \mathcal{P}_1 \leftrightarrow \mathcal{P}_2$, $\mathcal{Q} = \exists x \ \mathcal{P}(x)$.

Construction d'une théorie complète

Exercice 7. On se place dans le cadre plus simple de la logique propositionnelle. Soit un langage $\mathcal{L}$ contenant seulement les deux variables propositionnelles A et B.

1. Justifier que toute formule est une formule fermée. Donner des exemples de formules.
2. Justifier que construire un modèle d'une théorie $\mathcal{T}$, c'est attribuer une valeur V/F à A et une valeur V/F à B qui rendent toutes les formules de $\mathcal{T}$ vraies.
3. Justifier que si $\mathcal{T}$ admet un modèle, alors $\mathcal{T}$ est cohérente.

On veut maintenant construire (le début d') une théorie complète à partir de $\mathcal{T}_0 = \{A\}$. On considère le choix suivant pour l'énumération $(\mathcal{P}_n)_{n\geqslant 0}$ des formules de $\mathcal{L}$:

$$B, \quad A \vee B, \quad \neg A, \quad A \to B, \quad \dots$$

Pour $n \geqslant 0$ on définit :

- $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$ si cette union est cohérente,
- $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg\mathcal{P}_n\}$ sinon.

4. Justifier que $\mathcal{T}_0$ est cohérente.
5. Construire $\mathcal{T}_1, \dots, \mathcal{T}_4$.
6. La théorie complète $\mathcal{T} = \bigcup_{n\geqslant 0} \mathcal{T}_n$ prouvera-t-elle $A \wedge B$?
7. On repart de $\mathcal{T}'_0 = \{A\}$, mais cette fois on considère l'énumération :

   $$\neg B, \quad A \vee B, \quad \neg A, \quad A \to B, \quad \dots$$

   Construire $\mathcal{T}'_1, \dots, \mathcal{T}'_4$.
8. La théorie complète $\mathcal{T}' = \bigcup_{n\geqslant 0} \mathcal{T}'_n$ sera-t-elle identique à $\mathcal{T}$?

Construction des témoins

Exercice 8. On considère un langage initial $\mathcal{L}_0$ contenant un unique prédicat unaire $P(x)$, sans constante ni fonction. Soit la théorie $\mathcal{T}_0$ composée des deux axiomes suivants :

$$\mathcal{A}_1 : \exists x\ P(x) \qquad \text{et} \qquad \mathcal{A}_2 : \exists x\ \neg P(x)$$

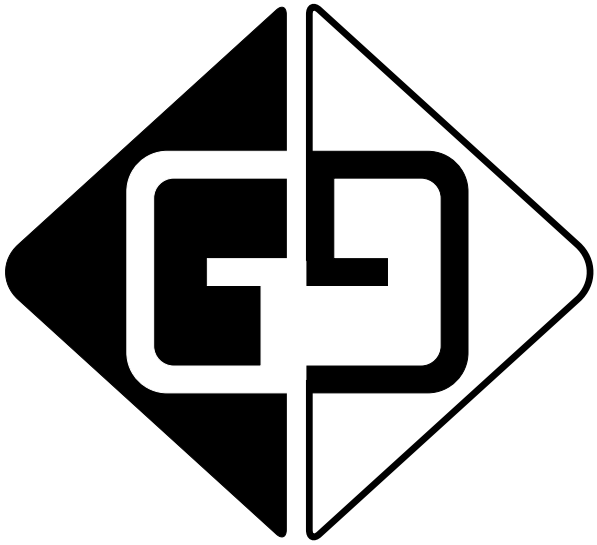
1. Justifier que $\mathcal{T}_0$ est cohérente (par exemple en construire un modèle).
2. Le langage $\mathcal{L}_0$ possède-t-il des termes clos ? La théorie $\mathcal{T}_0$ admet-elle des témoins de Henkin sur $\mathcal{L}_0$?
3. On applique la méthode de Henkin pour les deux formules $P(x)$ et $\neg P(x)$. On note alors $\mathcal{L}_1 = \mathcal{L}_0 \cup \{c_1, c_2\}$ le langage étendu par deux constantes c_1 et c_2. Quels axiomes de Henkin a-t-on ajouté ? Que vaut alors la théorie $\mathcal{T}_1$?
4. Montrer que $\mathcal{T}_1$ prouve $P(c_1) \wedge (\neg P(c_2))$. En déduire que dans tout modèle de $\mathcal{T}_1$, l'interprétation de c_1 et l'interprétation de c_2 sont nécessairement deux éléments distincts du domaine.

Exercice 9. Soit $\mathcal{L}'$ une extension d'un langage $\mathcal{L}$. Soit $\mathcal{T}$ une théorie sur le langage $\mathcal{L}'$, qui est complète et qui admet des témoins de Henkin. Soit $\mathcal{P}(x)$ une formule du langage initial $\mathcal{L}$ (ayant x pour seule variable libre).
Montrer que si $\mathcal{T}$ ne prouve pas $\forall x\ \mathcal{P}(x)$, alors il existe une constante de Henkin c du langage $\mathcal{L}'$ telle que :

$$\mathcal{T} \quad \vdash \quad \neg\mathcal{P}(c).$$

Comment appelle-t-on c en langage courant ?

Troisième partie



l'incomplétude de Gödel

Chapitre 12 — Un aperçu de l'incomplétude de Gödel

Avant de se lancer dans la construction des outils nécessaires à l'énoncé précis des théorèmes de Gödel et à leur preuve, on donne un aperçu d'ensemble des théorèmes d'incomplétude et du long chemin pour y parvenir.

1. Énoncés des théorèmes

1.1. Énoncé pour l'arithmétique des entiers

Voici une version encore informelle du premier théorème d'incomplétude.

Théorème 1.
Dans la théorie des entiers naturels, il existe des formules vraies mais non démontrables.

On note $\mathcal{PA}$ la théorie de l'arithmétique des entiers. C'est une liste de formules, appelées *axiomes de Peano* (par exemple « $x+0=x$ » et « $x\times 0=0$ »). Ils sont construits de sorte que les entiers $\mathbb{N}$ satisfont naturellement ces axiomes.
Gödel ne parlait en fait pas de vérité dans son énoncé afin d'éviter les discussions du type paradoxe du menteur qui affirme « Je suis un menteur ». Il construit plutôt une formule $\mathcal{G}$ telle que ni $\mathcal{G}$, ni $\neg\mathcal{G}$ n'est démontrable. On parle alors d'une théorie *incomplète* (d'où le nom du théorème).
Maintenant, si on se place sur le modèle $\mathbb{N}$ des entiers naturels, qui satisfont les axiomes de Peano, alors soit $\mathcal{G}$ est vraie, soit $\neg\mathcal{G}$ est vraie, et on a bien l'une des deux formules qui est vraie mais non démontrable.

1.2. Théorème général

On pourrait penser que c'est la faute des axiomes de Peano qui sont mal choisis ou pas assez nombreux. Mais le théorème de Gödel est en fait plus général.

Théorème 2.
Toute théorie $\mathcal{T}$ cohérente, contenant $\mathcal{PA}$ et ayant des axiomes récursifs est incomplète.

Oublions un moment les hypothèses et revenons à l'arithmétique de Peano. Puisque cette théorie est incomplète, il existe une formule $\mathcal{P}_0$ non démontrable dans $\mathcal{PA}$. Notons $\mathcal{T}_0=\mathcal{PA}$ et $\mathcal{T}_1=\mathcal{T}_0\cup\{\mathcal{P}_0\}$ la nouvelle théorie obtenue en ajoutant à $\mathcal{T}_0$ la formule indémontrable $\mathcal{P}_0$.
On a maintenant $\mathcal{P}_0$ démontrable dans $\mathcal{T}_1$: $\mathcal{T}_1\vdash\mathcal{P}_0$, car $\mathcal{P}_0$ est l'une des formules de $\mathcal{T}_1$. Malheureusement la version générale du théorème d'incomplétude affirme que cette nouvelle théorie $\mathcal{T}_1$ est aussi incomplète : il existe une formule $\mathcal{P}_1$ telle que ni $\mathcal{P}_1$, ni $\neg\mathcal{P}_1$ ne sont démontrables. On pourrait rajouter $\mathcal{P}_1$ (ou $\neg\mathcal{P}_1$) afin de construire une théorie $\mathcal{T}_2$,… mais on ne fait que repousser le problème à l'infini.
Ainsi, on aura beau rajouter une à une des formules indémontrables comme axiomes, il existera toujours des formules indémontrables. C'est l'incomplétude !

1.3. Notions en jeu

Revenons sur les définitions des termes des hypothèses et de la conclusion.

Complétude.

- Une théorie $\mathcal{T}$ est ***incomplète*** s'il existe une formule fermée $\mathcal{Q}$ telle que :
$$\mathcal{T} \nvdash \mathcal{Q} \quad \text{et} \quad \mathcal{T} \nvdash \neg\mathcal{Q}$$
- Si on se place dans un ***modèle*** de $\mathcal{T}$, c'est-à-dire une structure où toutes les formules de $\mathcal{T}$ sont satisfaites, alors dans ce cas, un des énoncés $\mathcal{Q}$ ou $\neg\mathcal{Q}$ est vrai mais pas démontrable.

Cohérence.

- Une théorie $\mathcal{T}$ est ***incohérente*** s'il existe une formule fermée $\mathcal{Q}$ telle que :
$$\mathcal{T} \vdash \mathcal{Q} \quad \text{et} \quad \mathcal{T} \vdash \neg\mathcal{Q}$$
Elle est dite ***cohérente*** sinon.
- Notez bien que la définition de « incohérente » n'est pas la négation de « incomplète » car la négation d'un « et » est un « ou ».
- Ainsi, une théorie incohérente prouve $\mathcal{T} \vdash (\mathcal{Q} \wedge \neg\mathcal{Q})$ donc $\mathcal{T} \vdash \bot$. Alors par la règle d'explosion, $\mathcal{T} \vdash \mathcal{Q}$ (quelle que soit la formule $\mathcal{Q}$) donc on a aussi $\mathcal{T} \vdash \neg\mathcal{Q}$.
- Une théorie incohérente n'a donc pas d'intérêt puisqu'elle ne peut pas avoir de modèle. En effet, si elle possédait un modèle, et comme $\mathcal{T} \vdash \mathcal{Q}$ et $\mathcal{T} \vdash \neg\mathcal{Q}$, par le théorème de validité, alors $\mathcal{Q}$ et $\neg\mathcal{Q}$ devraient être toutes les deux vraies, ce qui fournit une contradiction.
- Autrement dit si $\mathcal{T}$ admet un modèle alors $\mathcal{T}$ est cohérente.

Contenir $\mathcal{PA}$.

- Une théorie est un ensemble de formules, appelées axiomes.
- $\mathcal{PA}$ est une théorie spécifique dont les formules sont les axiomes de Peano qui modélisent les entiers naturels $\mathbb{N}$.
- L'hypothèse « $\mathcal{T}$ contient $\mathcal{PA}$ » signifie que tous les axiomes de Peano doivent être des axiomes de $\mathcal{T}$.
- On dit alors que $\mathcal{T}$ est « suffisamment riche », c'est-à-dire qu'on doit pouvoir travailler avec les entiers naturels. C'est une exigence relativement faible !

Axiomes récursifs

- L'hypothèse que $\mathcal{T}$ a des axiomes récursifs (le terme courant est « récursivement axiomatisable ») est plus technique. Cela signifie que l'on doit pouvoir reconnaître les axiomes.
- Plus précisément pour une formule donnée, on doit pouvoir décider par un calcul (ou un algorithme) si c'est un axiome de $\mathcal{T}$ ou pas.
- Encore une fois, c'est une hypothèse raisonnable : pour décider quels théorèmes on peut prouver il faut pouvoir identifier les axiomes qui servent d'hypothèse !

1.4. Second théorème

Théorème 3.

Une théorie ne peut prouver sa propre cohérence.

Cela signifie concrètement qu'on ne peut espérer vérifier les hypothèses du théorème d'incomplétude en restant dans la théorie $\mathcal{T}$, il faudrait se placer dans un cadre plus large.

Par exemple, on considère généralement que $\mathcal{PA}$ est une théorie qui a un modèle. En réalité, on a accepté implicitement l'existence des entiers naturels $\mathbb{N}$. Pour avoir une construction rigoureuse de $\mathbb{N}$, il faut se placer dans un cadre plus large, par exemple la théorie $\mathcal{ZFC}$ (voir le chapitre « Arithmétiques de Peano »).

2. Conséquences

2.1. Conséquences mathématiques

Autour de 1900, Hilbert souhaite définir un cadre rigoureux pour les fondations de la logique. Dans les années 1800, on a déjà formalisé beaucoup de notions mathématiques : les réels, les limites, la continuité,... Il est temps d'aller plus loin et d'axiomatiser complètement les mathématiques.
L'objectif est de faire abstraction du monde physique (la continuité c'est tracer le graphe sans lever le crayon) et de n'avoir à écrire que des formules logiques. Cependant, on se retrouve vite confronté à des problèmes, comme le paradoxe de Russell : il n'est pas facile de définir ce qu'est un ensemble.
Le premier et le second théorèmes de Gödel font exploser le programme de Hilbert. Après Gödel, il n'y a plus aucune chance d'obtenir une axiomatisation parfaite :

- d'une part, si on se place dans une théorie (suffisamment riche et cohérente), il y aura des formules vraies mais non démontrables,
- en plus on ne peut même pas prouver que la théorie est cohérente afin de garantir qu'on ne puisse pas prouver une chose et son contraire.

C'est donc fatal pour le programme de Hilbert.

Malgré ce choc, les théorèmes de Gödel ont eu des conséquences positives profondes bien au-delà d'un résultat concernant un petit cercle de logiciens.
Tout d'abord, les outils développés par Gödel sont précurseurs de la théorie de l'informatique. Par exemple, l'arithmétisation de la syntaxe (coder toute formule par un nombre) est une idée naturelle dans le monde numérique dans lequel nous vivons aujourd'hui, mais ce n'était pas le cas en 1930.
Plus généralement, Gödel s'intéresse à savoir ce qui est calculable, bien avant la notion d'algorithme et les travaux de Turing (1936) sur un ordinateur théorique, la machine de Turing, et l'indécidabilité.

L'hypothèse du continu.
Même si Gödel construit une formule $\mathcal{G}$ explicite qui n'est pas démontrable, cette formule a pour seul intérêt de prouver le théorème.
L'*hypothèse du continu* est un énoncé beaucoup plus concret qui est, lui aussi, un énoncé indécidable (ni $\mathcal{HC}$ ni $\neg\mathcal{HC}$ ne sont démontrables dans $\mathcal{ZFC}$).
Cela a été prouvé par Cohen (en 1963), résolvant ainsi le premier de la liste des 23 problèmes de Hilbert.
Expliquons ce problème.

- Les ensembles $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ sont dénombrables (en bijection avec $\mathbb{N}$), ils ont donc le même cardinal, même si ce sont des ensembles infinis. On note ce cardinal $\aleph_0$ (*aleph* est la première lettre de l'alphabet hébreu).
- Les ensembles $\mathbb{R}$, $[0,1]$, $\mathbb{R}^2$, $\mathbb{C}$ sont indénombrables, on note $2^{\aleph_0}$ leur cardinal (voir plus loin pour l'explication de cette notation).
- L'hypothèse du continu affirme qu'il n'y a pas d'autre type d'infini entre $\aleph_0$ et $2^{\aleph_0}$.
- Autrement dit, si on note $\aleph_1$ le premier type d'infini strictement plus grand que $\aleph_0$, alors on a $\aleph_1 = 2^{\aleph_0}$.
- Cohen prouve donc qu'on ne peut pas répondre à cette question, ce qui fait qu'on peut choisir d'accepter ou de refuser $\mathcal{HC}$ indépendamment du reste.

Revenons sur cette notation $2^{\aleph_0}$ pour le cardinal de $\mathbb{R}$.
Pour E, F deux ensembles, on note parfois F^E l'ensemble des fonctions de $f : E \to F$. Cela se justifie par le fait que si E et F sont finis, le nombre de telles fonctions est $\mathrm{Card}(F)^{\mathrm{Card}(E)}$. Par exemple, si $F = \{0,1\}$ et $\mathrm{Card}(E) = n$, alors le nombre de fonctions $f : E \to \{0,1\}$ est 2^n.
Expliquons comment les réels de $[0,1]$ peuvent être décrits par des fonctions de $\mathbb{N} \to \{0,1\}$. Pour $x \in [0,1]$, on calcule l'écriture binaire : $x = 0.100110101\ldots = \sum_{n=1}^{\infty} a_n 2^{-n}$ où $a_n \in \{0,1\}$. C'est comme

l'écriture décimale à virgule mais avec seulement les chiffres 0 et 1. Ainsi, à chaque écriture binaire, on peut associer une fonction $f : \mathbb{N} \to \{0,1\}$ définie par $f(n) = a_n$. Cette correspondance est (à peu près) bijective.
Ainsi le cardinal de $[0,1]$ est celui de $\{0,1\}^{\mathbb{N}}$ qui est $2^{\aleph_0}$ (où $\aleph_0$ désigne le cardinal de $\mathbb{N}$).
C'est aussi l'occasion de rappeler le théorème de Cantor : $\text{Card}(\mathcal{P}(E)) > \text{Card}(E)$ quel que soit l'ensemble E. Il est facile de voir que $\mathcal{P}(\mathbb{N})$ correspond à $\{0,1\}^{\mathbb{N}}$. En effet, $A \subset \mathbb{N}$ définit naturellement un réel $x = 0.0110001\ldots$ où on a 1 à la n-ème place exactement lorsque $n \in A$. Ainsi $[0,1]$ a le même cardinal que $\mathcal{P}(\mathbb{N})$: $2^{\aleph_0}$ et ce cardinal est strictement plus grand que $\aleph_0$.

2.2. Conséquence philosophique

Les théorèmes d'incomplétude sont parmi les rares résultats mathématiques modernes connus par les non-mathématiciens pour leur influence sur certains courants de pensée.
Ces théorèmes rendent indéniable la différence fondamentale entre ce qui est vrai et ce qui est démontrable. Cela renforce le point de vue des platoniciens : la vérité est une réalité absolue, indépendante du monde qui nous entoure.
Les théorèmes de Gödel prouvent aussi que les mathématiques ne sont pas qu'un jeu d'écriture sur des symboles abstraits, en montrant que la syntaxe (ce que l'on peut démontrer) ne correspond pas forcément à toute la sémantique (ce qui est vrai).
Ils questionnent aussi la différence entre humain et machine. Si l'on modélise une machine par une théorie (les axiomes correspondant à des instructions), la machine ne pourra pas sortir de son cadre pour prouver sa propre cohérence. Par exemple, pour la phrase $\mathcal{G}$ de Gödel, qui dit « Je ne suis pas démontrable dans $\mathcal{PA}$ », il faut se placer dans un cadre plus large, par exemple $\mathcal{ZFC}$, pour pouvoir dire que cette phrase est vraie. Est-ce qu'une machine peut englober toute la complexité des raisonnements humains ?

2.3. Ce que ne sont pas les théorèmes d'incomplétude

Complétude et incomplétude.
Les théorèmes de complétude et les théorèmes d'incomplétude sont des théorèmes vrais, prouvés par Gödel, et ne sont pas contradictoires.
On rappelle la complétude, pour toute formule $\mathcal{Q}$:

$$\mathcal{T} \models \mathcal{Q} \quad \text{ssi} \quad \mathcal{T} \vdash \mathcal{Q}$$

Plaçons-nous maintenant dans le cadre d'une théorie $\mathcal{T} = \mathcal{PA}$ (ou contenant $\mathcal{PA}$) ayant pour modèle $\mathbb{N}$.
Le théorème d'incomplétude affirme qu'il existe une formule $\mathcal{Q}$ telle que :

$$\mathcal{T} \not\vdash \mathcal{Q} \quad \text{et} \quad \mathcal{T} \not\vdash \neg\mathcal{Q}$$

Il concerne des théories spécifiques (modélisant $\mathbb{N}$), alors que la complétude vaut quelle que soit la structure $\mathcal{S}$.
L'incomplétude alliée à la complétude prouve que $\mathcal{Q}$ ne peut être vraie pour toutes les structures (car sinon, par la complétude, $\mathcal{Q}$ serait démontrable).
Il existe donc des structures $\mathcal{S}$, différentes de $\mathbb{N}$, dans lesquelles $\mathcal{Q}$ est fausse (donc on n'a pas $\mathcal{T} \models \mathcal{Q}$) même si $\mathcal{S}$ satisfait les axiomes de Peano. On parle de modèle non standard des entiers.

Les mathématiques et la logique sont-elles incohérentes ?
Le second théorème dit qu'une théorie ne peut prouver sa propre cohérence. Cela signifie qu'en restant dans le cadre strict de cette théorie, on ne sera jamais sûr qu'on ne peut pas prouver à la fois une formule $\mathcal{Q}$ et sa négation $\neg\mathcal{Q}$.

Cependant, si on se place dans un cadre plus grand, on peut s'en sortir. Par exemple, on prouve la cohérence de l'arithmétique à partir de la théorie $\mathcal{ZFC}$. Apparaît alors un problème de régression infinie : on ne peut pas être sûr de la cohérence de $\mathcal{ZFC}$. Il faudrait se placer dans une théorie encore plus grande...
Il n'y a donc pas de cadre logique unique et universel au fondement des mathématiques. Le consensus empirique des mathématiques modernes est d'utiliser le cadre $\mathcal{ZFC}$ dans lequel personne n'a trouvé d'incohérence.

3. Plan de travail

Prouver le premier théorème d'incomplétude est un gros travail, assez technique avec quelques idées lumineuses (par exemple la fonction β qui encode une liste quelconque de nombres avec seulement deux entiers !). Même cent ans après, cela reste difficile à aborder. On ne peut être qu'admiratif qu'une seule personne ait fait tout cela.
On présente un par un les prochains chapitres.

3.1. Arithmétique de Peano

On y décrit les 7 axiomes de Peano et le schéma d'axiomes de la récurrence qui régissent le comportement des entiers naturels. On décrira aussi d'autres théories plus faibles (de Presburger, de Robinson) et une théorie plus forte, la théorie de Zermelo-Fraenkel éventuellement augmentée de l'axiome du choix $\mathcal{ZFC}$.

3.2. Codage de Gödel

C'est la partie « arithmétisation de la syntaxe » : il faut coder chaque formule, et même chaque preuve, par un seul entier. On pourra alors donner une preuve informelle, mais qui reste dans l'esprit du premier théorème de Gödel, par un argument de diagonalisation. On en profitera pour motiver cette méthode de diagonalisation via le paradoxe de Russell et l'argument diagonal de Cantor.

3.3. Fonctions récursives primitives

Il s'agit d'un des chapitres techniques. Les fonctions récursives primitives sont les fonctions $f : \mathbb{N}^p \to \mathbb{N}$ « facilement calculables ». Le but est de voir que les opérations sur les formules, par exemple vérifier qu'une suite de formules est bien une preuve, sont facilement calculables.
On abordera la construction de preuves à la Hilbert, plus adaptée que les preuves en déduction naturelle pour un traitement algorithmique.

3.4. Fonctions représentables

Encore un chapitre technique qui fait le passage entre l'univers mathématique des entiers et le langage $\mathcal{L}_0$ de l'arithmétique de Peano. On prouvera que toute fonction récursive primitive peut être traduite par une formule de $\mathcal{L}_0$. On obtient ainsi une formule qui décide si une preuve est valide ou non, ainsi qu'une formule exprimant si un énoncé est démontrable ou pas. C'est dans ce chapitre qu'on expliquera la fonction β de Gödel.

3.5. Diagonalisation

On y expliquera comment obtenir une formule $\mathcal{G}$ qui parle d'elle-même.
On y parlera aussi du paradoxe du menteur et d'un théorème de Tarski qui dit qu'il n'existe pas de formule qui définisse les formules vraies (alors qu'on a vu qu'on sait détecter les formules démontrables).

3.6. Preuve du théorème d'incomplétude

On énoncera et prouvera le premier théorème d'incomplétude d'abord dans le cadre de l'arithmétique de Peano, puis dans le cadre des théories plus larges. La preuve est basée sur le lemme de diagonalisation afin d'obtenir une phrase qui dit « Je ne suis pas démontrable ».

3.7. Le second théorème d'incomplétude

On terminera par l'énoncé du second théorème et on en donnera une preuve sans entrer dans les détails.

Chapitre 13

Arithmétique de Peano

Nous étudions les axiomes qui définissent l'arithmétique usuelle des entiers naturels.

1. Les entiers naturels

1.1. Axiomatisation

L'ensemble $\mathbb{N} = \{0, 1, 2, \ldots\}$ des entiers naturels nous est familier et on en connaît beaucoup de propriétés :

$$x + 0 = x, \quad 0 + x = x, \quad x + y = y + x, \quad \ldots$$

Mais peut-on dégager ce qui fait la « substantifique moelle » de $\mathbb{N}$? Quelles sont les propriétés qui jouent le rôle d'axiomes et quelles sont celles qui peuvent être démontrées à partir de ces axiomes ? On a déjà l'habitude de procéder ainsi pour définir ce qu'est un groupe $(G, \circ)$ ou un espace vectoriel $(E, +, \cdot)$ par des définitions axiomatiques.
On va faire cette démarche pour obtenir les axiomes qui définissent l'arithmétique de $\mathbb{N}$. Il y aura sept axiomes de base (sur l'addition, la multiplication) et une série d'axiomes pour la récurrence : l'ensemble forme l'arithmétique de Peano.
Qu'est-ce qu'un axiome ? De notre point de vue, un ***axiome*** est une formule fermée. Un ensemble d'axiomes est donc une théorie. Pour un ensemble d'axiomes $\mathcal{A}$ et une formule $\mathcal{Q}$, on s'intéresse souvent à savoir si $\mathcal{A} \vdash \mathcal{Q}$. On s'intéresse aussi à l'assertion équivalente suivante (grâce au théorème de complétude) : est-ce que $\mathcal{A} \vDash \mathcal{Q}$? Mais on va surtout se placer dans une structure $\mathcal{S}$ précise et discuter si $\vDash_{\mathcal{S}} \mathcal{A}$ implique $\vDash_{\mathcal{S}} \mathcal{Q}$, c'est-à-dire, si les axiomes sont vrais dans $\mathcal{S}$, est-ce que la formule $\mathcal{Q}$ est aussi vraie dans $\mathcal{S}$? Ainsi, on va utiliser le terme *axiomes* plutôt que *théorie* car on ne cherche pas à étudier les propriétés générales des théories. On va expliciter et fixer un ensemble de formules : les axiomes de l'arithmétique de Peano qui seront notés $\mathcal{PA}$. Les entiers naturels $\mathbb{N}$ donnent un modèle de $\mathcal{PA}$, c'est-à-dire $\vDash_{\mathbb{N}} \mathcal{PA}$.

1.2. Lien avec les théorèmes d'incomplétude

Les théorèmes d'incomplétude de Gödel s'appliquent à des théories $\mathcal{T}$ suffisamment « riches ». Cela signifie concrètement que la théorie doit contenir les axiomes de Peano (c'est-à-dire $\mathcal{PA} \subset \mathcal{T}$; ou doit pouvoir les démontrer). Au final, demander qu'une théorie contienne ces axiomes n'est quand même pas une grande exigence, car il est difficile de faire des mathématiques sans avoir le droit de travailler avec les entiers.
Les entiers naturels interviendront aussi dans les chapitres suivants car la preuve des théorèmes d'incomplétude repose sur une *arithmétisation* : chaque symbole, chaque formule, chaque preuve va être encodée par un entier de $\mathbb{N}$.

1.3. Qu'est-ce que $\mathbb{N}$?

Pour chercher à axiomatiser $\mathbb{N}$, il serait bon de savoir en donner une définition. Malheureusement, ce n'est pas aussi évident.

- Une première approche est de définir un langage $\mathcal{L}$ contenant le symbole de constante 0 et une fonction « successeur » S qui joue le rôle de $S(n) = n+1$. On note alors $1 = S(0)$, $2 = S(1) = S(S(0))$, $3 = S(2)\ldots$ $\mathbb{N}$ est le plus petit ensemble pour lequel c'est possible.
- On peut aussi définir $\mathbb{N}$ à partir de la théorie des ensembles : les entiers sont construits à partir de l'ensemble vide $\varnothing$. On part d'un langage où tout objet est un ensemble (voir le chapitre « Variables et structures »).

$$\begin{aligned} 0 &= \varnothing \\ 1 &= \{\varnothing\} = \{0\} \\ 2 &= \{\varnothing, \{\varnothing\}\} = \{0, 1\} \\ 3 &= \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\} = \{0, 1, 2\} \\ &\ldots \end{aligned}$$

  On définit ainsi $S(n) = n \cup \{n\} = \{0, 1, \ldots, n\}$ désigné par $n+1$.
  On exige que 0 et les $S(n)$ soient dans l'ensemble, $\mathbb{N}$ est alors le plus petit ensemble pour lequel c'est possible. On reviendra sur ce point de vue de la théorie des ensembles lorsqu'on discutera de l'axiomatique $\mathcal{ZF}$.
- $\mathbb{N}$ peut aussi être défini de façon unique dans la logique du *second ordre*, si on écrit l'axiome de la récurrence sous la forme :

$$\forall \mathcal{P} \quad \mathcal{R}ec_{\mathcal{P}(x)}$$

  (Le principe de récurrence $\mathcal{R}ec_{\mathcal{P}(x)}$ sera défini plus tard.) Attention, on sort du cadre de la logique du premier ordre car le « pour tout » ne porte pas sur une variable individuelle, mais sur un sous-ensemble de formules, ce qui est interdit en logique du premier ordre.

Dans tous les cas, même si la définition de $\mathbb{N}$ n'est pas simple, cela correspond bien à l'ensemble que l'on connaît. Cependant, il existe aussi des modèles non standards, c'est-à-dire des ensembles qui vérifient les axiomes de Peano mais qui ne sont pas $\mathbb{N}$. On pourrait en construire à la main, mais ce sera aussi une conséquence du théorème d'incomplétude.

2. Axiomes de Peano

Les axiomes de Peano sont d'abord une liste de sept axiomes qui définissent l'arithmétique élémentaire, suivie d'un schéma d'axiomes pour le principe de récurrence.
On considère le langage du premier ordre $\mathcal{L}_0 = (0, S, +, \times)$ où :

- 0 est un symbole de constante (penser à $0 \in \mathbb{N}$).
- S est une fonction unaire, nommée ***successeur*** (penser à $S(x) = x+1$).
- $+$ est une fonction binaire (penser à l'addition).
- $\times$ est une fonction binaire (penser à la multiplication).
- $=$ est le prédicat d'égalité.

On notera $x \neq y$ pour $\neg(x = y)$. On introduira plus loin les symboles $\leqslant$ et $<$.

2.1. Arithmétique élémentaire

Voici les sept axiomes de base :

- $\mathcal{A}_1$: $\forall x \quad S(x) \neq 0$
- $\mathcal{A}_2$: $\forall x \quad (x = 0) \vee (\exists y \quad x = S(y))$
- $\mathcal{A}_3$: $\forall x \ \forall y \quad (S(x) = S(y)) \rightarrow (x = y)$
- $\mathcal{A}_4$: $\forall x \quad x + 0 = x$
- $\mathcal{A}_5$: $\forall x \ \forall y \quad x + S(y) = S(x + y)$
- $\mathcal{A}_6$: $\forall x \quad x \times 0 = 0$
- $\mathcal{A}_7$: $\forall x \ \forall y \quad x \times S(y) = (x \times y) + x$

L'ensemble de ces axiomes forme l'***arithmétique élémentaire***. On ajoutera juste après les axiomes de récurrence.
Voici des commentaires pour chacun de ces axiomes et leur interprétation dans $\mathbb{N}$:

- $\mathcal{A}_1$: Un successeur n'est jamais nul. Dans $\mathbb{N}$, on a toujours $n+1 \neq 0$. C'est important qu'on ne puisse pas revenir au point de départ, ce qui arriverait par exemple si on cyclait modulo N : $0, 1, 2, \ldots, N-1$, puis 0.
- $\mathcal{A}_2$: Tout élément non nul est un successeur. Dans $\mathbb{N}$, si $n \neq 0$ alors il existe $m \in \mathbb{N}$ tel que $n = m + 1$. (En fait « $m = n - 1$ », mais on n'a pas le droit de faire de soustraction.) Cet axiome est le seul axiome redondant car il se déduit des autres axiomes et du principe de récurrence (voir les exercices). Cependant, outre sa justification historique, il s'avérera important pour l'arithmétique $\mathcal{Q}$ de Robinson (voir en fin de chapitre).
- $\mathcal{A}_3$: Les successeurs de x et y sont égaux si et seulement si x et y sont égaux, autrement dit la fonction S est injective. Avec l'axiome $\mathcal{A}_1$, cela implique qu'en partant de 0, puis $1 = S(0)$, $2 = S(1)\ldots$ on ne crée que des nouveaux éléments. Dans $\mathbb{N}$ cela signifie $(n+1 = m+1) \rightarrow (n = m)$. Noter que par définition d'une fonction, on a toujours l'implication : si $x = y$ alors $S(x) = S(y)$.
- $\mathcal{A}_4$: Ajouter 0 (à droite) ne change rien. Dans $\mathbb{N}$, $n+0 = n$. Noter que pour l'arithmétique de Peano on prouvera que c'est aussi vrai à gauche « $0 + x = x$ », mais cela nécessite l'utilisation de la récurrence. On verra que cela devient problématique pour l'arithmétique $\mathcal{Q}$ de Robinson (voir en fin de chapitre).
- $\mathcal{A}_5$: Le successeur se comporte bien vis-à-vis de l'addition. Dans $\mathbb{N}$ on écrirait $n+(m+1) = (n+m)+1$, une conséquence de l'associativité dans $\mathbb{N}$.
- $\mathcal{A}_6$: Multiplier par 0 donne 0. Dans $\mathbb{N}$, $n \times 0 = 0$. Ce sera aussi vrai à gauche.
- $\mathcal{A}_7$: L'addition se comporte bien vis-à-vis de la multiplication. Dans $\mathbb{N}$, $n \times (m+1) = (n \times m) + n$, une conséquence de la distributivité dans $\mathbb{N}$.

2.2. Axiomes de la récurrence

On ne va pas ajouter un axiome mais une infinité d'axiomes, qu'on appelle un ***schéma d'axiomes***. Il va y avoir un axiome pour chaque formule $\mathcal{P}$ ayant x comme variable libre.
Soit $\mathcal{P}$ une formule de $\mathcal{L}_0$ ayant une variable libre x. On notera $\mathcal{P}(x)$ pour signifier la dépendance de $\mathcal{P}$ vis-à-vis de x. Cela permet aussi d'écrire $\mathcal{P}(y)$ au lieu de $\mathcal{P}[y/x]$.
Le principe de récurrence pour la formule $\mathcal{P}(x)$ s'écrit :

$$\mathcal{R}ec_{\mathcal{P}(x)} : \quad \mathcal{P}(0) \wedge \big[\forall x \quad \mathcal{P}(x) \rightarrow \mathcal{P}(S(x))\big] \quad \rightarrow \quad \forall x \ \mathcal{P}(x)$$

Dans $\mathbb{N}$, on retrouve le principe de récurrence classique :

- si $\mathcal{P}(0)$ est vraie (initialisation),
- et si $\mathcal{P}(n) \rightarrow \mathcal{P}(n+1)$ quel que soit $n \in \mathbb{N}$ (hérédité),
- alors $\mathcal{P}(n)$ est vraie pour tout $n \in \mathbb{N}$ (conclusion).

On note :

$$\mathcal{R}ec = \left\{\mathcal{R}ec_{\mathcal{P}(x)} \mid \mathcal{P} \text{ formule de } \mathcal{L}_0 \text{ ayant une variable libre } x\right\}$$

$\mathcal{R}ec$ contient une infinité (dénombrable) de formules.

Définition.
On note $\mathcal{PA} = \{\mathcal{A}_1, \dots, \mathcal{A}_7\} \cup \mathcal{R}ec$. Ce sont les axiomes de l'***arithmétique de Peano***.

Remarque.
Si on veut être un peu plus strict sur l'écriture, on peut écrire :

$$\mathcal{R}ec_{\mathcal{P}(x)} : \quad \mathcal{P}[0/x] \wedge \big(\forall y \quad \mathcal{P}[y/x] \to \mathcal{P}[S(y)/x]\big) \quad \to \quad \forall x \;\; \mathcal{P}(x)$$

En plus, si on a d'autres variables libres $x_1, \dots, x_n$, on note $\mathcal{P}(x, x_1, \dots, x_n)$ et on ajoute les quantificateurs universels au début :

$$\mathcal{R}ec_{\mathcal{P}} : \quad \forall x_1 \;\; \dots \;\; \forall x_n \;\; \Big[\mathcal{P}(0, x_1, \dots, x_n) \wedge \big[\forall x \quad \mathcal{P}(x, x_1, \dots, x_n) \to \mathcal{P}(S(x), x_1, \dots, x_n)\big] \to \forall x \;\; \mathcal{P}(x, x_1, \dots, x_n)\Big]$$

Cette version est indispensable pour prouver $x + y = y + x$ pour tout x, y en faisant une récurrence sur x.

2.3. Modèles de $\mathcal{PA}$

Proposition 1.
$\mathbb{N}$ est un modèle de $\mathcal{PA}$.

On considère la structure $\mathcal{S}$, qu'on note abusivement $\mathbb{N}$, de domaine $\mathbb{N}$, naturellement associée au langage $\mathcal{L}_0$ (0 est le zéro, $+$ est l'addition, $\times$ la multiplication et $S(n) = n+1$). On a vu dans les commentaires des axiomes que $\mathbb{N}$ vérifie les axiomes de $\mathcal{PA}$, c'est-à-dire $\vDash_{\mathbb{N}} \mathcal{A}_1, \dots, \vDash_{\mathbb{N}} \mathcal{A}_7$ et, quel que soit $\mathcal{P}$, $\vDash_{\mathbb{N}} \mathcal{R}ec_{\mathcal{P}(x)}$. Cela prouve la proposition.

Corollaire 1.
$\mathcal{PA}$ est cohérente.

On rappelle que cela signifie $\mathcal{PA} \nvdash \bot$, autrement dit, il n'existe aucune formule $\mathcal{Q}$ pour laquelle on a à la fois $\mathcal{PA} \vdash \mathcal{Q}$ et $\mathcal{PA} \vdash \neg\mathcal{Q}$. On sait qu'une théorie $\mathcal{T}$ est cohérente si et seulement si $\mathcal{T}$ admet un modèle (voir le chapitre « Théorème de complétude »). Comme $\mathbb{N}$ est un modèle de $\mathcal{PA}$, alors $\mathcal{PA}$ est cohérente.

Cependant, il existe d'autres modèles de $\mathcal{PA}$, appelés modèles ***non standards***. Ils sont tous infinis et contiennent $\mathbb{N}$. C'est un point très intéressant : à la fois notre but est d'axiomatiser $\mathbb{N}$, et effectivement on a réussi à en formaliser les propriétés essentielles ; mais en même temps on vient de dire que cela ne suffit pas à caractériser $\mathbb{N}$ de façon unique. C'est un peu paradoxal, mais ce sont ces idées qui font le cœur de ce livre. En fait, aucune axiomatisation de la logique du premier ordre ne permet de caractériser $\mathbb{N}$ de façon unique.

Avertissement. L'énoncé du premier théorème d'incomplétude de Gödel commence par l'hypothèse « Si $\mathcal{PA}$ est cohérente, … ». Pourquoi cette hypothèse, , puisqu'on vient de dire que c'est le cas ? En fait, si on se place uniquement dans le langage $\mathcal{L}_0$ de l'arithmétique de Peano, on ne peut pas prouver, à l'intérieur de ce langage, que $\mathcal{PA}$ est cohérente. C'est en fait exactement le second théorème de Gödel, qui dit qu'aucune théorie (suffisamment riche) ne peut prouver sa propre cohérence.
Ici, on admet que $\mathbb{N}$ existe. Ainsi, la théorie $\mathcal{PA}$ possède un modèle, donc elle est nécessairement cohérente. Cette certitude quant à la cohérence de $\mathcal{PA}$ repose sur une acceptation méta-mathématique de l'existence $\mathbb{N}$, c'est-à-dire qu'il faut se placer dans une théorie plus générale (par exemple $\mathcal{ZFC}$) pour confirmer cette construction.

3. Propriétés de l'arithmétique de Peano

On a défini un nombre volontairement restreint d'axiomes. Il s'agit maintenant de retrouver les propriétés arithmétiques usuelles à partir de ces axiomes. On va en prouver très peu car cela relèverait davantage d'un cours de mathématiques que d'un cours de logique. Surtout, c'est l'occasion idéale pour utiliser un logiciel de preuve formelle.

3.1. Avec les axiomes de base

Lemme 1.
$2+2=4$

Eh oui, rien n'est acquis ! On n'ose pas appeler cela un « théorème » et pourtant il faut bien justifier que $\mathcal{PA} \vdash$ « $2+2=4$ ».

Remarque. On note $1 = S(0)$, $2 = S(1)$, $3 = S(2)$, $4 = S(3)$. Ce sont juste des notations, $1, 2, 3, 4$ ne sont pas des symboles du langage mais des abréviations syntaxiques. Sans elles, il faudrait écrire $S(S(0)) + S(S(0)) = S(S(S(S(0))))$.

Démonstration.

$$\begin{aligned}
2+2 &= 2+S(1) && \text{par définition de } 2 = S(1)\\
&= S(2+1) && \text{par } \mathcal{A}_5\\
&= S\big(2+S(0)\big) && \text{par définition de } 1 = S(0)\\
&= S\big(S(2+0)\big) && \text{par } \mathcal{A}_5\\
&= S\big(S(2)\big) && \text{par } \mathcal{A}_4\\
&= S(3) && \text{par définition de } S(2) = 3\\
&= 4 && \text{par définition de } S(3) = 4
\end{aligned}$$

□

Sans la récurrence, aucune propriété algébrique générale n'est démontrable.

3.2. Avec la récurrence

Lemme 2.
$\forall x \quad 0 + x = x$

Cela ressemble à l'axiome $\mathcal{A}_4$: « $\forall x \;\; x+0 = x$ », mais on ne sait pas encore que l'addition est commutative.

Démonstration. Soit la formule $\mathcal{P}(x) : 0 + x = x$.
Nous écrivons la preuve par récurrence dans le style mathématique classique :

- Initialisation. $\mathcal{P}(0) : 0 + 0 = 0$. C'est vrai par $\mathcal{A}_4$ (c'est le 0 à droite qui s'applique, et pas celui à gauche !).
- Hérédité. Soit n fixé. On suppose $\mathcal{P}(n)$ vraie et on doit démontrer $\mathcal{P}(S(n))$ (l'équivalent de $\mathcal{P}(n+1)$).

$$\begin{aligned}
& \mathcal{P}(n) \text{ est vraie}\\
\text{donc} \quad & 0+n = n\\
\text{donc} \quad & S(0+n) = S(n) && \text{car } S \text{ est une fonction}\\
\text{donc} \quad & 0+S(n) = S(n) && \text{par l'axiome } \mathcal{A}_5\\
\text{donc} \quad & \mathcal{P}(S(n)) \text{ est vraie.}
\end{aligned}$$

On a ainsi $\mathcal{P}(n) \to \mathcal{P}(S(n))$ et si on revient à l'écriture logique dans $\mathcal{PA}$, on a prouvé $\mathcal{P}(x) \to \mathcal{P}(S(x))$ quel que soit x.

- Conclusion. Par le principe de récurrence, ici $\mathcal{R}ec_{\mathcal{P}(x)}$, on en déduit « $\forall x \;\; \mathcal{P}(x)$ », c'est-à-dire « $\forall x \;\; 0+x=x$ ».

□

Lemme 3.
$\forall x \;\; \forall y \quad S(x)+y=S(x+y)$
C'est comme l'axiome $\mathcal{A}_5$, mais interverti.

Démonstration. Fixons x. Nous allons faire une récurrence sur la variable y. Soit $\mathcal{P}(y) : S(x)+y=S(x+y)$.

- Initialisation. $\mathcal{P}(0) : S(x)+0=S(x)=S(x+0)$. C'est vérifié.
- Hérédité. Supposons $\mathcal{P}(y)$.

$$\begin{array}{lll} & S(x)+y=S(x+y) & \text{par hypothèse de récurrence} \\ \text{donc} & S(S(x)+y)=S(S(x+y)) & \text{car } S \text{ est une fonction} \\ \text{donc} & S(x)+S(y)=S(x+S(y)) & \text{par application de } \mathcal{A}_5 \text{ à gauche et à droite} \\ \text{donc} & \mathcal{P}(S(y)) \text{ est vraie.} & \end{array}$$

  Ainsi, quel que soit y, $\mathcal{P}(y) \to \mathcal{P}(S(y))$.
- Conclusion. Par le principe de récurrence, on a $\forall y \;\; \mathcal{P}(y)$. Comme c'est valable quel que soit x, on a bien $\forall x \;\forall y \;\; S(x)+y=S(x+y)$.

Remarque. Si on voulait être rigoureux, on devrait noter $\mathcal{P}(x,y) : S(x)+y=S(x+y)$ puis prouver $\forall x \;\; \mathcal{P}(x,0)$, puis $\forall x \;\forall y \;\; \mathcal{P}(x,y) \to \mathcal{P}(x,S(y))$, ce qui conduit à $\forall x \;\forall y \;\; \mathcal{P}(x,y)$. □

Lemme 4.
$\forall x \;\; \forall y \quad x+y=y+x$
C'est la commutativité attendue de l'addition.

Démonstration. Fixons x. Nous allons faire une récurrence sur y. Soit $\mathcal{P}(y) : x+y=y+x$.

- Initialisation. $\mathcal{P}(0) : x+0=x$ (par l'axiome $\mathcal{A}_4$) et $0+x=x$ (par le Lemme 2). Donc $x+0=0+x$ est vérifié.
- Hérédité. Supposons $\mathcal{P}(y)$ vraie.

$$\begin{array}{lll} & x+y=y+x & \\ \text{donc} & S(x+y)=S(y+x) & \\ \text{donc} & x+S(y)=S(y)+x & \text{par } \mathcal{A}_5 \text{ à gauche et le Lemme 3 à droite} \\ \text{donc} & \mathcal{P}(S(y)) \text{ est vraie.} & \end{array}$$

- Conclusion. Quel que soit x : $\forall y \;\; x+y=y+x$.

□

3.3. Autres opérations

- **Non-égalité.** $x \neq y$ signifie $\neg(x=y)$.
- **Inégalité.** $x \leqslant y$ signifie $\exists z \;\; x+z=y$ (on rappelle qu'on modélise les entiers positifs, il faut penser à z comme un élément de $\mathbb{N}$).
- **Inégalité stricte.** $x<y$ signifie $(x \leqslant y) \wedge (x \neq y)$.
- Soustraction. Attention, $x-y$ n'est pas une fonction car elle n'est pas toujours bien définie, par exemple $2-3$ n'a pas de sens ici. Par contre, pour $x \leqslant y$, on sait qu'il existe z tel que $x+z=y$, on pourrait définir $y-x=z$.
- Il n'y a pas non plus une division x/y bien définie.

Il y a maintenant du travail à volonté pour démontrer par exemple :

$$(x \leqslant y) \wedge (y \leqslant z) \quad \to \quad x \leqslant z$$
$$x \leqslant y \quad \to \quad x \times z \leqslant y \times z$$
$$\dots$$

3.4. Ensembles définissables

Fixons un langage $\mathcal{L}$ du premier ordre et une structure $\mathcal{S}$ associée à $\mathcal{L}$ de domaine E.

Définition.
Un ensemble $A \subset E$ est ***définissable*** s'il existe une formule $\mathcal{P}(x)$ de $\mathcal{L}$ telle que :

$$A = \{a \in E \mid \vDash_{\mathcal{S}} \mathcal{P}(a)\}$$

Autrement dit, A est exactement l'ensemble des a pour lesquels $\mathcal{P}(a)$ est vraie.

Exemple.

- L'ensemble des nombres pairs de $\mathbb{N}$ est définissable, défini par la formule :
$$\mathcal{P}(x) : \quad \exists y \quad x = y + y$$
- Dans $\mathbb{N}$, l'ensemble des nombres premiers est définissable par la formule :
$$\mathcal{P}(x) : \quad (x \neq 1) \wedge \Big[\forall y \quad \forall z \quad (x = y \times z) \quad \to \quad (y = 1) \vee (z = 1)\Big]$$
Lorsqu'on travaille formellement dans l'arithmétique de Peano (indépendamment du modèle $\mathbb{N}$), c'est cette formule qui sert de définition à la notion de nombre premier (avec l'abréviation $1 = S(0)$).

3.5. Récurrence forte

Dans une démonstration par récurrence classique, on suppose $\mathcal{P}(n)$ pour en déduire $\mathcal{P}(n+1)$. Dans une récurrence forte, on suppose $\mathcal{P}(0), \mathcal{P}(1), \dots, \mathcal{P}(n)$ vraies pour en déduire $\mathcal{P}(n+1)$ vraie. C'est parfois très utile dans certaines démonstrations.
Voici la formule de la récurrence forte pour $\mathcal{P}$:

$$\mathcal{R}ec\mathcal{F}_{\mathcal{P}(x)} : \quad \Big[\forall x \quad \big(\forall y \quad y < x \to \mathcal{P}(y)\big) \to \mathcal{P}(x)\Big] \quad \to \quad \forall x \quad \mathcal{P}(x)$$

Sur $\mathbb{N}$, c'est bien la traduction de l'idée que l'hérédité forte $[\mathcal{P}(0) \wedge \dots \wedge \mathcal{P}(n-1)] \to \mathcal{P}(n)$ entraîne $\forall n \;\; \mathcal{P}(n)$.
L'initialisation est implicite dans cette formule. En prenant $x = 0$ dans le crochet, le terme de gauche de l'implication, « $\forall y \quad y < 0 \to \mathcal{P}(y)$ », est alors toujours vrai (car aucun y n'est strictement inférieur à 0 dans $\mathbb{N}$) ; donc $\mathcal{P}(0)$ doit être vraie afin que l'implication du crochet soit vraie.
On montrerait en exercice que la récurrence forte est en fait équivalente à la récurrence classique.

4. Autres arithmétiques

4.1. Arithmétique de Presburger

C'est une arithmétique plus simple que celle de Peano, car sans la multiplication. En effet, le langage est $\mathcal{L} = (0, S, +)$ et on considère seulement les axiomes $\mathcal{A}_1, \ldots, \mathcal{A}_5$ et $\mathcal{R}ec$. On note $\mathcal{PR}$ cette théorie. Noter que, même si la multiplication est absente, on peut définir $3x$ comme $x + x + x$, en revanche, une formule « $x \times y$ » est interdite.

Théorème 1.
$\mathcal{PR}$ est cohérente et complète.

Rappelons ce que cela signifie. Pour n'importe quelle formule $\mathcal{Q}$ (fermée) :

- $\mathcal{PR}$ est cohérente, donc ne peut pas prouver à la fois $\mathcal{Q}$ et $\neg\mathcal{Q}$.
- $\mathcal{PR}$ est complète, par conséquent $\mathcal{PR}$ prouve $\mathcal{Q}$ ou bien $\mathcal{PR}$ prouve $\neg\mathcal{Q}$.

La preuve est assez technique, elle se base sur l'« élimination des quantificateurs ». Il s'agit de transformer une formule de $\mathcal{L}$ en une formule équivalente sans quantificateur, mais en introduisant une relation d'équivalence modulo des entiers n. On transforme d'abord les $\forall$ en $\exists$, car « $\forall x \quad \mathcal{P}(x)$ » équivaut à « $\neg(\exists x \quad \neg\mathcal{P}(x))$ ». Voyons comment éliminer les quantificateurs pour l'exemple de la formule « y est pair » :

$$\exists x \quad y = x + x$$

c'est-à-dire $\exists x \quad y = 2x$, ce qui équivaut à :

$$y \equiv 0 \pmod 2$$

Ainsi, à l'aide du modulo, la formule n'a plus de quantificateurs. De façon générale, partant d'une formule fermée, on obtient une expression algébrique modulo un entier n. La formule est prouvable si et seulement si cette expression algébrique est satisfaite. Ainsi, une formule est prouvable ou bien c'est sa négation qui est prouvable.

4.2. Arithmétique $\mathcal{Q}$ de Robinson

On va définir une nouvelle axiomatique, avec addition et multiplication mais sans le principe de récurrence. On note $\mathcal{Q}$ la théorie composée des axiomes $\mathcal{A}_1, \ldots, \mathcal{A}_7$ sans ajouter d'axiomes pour la récurrence. Cette fois l'axiome $\mathcal{A}_2$ qui affirme que tout élément non nul est un successeur est indispensable.

Théorème 2.
$\mathcal{Q}$ est cohérente mais non complète.

La théorie $\mathcal{Q}$ est cohérente car $\mathbb{N}$ en est un modèle.
Pour démontrer qu'elle n'est pas complète, nous allons montrer que la formule :

$$\mathcal{G} : \quad \forall x \quad 0 + x = x$$

n'est pas démontrable, et $\neg\mathcal{G}$ non plus. (La lettre G est pour « Gödel ».)
Il ne faut pas confondre $\mathcal{G}$ avec l'axiome $\mathcal{A}_4$ « $\forall x \quad x + 0 = x$ ». Souvenez-vous que pour démontrer $\mathcal{G}$ dans le cadre de l'arithmétique de Peano, on a utilisé la récurrence, ce qui est interdit ici.

Démonstration.

- $\neg\mathcal{G}$ n'est pas prouvable.
  Par l'absurde, si $\neg\mathcal{G}$ est prouvable, alors $\mathcal{Q} \vdash \neg\mathcal{G}$. Ainsi, par le théorème de validité, on a $\mathcal{Q} \vDash \neg\mathcal{G}$. Or $\mathbb{N}$ vérifie les axiomes de $\mathcal{Q}$, c'est-à-dire $\vDash_{\mathbb{N}} \mathcal{Q}$, et donc on doit avoir $\vDash_{\mathbb{N}} \neg\mathcal{G}$. Mais dans $\mathbb{N}$, $\mathcal{G}$ est vraie, donc $\neg\mathcal{G}$ est faux. On obtient une contradiction.

- $\mathcal{G}$ n'est pas prouvable.
  On va suivre la même idée que ci-dessus. Il s'agit de trouver un modèle de $\mathcal{Q}$ dans lequel $\mathcal{G}$ est faux. Bien sûr ce modèle ne peut pas être $\mathbb{N}$, puisque dans $\mathbb{N}$, $\mathcal{G}$ est vrai.
  Soit $\mathcal{S}$ la structure de domaine $\mathbb{N} \cup \{a, b\}$. a et b sont deux objets nouveaux et distincts, on peut les considérer comme deux infinis ∞_a et ∞_b.
  Pour deux entiers, l'addition est l'addition usuelle et on pose :
  — $a + 0 = a$
  — $0 + a = b$
  On a aussi $S(a) = a$ et $S(b) = b$ (dans l'esprit « $\infty + 1 = \infty$ »), on n'a pas besoin de définir le détail des autres opérations pour la suite.
  On peut vérifier que tous les axiomes $\mathcal{A}_1, \ldots, \mathcal{A}_7$ sont satisfaits (voir les exercices). Par contre $\mathcal{G}$ est fausse, car $0 + a = b$ et $a \neq b$.
  On conclut comme précédemment. Par l'absurde, si $\mathcal{Q} \vdash \mathcal{G}$ alors $\mathcal{Q} \vDash \mathcal{G}$, donc $\vDash_{\mathcal{S}} \mathcal{G}$, donc $\mathcal{G}$ est vraie dans $\mathcal{S}$, ce qui contredit notre construction.

□

Mais alors est-ce la fin de notre histoire ? L'objectif de Gödel et de ce livre est d'obtenir un théorème d'incomplétude et on vient de trouver une théorie incomplète. À quoi bon continuer ? La raison est la suivante : si on n'est pas content que $\mathcal{Q}$ soit incomplète car on ne sait pas démontrer $\mathcal{G}$ (ni $\neg\mathcal{G}$), on pourrait tout simplement augmenter la théorie $\mathcal{Q}$ en une théorie $\mathcal{Q}_1 = \mathcal{Q} \cup \{\mathcal{G}\}$ qui, elle, bien sûr, prouve $\mathcal{G}$. Le problème est que $\mathcal{Q}_1$ non plus n'est pas complète. Il existe $\mathcal{G}_1$ tel que ni $\mathcal{Q}_1 \vdash \mathcal{G}_1$, ni $\mathcal{Q}_1 \vdash \neg\mathcal{G}_1$ (voir les exercices). On pourrait alors considérer $\mathcal{Q}_2 = \mathcal{Q}_1 \cup \{\mathcal{G}_1\}$ … ce qui ne fait que repousser le problème. On prend alors conscience qu'il faut un énoncé général afin de justifier qu'on n'obtiendra jamais une théorie complète. C'est cela la puissance du théorème d'incomplétude de Gödel : quelle que soit la théorie, elle sera incomplète, donc ce n'est pas en ajoutant une formule non démontrable qu'on obtiendra une théorie complète.

4.3. Axiomatique $\mathcal{ZF}$ et $\mathcal{ZFC}$

La théorie de Zermelo-Fraenkel, notée $\mathcal{ZF}$, est une théorie axiomatique des ensembles qui permet, entre autres, de fonder l'arithmétique. Le langage est $\mathcal{L} = \{\in\}$ (avec égalité) et tout objet est un ensemble. Voir le chapitre « Variables et structures » pour une introduction. Ainsi, « $x \in y$ » signifie « l'ensemble x appartient à l'ensemble y ». Par exemple $\{0, 1\} \in \big\{\varnothing, \{0\}, \{1\}, \{0, 1\}\big\}$.
C'est une théorie plus moderne et beaucoup plus puissante que l'arithmétique de Peano. Cependant, nous nous en passerons pour les théorèmes d'incomplétude. Ainsi, la mini-introduction suivante à la théorie $\mathcal{ZF}$ ne sera pas utile pour la suite des chapitres.
On a déjà vu comment construire les entiers en partant de l'ensemble vide :

$$0 = \varnothing, \quad 1 = \{\varnothing\}, \quad 2 = \{\varnothing, \{\varnothing\}\} = \{0, 1\}, \quad 3 = \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\} = \{0, 1, 2\}, \quad \ldots$$

et de façon plus générale $S(n) = n \cup \{n\}$.
$\mathcal{ZF}$ est régie par une liste d'axiomes permettant de manipuler les ensembles. On va en donner la liste mais pas tous les détails :

- **Axiome de l'ensemble vide** : il existe un ensemble ne contenant aucun élément.
- **Axiome de l'infini** : il garantit l'existence d'un ensemble infini (contenant tous les entiers naturels).
- **Axiome d'extensionalité** : deux ensembles ayant les mêmes éléments sont égaux.

$$\forall x \ \ \forall y \quad \big(\forall z \ (z \in x \ \leftrightarrow \ z \in y) \ \rightarrow \ x = y\big)$$

- **Axiome de fondation** : tout ensemble non vide x possède un élément y qui est disjoint de x. Cela interdit l'existence de cycles d'appartenance (comme $x \in x$).

$$\forall x \neq \varnothing \quad \exists y \in x \quad (y \cap x = \varnothing)$$

- **Axiome de la réunion** : pour un ensemble x, il existe un ensemble $\bigcup_x$ qui est la réunion de tous les ensembles appartenant à x.
$$\forall x \ \exists y \ \forall z \quad \big(z \in y \iff \exists t \ (t \in x) \wedge (z \in t)\big)$$
Par exemple, si $x = \big\{\{1,2\},\{2,3\},\{4,5\}\big\}$, alors $\bigcup_x = \{1,2,3,4,5\}$ (1, 2, ... sont considérés comme des ensembles).
- **Axiome de la paire** : à partir de deux ensembles x et y, il existe un ensemble $\{x,y\}$ exactement formé de x et de y.
Exemple : $x = \big\{\{1,2\},\{3\}\big\}$ et $y = \{3,4\}$, alors $\{x,y\} = \Big\{\big\{\{1,2\},\{3\}\big\},\{3,4\}\Big\}$. (Ce n'est pas l'union de x et y.)
- **Axiome des parties** : pour un ensemble x, il existe un ensemble $\mathcal{P}(x)$ qui est l'ensemble des parties de x. Par exemple, si $x = \{0,1\}$, alors $\mathcal{P}(x) = \big\{\varnothing,\{0\},\{1\},\{0,1\}\big\}$.
- **Schéma d'axiomes de compréhension** : ils permettent de définir des sous-ensembles à l'aide d'une formule $\mathcal{P}$: ainsi $\{x \in y \mid \mathcal{P}(x)\}$ est un ensemble.
- **Schéma d'axiomes de remplacement** : il permet de remplacer les éléments d'un ensemble par d'autres objets, le résultat reste un ensemble.

Ces axiomes forment la théorie $\mathcal{ZF}$.

On considère souvent en plus l'axiome suivant :

- **Axiome du choix** $\mathcal{AC}$: pour tout ensemble A, il existe une fonction qui à chaque partie non vide de A, associe un élément de cette partie.

Cet axiome est indépendant de $\mathcal{ZF}$ (on ne peut ni le démontrer, ni démontrer sa négation à partir de $\mathcal{ZF}$) ce qui fait qu'on peut l'ajouter ou non selon ses besoins quand on est confronté à des ensembles infinis. Il se retrouve sous des formes différentes (lemme de Zorn, théorème de Krull), il est indispensable pour certaines démonstrations classiques (existence de bases dans des espaces vectoriels de dimension infinie, existence de parties de $\mathbb{R}$ non mesurables au sens de Lebesgue) et peut aussi simplifier certaines démonstrations (théorème de Hahn-Banach dans le cas dénombrable). Si on décide d'adopter l'axiome du choix, on note la théorie $\mathcal{ZFC}$. C'est le cadre classique des mathématiques modernes.

Chapitre 14

Codage de Gödel

On transforme les symboles, les formules et même les preuves en des entiers. On expliquera ensuite les grandes lignes de la preuve du premier théorème d'incomplétude.

1. Codage des symboles, formules, preuves

1.1. Motivation

Le but est de coder chaque formule par un entier, afin de permettre un traitement mathématique et algorithmique. Une analogie parlante se fait avec le code ASCII qui code un caractère par un entier ('A' : 65, 'B' : 66, ...). Pour coder un texte (une chaine de caractères), il suffit de juxtaposer les entiers ('BAC' : 66/65/67). Comme un ordinateur ne code pas les entiers par les chiffres de 0 à 9 mais seulement avec les symboles 0 et 1, nos caractères ASCII sont écrits sur 8 bits :

A : 01000001
B : 01000010
Z : 01011010

Pour une chaine de caractères il suffit de juxtaposer les codes de chaque caractère en un seul entier.

'BAC' : 010000100100000101000011

Le décodage est facile et commence par séparer le code par paquets de 8 bits.

Il y a une distinction importante à faire entre un nombre et la représentation de ce nombre :

$$7, \quad \text{VII}, \quad \text{sept}, \quad 111_2, \quad \text{SSSSSSS}(0)$$

sont différentes représentations du même entier désigné par '7' (ici en chiffres arabes, romains, en toutes lettres, en écriture binaire et dans le langage $\mathcal{L}_0$ de l'arithmétique de Peano). C'est une distinction bien connue en informatique. Il y a un aspect philosophique profond à donner un nom aux choses, même si le nom ne change pas la chose elle-même. Ainsi, tout comme le mot *table* n'a pas quatre pieds, les symboles arithmétiques servent uniquement de support formel pour représenter les entiers.
Dans la suite, ce qui est vraiment fondamental c'est l'existence d'un codage et d'un décodage. Les rouages internes ne sont pas si importants et varient d'un livre à l'autre. De toute façon, on verra que ce codage est absolument inutilisable en pratique.

1.2. Coder les symboles

On considère le langage $\mathcal{L}_0 = (0, S, +, \times)$ de la logique du premier ordre pour l'arithmétique de Peano. Il faut coder les symboles du langage, les connecteurs, les quantificateurs et les variables $(x_i)_{i\geqslant 1}$.
On choisit (arbitrairement) le codage suivant (voir le livre de Peter Smith) :

$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\forall$	$\exists$	$=$	$($	$)$	0	S	$+$	$\times$
1	3	5	7	9	11	13	15	17	19	21	23	25	27

x_1/x	x_2/y	x_3/z	x_4	$\cdots$	x_i
2	4	6	8	$\cdots$	$2i$

Par exemple, le code du symbole « $\exists$ » est $c = 13$. Le symbole du code $c = 21$ est « 0 » (la constante zéro du langage). La variable x_i est codée par l'entier $2i$.
Pour décoder un code, il suffit de lire le tableau de bas en haut :

- si c est impair, chercher à quel symbole il correspond,
- si c est pair, le symbole est la variable $x_{c/2}$.

Remarques :

- Pour certaines preuves on peut limiter le nombre de connecteurs, par exemple à $(\neg, \rightarrow, \forall)$ ou bien à $(\neg, \vee, \forall)$.
- Il est important pour le décodage de ne pas avoir de code qui vaut 0.

1.3. Coder une formule

On considère un mot du langage $\mathcal{L}_0$, c'est donc une suite finie de symboles. Cela peut être une formule « $\forall S(x) = y$ » ou une suite de symboles qui n'a pas de sens « $S(\forall) \rightarrow \neg \times 0$ ».
On va noter $c_1, c_2, c_3, \ldots$ les codes des symboles de ce mot (lu de gauche à droite). Ces codes seront les exposants dans un produit de nombres premiers :

$$2^{c_1} \times 3^{c_2} \times 5^{c_3} \times \cdots$$

On note $\pi_1 = 2$, $\pi_2 = 3$, $\pi_3 = 5, \ldots$ la suite ordonnée des nombres premiers. On peut ainsi formaliser notre construction :

- On part d'un mot (ou d'une formule) $\mathcal{P}$ qui est une suite de symboles $a_1, a_2, \ldots, a_l$ de $\mathcal{L}_0$.
- À chaque symbole a_i, on associe son code c_i.
- Le ***nombre de Gödel*** $n(\mathcal{P})$ de la formule $\mathcal{P}$ est :

$$\boxed{n = \prod_{i=1}^{l} \pi_i^{c_i}}$$

Remarques. C'est généralement un très grand nombre. Cela s'applique aussi à un mot quelconque de $\mathcal{L}_0$, même si ce n'est pas une formule.

> **Exemple.**
> - $\mathcal{P}$: « $\neg(x = 0)$ »
> — $[a_1, \ldots, a_6] = [\neg, (, x, =, 0,)]$
> — $[c_1, \ldots, c_6] = [1, 17, 2, 15, 21, 19]$
> — $n(\mathcal{P}) = 2^1 \cdot 3^{17} \cdot 5^2 \cdot 7^{15} \cdot 11^{21} \cdot 13^{19}$
> - « 2 » n'existe pas dans $\mathcal{L}_0$, son écriture dans $\mathcal{L}_0$ est « $SS0$ » dont le nombre de Gödel est $2^{23} \cdot 3^{23} \cdot 5^{21}$.

1.4. Décoder

Le décodage est basé sur la décomposition en facteurs premiers des entiers (aussi appelé le principe fondamental de l'arithmétique) et en particulier sur l'unicité de cette décomposition.

Théorème 1.
Pour tout entier $n \geqslant 1$, il existe un entier $l \geqslant 1$ et des entiers $\alpha_1, \ldots, \alpha_l \geqslant 0$ tels que :

$$n = \pi_1^{\alpha_1} \times \pi_2^{\alpha_2} \times \cdots \times \pi_l^{\alpha_l}$$

De plus, les α_i sont uniques.

Ainsi, pour décoder un nombre de Gödel n, c'est-à-dire retrouver la formule encodée, on commence par décomposer n en produit de facteurs premiers. Les exposants $c_1, \ldots, c_l$ sont les codes qui redonnent un à un les symboles de la formule.
Il n'y a qu'une seule façon de décoder un nombre. D'un point de vue mathématique, cela signifie que le codage est injectif :

$$\text{Si } \; n(\mathcal{P}) = n(\mathcal{Q}) \; \text{ alors } \; \mathcal{P} = \mathcal{Q}.$$

> **Exemple.**
> $n = 35\,872\,267\,500$
> - Décomposition en facteurs premiers : $n = 2^2 \times 3^{15} \times 5^4$.
> - Les exposants donnent les codes $[2, 15, 4]$.
> - Chaque code se décode en un symbole $[x, =, y]$.
> - La formule de n est donc « $x = y$ ».

1.5. Coder les preuves

On souhaite aussi coder une preuve entière par un seul entier. Une preuve est ici une suite finie de formules $(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_L)$. Comment coder une preuve ? Chaque $\mathcal{P}_j$ est codée par son nombre de Gödel $n_j = n(\mathcal{P}_j)$ Ensuite, on associe à la preuve son ***super-nombre de Gödel*** :

$$N = 2^{n_1} \cdot 3^{n_2} \cdots \pi_L^{n_L}$$

c'est-à-dire :

$$\boxed{N = \prod_{j=1}^{L} \pi_j^{n(\mathcal{P}_j)}}$$

Remarques :
- C'est un nombre très très grand. On ne le calculera jamais en pratique.
- On ne mélangera jamais les nombres de Gödel et les super-nombres de Gödel. Ce sont deux codages distincts. On saura au préalable si un entier donné désigne un nombre de Gödel ou un super-nombre de Gödel.

Comment décoder un super-nombre de Gödel ?
- On décompose N en produit de facteurs premiers.
- On en extrait la liste des exposants $(n_1, \ldots, n_L)$.
- On décode chaque nombre de Gödel n_j, en une formule $\mathcal{P}_j$.
- On obtient la liste de formules $(\mathcal{P}_1, \ldots, \mathcal{P}_L)$.

2. Algorithme de décodage

Nous avons réussi à coder les symboles, les formules, les preuves par des entiers. Il va falloir montrer que les opérations associées sur les entiers sont « suffisamment simples », ce qui correspondra à la notion de *fonction récursive primitive* discutée dans les chapitres suivants. Pour l'instant on va aborder cette problématique d'un point de vue algorithmique, afin de mieux appréhender le travail futur.
On commence par les opérations de codage/décodage. Que ces opérations soient « suffisamment simples » signifie qu'on peut les programmer en utilisant uniquement des boucles « pour » (*for*) bornées (le nombre d'itérations est borné en fonction de l'entrée). Cela signifie qu'on s'interdit les boucles « tant que » (*while*) pour lesquelles on ne sait pas à l'avance quand sera atteinte la condition d'arrêt.

2.1. Décoder le code d'un symbole

Il est facile d'écrire un algorithme `decode_code_symbole(c)` qui en entrée reçoit un entier c (le code valide d'un symbole) et qui en sortie renvoie le symbole $a \in \mathcal{L}_0$ correspondant (voir la section précédente). Il n'y a même pas besoin d'une boucle « pour ».

Algorithme (`decode_code_symbole(c)`).
- — Entrée : c un entier.
  — Sortie : a un symbole de $\mathcal{L}_0$.
- Si $c = 1$, $a \leftarrow$ « $\neg$ ».
- Si $c = 3$, $a \leftarrow$ « $\wedge$ ».
- ...
- Si $c = 27$, $a \leftarrow$ « $\times$ ».
- Si c est pair, $a \leftarrow x_{c/2}$.
- Renvoyer a.

2.2. Décoder un nombre de Gödel

Lemme 1.
Soit $n = \pi_1^{\alpha_1} \times \cdots \times \pi_l^{\alpha_l}$ *avec* $\alpha_i \geqslant 1$ $(i = 1, \ldots, l)$.
- $l \leqslant \log_2(n) \leqslant n$
- $\alpha_i \leqslant \log_2(n) \leqslant n \quad (i = 1, \ldots, l)$

Démonstration. On rappelle que par définition $\log_2(2^n) = n$. Comme $2^n \geqslant n$ alors $n \geqslant \log_2(n)$.
- $n = \pi_1^{\alpha_1} \cdots \pi_l^{\alpha_l} \geqslant \pi_1 \times \cdots \times \pi_l \geqslant 2^l$ donc $\log_2(n) \geqslant l$.
- $n = \pi_1^{\alpha_1} \cdots \pi_l^{\alpha_l} \geqslant \pi_i^{\alpha_i} \geqslant 2^{\alpha_i}$ donc $\log_2(n) \geqslant \alpha_i$.

□

La boucle de l'algorithme suivant est bornée par $K = \lfloor \log_2(n) \rfloor$. Le lecteur pointilleux qui considère que calculer $\log_2(n)$ est une opération compliquée remplacera $\log_2(n)$ par n.

Algorithme (`decode_nombre_godel(n)`).
- — Entrée : n un nombre de Gödel.
  — Sortie : un mot/formule $[a_1, \ldots, a_l]$.
- `liste` $\leftarrow$ `[ ]` (liste vide).
- Pour k allant de 1 à $K = \lfloor \log_2(n) \rfloor$:
  $c \leftarrow$ `exposant`(n, π_k)

Si $c > 0$, ajouter `decode_code_symbole(c)` à `liste`.
- Renvoyer `liste`.

On laisse au lecteur le soin d'écrire l'algorithme de décodage d'un super-nombre de Gödel N.

2.3. Nombres premiers

On s'autorise les opérations arithmétiques de base $+$, $-$, $\times$, $/$. Cependant, il faut s'assurer que les opérations sur les nombres premiers sont faciles à réaliser.
Tout d'abord, le test de divisibilité `est_divisible(n,d)` qui décide si d divise n serait facile à programmer avec une boucle « tant que ». Voici l'algorithme avec une boucle « pour ».

Algorithme (`est_divisible(n,d)`).
- — Entrée : deux entiers n, d.
  — Sortie : Vrai ssi d divise n.
- Pour i allant de 1 à n :
  $n \leftarrow n - d$
  Si $n = 0$, renvoyer Vrai (et terminer la boucle).
  Si $n < 0$, renvoyer Faux (et terminer la boucle).

On en déduit un algorithme `est_premier(n)` qui renvoie Vrai lorsque n est un nombre premier et Faux sinon, en testant s'il existe un diviseur d parmi $2, 3, \ldots, n-1$.
Ensuite, on sait récupérer l'exposant de p dans la décomposition d'un entier n en facteurs premiers. Par exemple `exposant`$(2^a 3^b 5^c, 3)$ est b.

Algorithme (`exposant(n,p)`).
- — Entrée : n un entier, p un nombre premier.
  — Sortie : l'exposant e de p dans n.
- $e \leftarrow 0$.
- Pour k allant de 1 à $K = \lfloor \log_2(n) \rfloor$:
  Si `est_divisible(n, p)` :
  $e \leftarrow e + 1$
  $n \leftarrow n/p$
  Sinon quitter la boucle.
- Renvoyer e.

Enfin, on a partout utilisé la liste $\pi_1 = 2, \pi_2 = 3, \ldots$ de tous les nombres premiers. Il faut vérifier qu'on sait calculer cette liste aussi loin que nécessaire. On programme une fonction `prochain_premier(n)` qui renvoie le premier nombre premier plus grand que n. Le postulat de Bertrand (prouvé par Tchebychev) affirme qu'il existe toujours un nombre premier entre n et $2n$. Il suffit donc de tester la primalité de $n+1, \ldots, 2n$. On obtient alors facilement une fonction `liste_premiers(K)` qui renvoie la liste de tous les nombres premiers inférieurs à K.

3. Opérations sur les formules

Nous allons maintenant justifier qu'à partir d'un nombre de Gödel n, on peut « facilement » dire si la formule correspondante vérifie une propriété. Par exemple, la formule est-elle bien définie ? Est-elle une formule fermée ? Est-elle réduite à une variable ? Et pour un super-nombre de Gödel N : encode-t-il une preuve correcte ?

3.1. Formules

- `est_variable(n)` teste si la formule de nombre de Gödel n est une des variables x_i.
- `est_terme(n)` teste si la formule de nombre de Gödel n est un terme de $\mathcal{L}_0$. On rappelle que les termes sont la constante 0, les variables x_i et si t_1 et t_2 sont des termes alors $S(t_1)$, $t_1 + t_2$, $t_1 \times t_2$ sont aussi des termes.
- `est_formule(n)` renvoie Vrai lorsque n encode une formule valide du langage $\mathcal{L}_0$ (« $x = \forall$ » n'est pas valide par exemple).
- `est_formule_fermee(n)` renvoie Vrai si n encode une formule sans variable libre.

Voici par exemple l'algorithme pour tester si n encode un terme de $\mathcal{L}_0$.
Algorithme `est_terme(n)` :

- — Entrée : un entier n.
  — Sortie : Vrai ssi n encode un terme de $\mathcal{L}_0$.
- $\mathcal{P} \leftarrow$ `decode_nombre_godel(n)`
- Si $\mathcal{P} =$ « 0 », renvoyer Vrai
- Si $\mathcal{P}$ est une variable, renvoyer Vrai
- Si $\mathcal{P}$ s'écrit « $S(t)$ », renvoyer `est_terme(t)`
- Si $\mathcal{P}$ s'écrit « $t_1 + t_2$ », renvoyer `est_terme(t_1)` ET `est_terme(t_2)`
- Si $\mathcal{P}$ s'écrit « $t_1 \times t_2$ », renvoyer `est_terme(t_1)` ET `est_terme(t_2)`
- Sinon, renvoyer Faux

C'est une fonction récursive, dont la profondeur est inférieure à la longueur de $\mathcal{P}$, donc inférieure à $\log_2(n)$. On pourrait en écrire un algorithme avec des boucles en utilisant des piles.

3.2. Preuve

Il est fondamental qu'on sache décider efficacement si une preuve donnée est valide ou pas. C'est le rôle de la fonction `est_preuve(N,n)`.

- Soit N super-nombre de Gödel représentant une suite de formules $\mathcal{P}_1, \ldots, \mathcal{P}_L$.
- Soit n un nombre de Gödel représentant une formule $\mathcal{Q}$.
- La fonction `est_preuve(N,n)` renvoie Vrai ssi $\mathcal{P}_L = \mathcal{Q}$ et que $\mathcal{P}_1, \ldots, \mathcal{P}_L$ forment une preuve (une suite de formules obtenues par les règles d'inférence).

$$\begin{array}{cl} & \mathcal{P}_1 \\ & \vdots \\ & \mathcal{P}_{L-1} \\ \vdash & \mathcal{P}_L = \mathcal{Q} \end{array}$$

Ici il est plus facile de supposer que les preuves sont écrites dans le système de Hilbert, dans lequel on a des axiomes et des règles d'inférence de la forme $\mathcal{P}_a \to \mathcal{P}_b$. Il n'y a pas d'hypothèses, ni sous-hypothèses, comme dans nos preuves en déduction naturelle. Pour obtenir $\mathcal{P} \vdash \mathcal{Q}$ on prouve en fait $\mathcal{P} \to \mathcal{Q}$.
Ainsi, on vérifie facilement :

- soit $\mathcal{P}_i$ est un axiome,
- soit $\mathcal{P}_i$ se déduit de $\mathcal{P}_a$ et $\mathcal{P}_b$ avec $a, b < i$ où $\mathcal{P}_a$ est de la forme $\mathcal{P}_b \to \mathcal{P}_i$.
- Si $\mathcal{P}_1, \dots, \mathcal{P}_L$ forment une preuve valide et que $\mathcal{P}_L = \mathcal{Q}$ alors on renvoie Vrai.

3.3. Valeur entière vs numéral

On revient sur la différence entre un nombre et l'écriture de ce nombre, avec deux nouvelles notations :
- $\ulcorner \mathcal{P} \urcorner$: le nombre de Gödel de $\mathcal{P}$,
- le ***numéral*** $\overline{n}$: c'est l'écriture d'un entier n dans le langage $\mathcal{L}_0$.

$\mathcal{P}$	$\longleftrightarrow$	$n = \ulcorner \mathcal{P} \urcorner$	$\longleftrightarrow$	$\overline{n} = \overline{\ulcorner \mathcal{P} \urcorner}$
formule de $\mathcal{L}_0$		nombre de Gödel, dans $\mathbb{N}$		numéral de n, dans $\mathcal{L}_0$

Exemple :

$$\mathcal{P} : x = 0 \quad \longleftrightarrow \quad n = \ulcorner \mathcal{P} \urcorner = 2^{\cdots} \cdot 3^{\cdots} \cdot 5^{\cdots} \quad \longleftrightarrow \quad \overline{n} = \overline{\ulcorner \mathcal{P} \urcorner} = \underbrace{SS \dots S}_{n \text{ occurrences}} 0$$

On écrira souvent $\mathcal{Q}(n)$, au lieu de $\mathcal{Q}(\overline{n})$ car il ne peut pas y avoir de confusion. Ainsi, lorsque l'on écrit « `est_formule(n)` », si l'on veut vraiment un algorithme dans le langage $\mathcal{L}_0$, n doit être écrit sous la forme du numéral $\overline{n} = SS \dots S0$.

3.4. Concaténation

Deux formules (ou mots) $\mathcal{P}$ et $\mathcal{Q}$ juxtaposées donnent la ***concaténation*** $\mathcal{P} * \mathcal{Q}$. Par exemple « $S(x)$ » $*$ « $+$ » $*$ « y » donne « $S(x) + y$ ». Cette opération semble immédiate, cependant elle est plus délicate à écrire lorsque nos formules sont représentées par des entiers :
- $\ulcorner S(x) \urcorner = 2^a \cdot 3^b \cdot 5^c \cdot 7^d$
- $\ulcorner + \urcorner = 2^e$
- $\ulcorner y \urcorner = 2^f$
- $\ulcorner S(x) + y \urcorner = 2^a \cdot 3^b \cdot 5^c \cdot 7^d \cdot 11^e \cdot 13^f$

Formellement, si $n = \prod_{i=1}^{l} \pi_i^{\alpha_i}$ et $m = \prod_{j=1}^{l'} \pi_j^{\beta_j}$ alors :

$$n * m = n \times \prod_{j=1}^{l'} \pi_{j+l}^{\beta_j}$$

3.5. Substitution

Considérons une formule $\mathcal{P}$ de variable libre x, que l'on note généralement $\mathcal{P}(x)$. Quand on écrit $\mathcal{P}(n)$ c'est en fait la substitution $\mathcal{P}[n/x]$. Il faut maintenant écrire cela dans le langage $\mathcal{L}_0$, c'est-à-dire qu'il nous faut une fonction `substitution`$(\ulcorner \mathcal{P} \urcorner, \ulcorner x \urcorner, n)$ qui renvoie $\overline{\ulcorner \mathcal{P}(\overline{n}) \urcorner}$, le nombre de Gödel de la substitution.

> **Exemple.**
> - $\mathcal{P}(x)$: « $y = x + 0$ », de nombre de Gödel $\ulcorner \mathcal{P}(x) \urcorner = 2^{\cdots} \cdot 3^{\cdots} \cdot 5^{\cdots} \cdot 7^{\cdots} \cdot 11^{\cdots}$.
> - $\mathcal{P}(2)$: « $y = 2 + 0$ », mais « 2 » n'est pas un symbole du langage, il faut écrire $\overline{2} = SS0$.
> - Ainsi, $\mathcal{P}(\overline{2})$: « $y = SS0 + 0$ », c'est bien une formule de $\mathcal{L}_0$.
> - $\ulcorner \mathcal{P}(\overline{2}) \urcorner = 2^{\cdots} \cdot 3^{\cdots} \cdot 5^{\cdots} \cdot 7^{\cdots} \cdot 11^{\cdots} \cdot 13^{\cdots} \cdot 17^{\cdots}$ est le nombre de Gödel de la substitution.
> - Il ne reste plus qu'à transformer $\ulcorner \mathcal{P}(\overline{2}) \urcorner$ en son numéral $\overline{\ulcorner \mathcal{P}(\overline{2}) \urcorner}$.

C'est un bon exercice, pas trop difficile, d'écrire l'algorithme de substitution en utilisant la concaténation.

4. Preuve du théorème d'incomplétude par diagonalisation

4.1. Le paradoxe de Russell

Nous allons donner l'idée générale de la preuve du premier théorème d'incomplétude. La preuve repose sur le principe de diagonalisation qui prend ici la forme d'auto-référencement, c'est-à-dire une formule qui parle d'elle-même. On commence par une petite analogie avec le paradoxe de Russell, popularisé sous la forme de l'énigme suivante :

Dans un village, le barbier rase tous les hommes qui ne se rasent pas eux-mêmes, et seulement ceux-là. Qui rase le barbier ?

Si on suppose que le barbier ne se rase pas lui-même, alors le barbier se rase lui-même. Et s'il se rase lui-même, il ne doit pas se raser lui-même. C'est inextricable !

La version suivante a servi à Russell pour justifier qu'il ne peut pas exister un ensemble contenant tous les ensembles. Si un tel ensemble $\mathcal{E}$ existait, on pourrait considérer :

$$y = \{x \in \mathcal{E} \mid x \notin x\}$$

Question : Est-ce que $y \in y$?
Remarque : y est bien un ensemble de $\mathcal{E}$, car $\mathcal{E}$ contient tous les ensembles.

- Si $y \in y$ alors par définition de y, $y \notin y$.
- Si $y \notin y$ alors par définition de y, $y \in y$.

On obtient une contradiction, aussi l'ensemble de tous les ensembles ne peut pas exister.

4.2. La diagonalisation et la formule de Gödel

Formule de départ.
Voici la formule que l'on va étudier :

> $\mathcal{P}(x)$: « La formule obtenue à partir de la formule de nombre de Gödel x, en y substituant sa variable libre par x, est non démontrable. »

Bien sûr, dans les chapitres suivants on écrira cette formule de façon purement mathématique.
Pour l'instant, remarquons :

- Pour un entier n, on sait qu'il peut être vu comme le nombre de Gödel d'une formule $\mathcal{Q}$ que l'on saurait retrouver (voir les sections précédentes).
- Si cette formule $\mathcal{Q}$ possède une variable libre, par exemple $\mathcal{Q}(y)$, alors on peut effectuer la substitution $\mathcal{Q}[n/y]$, notée $\mathcal{Q}(n)$.
- Enfin, pour un n donné on peut discuter si la formule fermée $\mathcal{Q}(n)$ est démontrable ou pas.

Il faut maintenant justifier que cette formule $\mathcal{P}(x)$ est bien une formule de notre langage $\mathcal{L}_0$. On a vu que « `est_formule()` » et « `est_preuve()` » sont des algorithmes faciles, ce qui signifie qu'ils pourraient s'exprimer dans le langage $\mathcal{L}_0$ de l'arithmétique de Peano. Mais qu'en est-il de l'affirmation « cette formule est démontrable » ?
On peut définir :

$$\texttt{est\_demontrable(n)} \quad \text{ssi} \quad \exists N\ \texttt{est\_preuve(N, n)}.$$

Le problème principal, d'un point de vue algorithmique, est qu'on n'a pas de borne à priori sur le N, donc un algorithme correspondant utiliserait une boucle « tant que » afin de tester $N = 1$, $N = 2$, ...
Si la formule est démontrable, l'algorithme finit par s'arrêter, mais ce n'est pas le cas si elle est non démontrable. C'est donc un mauvais algorithme de notre point de vue.

En revanche ; « $\exists N$ `est_preuve(N, n)` » est bien une formule de notre langage du premier ordre. Ainsi, « cette formule n'est pas démontrable » est bien une formule de $\mathcal{L}_0$. Ce qui fait de $\mathcal{P}(x)$, avec x comme variable libre, une formule de $\mathcal{L}_0$.

Diagonalisation.
Comme $\mathcal{P}(x)$ est une formule de $\mathcal{L}_0$, elle admet un nombre de Gödel. Notons :

$$\boxed{n = \ulcorner \mathcal{P}(x) \urcorner}$$

le nombre de Gödel de la formule $\mathcal{P}$.
La formule de Gödel est alors définie ainsi :

$$\boxed{\mathcal{G} = \mathcal{P}(n)}$$

C'est donc la formule $\mathcal{P}$ évaluée en son propre nombre de Gödel, autrement dit :

$$\boxed{\mathcal{G} = \mathcal{P}(\ulcorner \mathcal{P} \urcorner)}$$

Le théorème d'incomplétude.
Avec cette formulation encore informelle, voici le théorème d'incomplétude et sa preuve.

Théorème 2.
Il existe des formules vraies et non démontrables.

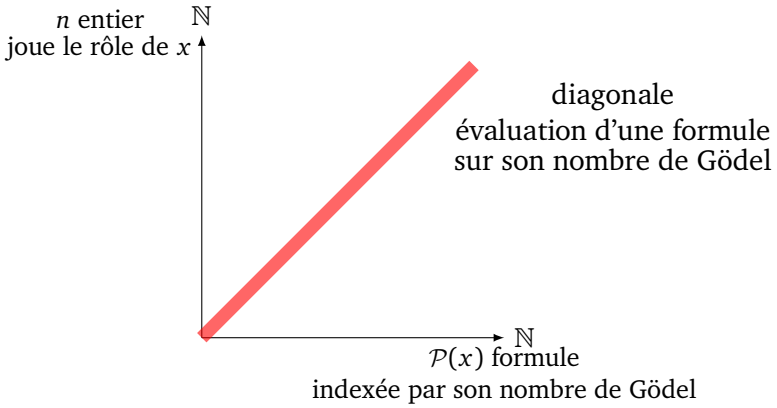
Nous allons montrer que $\mathcal{G}$ est une telle formule. Tout d'abord, $\mathcal{G}$ dit « la formule obtenue à partir de la formule de nombre de Gödel n, en y substituant sa variable libre par n, est non démontrable ». Mais la partie « la formule obtenue à partir de la formule de nombre de Gödel n, en y substituant sa variable libre par n » désigne exactement la formule $\mathcal{P}$ évaluée en n, c'est-à-dire $\mathcal{G} = \mathcal{P}(n)$, car n est le nombre de Gödel de $\mathcal{P}$. Ainsi, la phrase $\mathcal{G}$ dit exactement « $\mathcal{G}$ n'est pas démontrable ».

Démonstration.
- Par l'absurde, supposons $\mathcal{G}$ fausse. Alors $\neg\mathcal{G}$ est vraie, ce qui signifie « $\mathcal{G}$ est démontrable ». Mais par le théorème de validité : si $\mathcal{G}$ est démontrable alors $\mathcal{G}$ est vraie. Contradiction.
- Ainsi, $\mathcal{G}$ est vraie, mais alors cela signifie « $\mathcal{G}$ n'est pas démontrable ».
- Conclusion : $\mathcal{G}$ est vraie, mais n'est pas démontrable.

□

4.3. La diagonale de Cantor

Pourquoi le procédé expliqué ci-dessus s'appelle un procédé de diagonalisation ? C'est facile à comprendre sur le schéma ci-dessous. On a des formules $\mathcal{P}$ (ou $\mathcal{P}(x)$) qui, on l'a vu, sont indexées par des entiers de $\mathbb{N}$ (leur nombre de Gödel). Et on a des entiers $n \in \mathbb{N}$ qui jouent le rôle de x dans l'évaluation. La phrase de Gödel $\mathcal{G}$ est l'application de la formule $\mathcal{P}$ exactement en $n = \ulcorner \mathcal{P} \urcorner$.



D'où vient cette idée de la diagonalisation ? Elle est déjà présente dans le paradoxe de Russell où la question « Est-ce que $x \in y$? » donne sur la diagonale « Est-ce que $y \in y$? ». Cette méthode a surtout été utilisée par Cantor pour prouver que l'ensemble des réels n'est pas dénombrable. Étudions sa preuve.

Théorème 3.
$\mathbb{R}$ n'est pas dénombrable.

Démonstration. Supposons par l'absurde que l'ensemble des réels $[0,1[$ soit un ensemble dénombrable, c'est-à-dire en bijection avec $\mathbb{N}$. En termes plus familiers, cela signifie que tous les réels de $[0,1[$ peuvent être numérotés $x_0, x_1, x_2, \ldots$ c'est-à-dire $[0,1[= \{x_k\}_{k\in\mathbb{N}}$.
On écrit chaque réel de $[0,1[$ par son écriture décimale, par exemple

$$x_0 = 0.189\,335\ldots$$
$$x_1 = 0.772\,004\ldots$$
$$x_2 = 0.191\,542\ldots$$
$$x_3 = 0.053\,477\ldots$$

Cette écriture décimale est unique si on s'interdit l'écriture avec une infinité de 9 à la fin. Par exemple pour $x = 0.123\,999\,999\ldots$ on choisit l'écriture $x = 0.124\,000\,000\ldots$ On remplit un tableau avec les décimales de tous les $(x_k)_{k\geqslant 0}$ (sans le 0 avant la virgule).

⋮						
⋮						
x_3	0	5	3	**4**	7	7
x_2	1	9	**1**	5	4	2
x_1	7	**7**	2	0	0	4
x_0	**1**	8	9	3	3	5
	Décimales					

⋮					
⋮					
0	5	3	**1**	7	7
1	9	**2**	5	4	2
7	**1**	2	0	0	4
2	8	9	3	3	5

Ensuite, on change le chiffre numéro k de x_k selon la règle suivante :

- si ce chiffre est 1, on le change en 2,
- et si ce chiffre est différent de 1, on le change en 1.

Avec ce procédé on peut lire les décimales d'un nouveau réel $x' \in [0,1[$ sur la diagonale. Sur notre dessin ce serait $x' = 0.2121\ldots$
Mais par construction x' ne peut pas être l'un des x_k car x' et x_k ont leur décimale numéro k distincte. C'est une contradiction car $\{x_k\}_{k\in\mathbb{N}}$ devait énumérer tous les réels de $[0,1[$. Conclusion : $[0,1[$ n'est pas dénombrable et par conséquent $\mathbb{R}$ non plus. □

Chapitre 15

Fonctions récursives primitives

Nous définissons et étudions les fonctions qui expriment les calculs « faciles » de l'arithmétique.

Dans ce chapitre, notre point de vue est *sémantique*, c'est-à-dire qu'on va étudier des objets de la « réalité » : les entiers, les nombres premiers, l'addition, ...
Dans le chapitre suivant, nous ferons le lien avec le langage $\mathcal{L}_0$ de l'arithmétique de Peano, qui sera le point de vue *syntaxique* où tout s'exprime par des suites de symboles. Le lien entre les deux sera le théorème de représentabilité : tout ce qui est vrai pour les fonctions récursives primitives peut être prouvé dans le langage de l'arithmétique de Peano.

1. Les fonctions récursives primitives

Les fonctions récursives primitives sont les fonctions qui sont « faciles à calculer » au sens du chapitre précédent, c'est-à-dire qu'on peut en calculer les valeurs à l'aide d'un algorithme n'utilisant que des boucles « pour ».
Ce sont des fonctions $f : \mathbb{N}^p \to \mathbb{N}$. Elles sont définies à partir de trois fonctions de base et deux opérations élémentaires. Cependant, cet ensemble contient toutes les fonctions et opérations usuelles sur les entiers : addition, multiplication, puissance, ...

1.1. Définition

On note $\mathcal{F}_p = \{f : \mathbb{N}^p \to \mathbb{N}\}$ et $\mathcal{F} = \bigcup_{p \geqslant 1} \mathcal{F}_p$. On rappelle que $\mathbb{N}^p = \mathbb{N} \times \cdots \times \mathbb{N}$. On notera souvent le p-uplet $(x_1, \ldots, x_p) \in \mathbb{N}^p$ par $\mathbf{x}$. Les éléments de $\mathcal{F}$ sont donc des fonctions de p variables (où p peut varier d'une fonction à l'autre).
La définition des fonctions récursives repose sur trois fonctions de base et deux opérations.
Fonctions de base

- Z : la fonction nulle

$$\begin{aligned} Z : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto 0 \end{aligned}$$

- S : la fonction successeur

$$\begin{aligned} S : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto x + 1 \end{aligned}$$

- P_i^p : les projections (pour $1 \leqslant i \leqslant p$).

$$\begin{aligned} P_i^p : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \ldots, x_p) &\longmapsto x_i \end{aligned}$$

Opération de composition

- Données : $f_1, \dots, f_q : \mathbb{N}^p \to \mathbb{N}$ et $g : \mathbb{N}^q \to \mathbb{N}$.
- Composition : la fonction $g(f_1, \dots, f_q) : \mathbb{N}^p \to \mathbb{N}$ définie par

$$\boxed{\mathbf{x} \mapsto g(f_1(\mathbf{x}), \dots, f_q(\mathbf{x}))}$$

Opération de récurrence

- Données :
  — $g : \mathbb{N}^p \to \mathbb{N}$ (initialisation)
  — $h : \mathbb{N}^{p+2} \to \mathbb{N}$ (étape de récurrence)
- Nouvelle fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$ définie par :

$$\boxed{f(\mathbf{x}, 0) = g(\mathbf{x}) \quad \text{et} \quad f(\mathbf{x}, y+1) = h(\mathbf{x}, y, f(\mathbf{x}, y))}$$

Il faut y voir une récurrence comme pour toute suite définie par une initialisation u_0 et une relation $u_{n+1} = h(n, u_n)$. La récurrence se fait sur la dernière variable (ici y).

Définition.
L'ensemble PR des ***fonctions récursives primitives*** est le plus petit ensemble de $\mathcal{F}$ qui contient Z, S, P_i^p (pour tout $p \geqslant 1$, $1 \leqslant i \leqslant p$) et qui est stable pour les opérations de composition et de récurrence.

Que signifie « être stable »? Par exemple, pour la composition, si $f_1, \dots, f_q \in PR$ et $g \in PR$, alors la composée $g(f_1, \dots, f_q)$ est encore dans PR. Idem pour l'opération de récurrence.
Note. Le terme « récursive primitive » est mal choisi, mais c'est la terminologie usuelle. Un nom plus évocateur serait celui de fonction « facilement calculable », mais ce terme est utilisé dans d'autres situations.

1.2. Premiers exemples

(a) Fonctions constantes $\mathbb{N} \to \mathbb{N}$
Les fonctions constantes $f : \mathbb{N} \to \mathbb{N}$ sont récursives primitives. On le sait pour la fonction nulle Z. Pour la constante 1, on calcule $S(Z(x)) = S(0) = 1$, donc $f = S \circ Z$ est récursive primitive comme composée de deux fonctions de base. Plus généralement, $x \mapsto k$ est dans PR (par k compositions de S appliquées à Z).

(b) Fonctions constantes $\mathbb{N}^2 \to \mathbb{N}$
Soit $k \in \mathbb{N}$ un entier fixé. On définit :

- initialisation : $g : \mathbb{N} \to \mathbb{N}$, la fonction constante égale à k, qui est dans PR.
- étape de récurrence : $h : \mathbb{N}^3 \to \mathbb{N}$ définie par $h(x, y, z) = z$. C'est la projection P_3^3, donc elle est récursive primitive.

L'opération de récurrence construit la fonction $f : \mathbb{N}^2 \to \mathbb{N}$ appartenant à PR telle que $f(x, 0) = k$ et $f(x, y+1) = f(x, y)$. Par récurrence sur y, $f(x, y) = k$ pour tout y. Donc $f : \mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto k$ est récursive primitive.

On prouverait, par récurrence sur p, que toute fonction constante $f : \mathbb{N}^p \to \mathbb{N}$ est récursive primitive.

(c) Opérations arithmétiques élémentaires

Proposition 1.
Les fonctions suivantes sont récursives primitives :

- *l'addition :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x + y$
- *la multiplication :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x \times y$
- *l'exponentiation :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x^y$

Démonstration.

- L'addition vérifie la relation de récurrence : $x + (y+1) = (x + y) + 1$. Ainsi on pose l'initialisation $g : \mathbb{N} \to \mathbb{N}$ définie par $g(x) = x$ (en fait $g(x) = P_1^1(x) = x$ est donc récursive primitive). Ensuite on

pose $h : \mathbb{N}^3 \to \mathbb{N}$ définie par $h(x,y,z) = S(z)$ (en fait $h = S \circ P_3^3$, donc h est récursive primitive). Par l'opération de récurrence on obtient $f : \mathbb{N}^2 \to \mathbb{N}$ récursive primitive.
Vérifions que $f(x,y) = x + y$. On fixe x et on fait une récurrence sur y.
— Pour $y = 0$, $f(x,0) = g(x) = x = x + 0$.
— Supposons $f(x,y) = x + y$. Alors :

$$f(x,y+1) = h(x,y,f(x,y)) = S(f(x,y)) = f(x,y) + 1 = (x+y) + 1 = x + (y+1)$$

On a prouvé par récurrence sur y que pour tout x, y, $f(x,y) = x + y$. On savait que f était récursive primitive, donc l'addition l'est aussi.

- Pour la multiplication, on utilise la relation $x \times (y+1) = x \times y + x$. Ce sera à faire dans les exercices.
- L'exponentiation est aussi à faire dans les exercices.

□

(d) La soustraction entière

Proposition 2.

- *La fonction prédécesseur* prec : $\mathbb{N} \to \mathbb{N}$,

$$\text{prec}(x) = \begin{cases} x-1 & si\ x \geqslant 1 \\ 0 & si\ x = 0 \end{cases}$$

est récursive primitive.
- *La* **soustraction entière** $\mathbb{N}^2 \to \mathbb{N}$, $(x,y) \mapsto x \dot{-} y$ *est aussi récursive primitive.*

$$\boxed{x \dot{-} y = \begin{cases} x-y & si\ x \geqslant y \\ 0 & sinon \end{cases}}$$

Démonstration.

- La fonction prec : $\mathbb{N} \to \mathbb{N}$ vérifie la relation de récurrence suivante : $\text{prec}(0) = 0$, et pour $y \geqslant 0$, $\text{prec}(y+1) = y$. Cela en fait intuitivement une fonction récursive primitive. Mais techniquement c'est un peu plus compliqué car l'opération de récurrence fournit une fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$. On procède ainsi :
— $g(x) = Z(x) = 0$ est dans PR.
— $h(x,y,z) = P_2^3(x,y,z) = y$ est aussi dans PR.
Donc $f : \mathbb{N}^2 \to \mathbb{N}$ définie par $f(x,0) = 0$ et qui vérifie $f(x,y+1) = h(x,y,f(x,y)) = y$ est dans PR. Ainsi $f(x,y)$ est récursive primitive.
Note : $f(x,y)$ est une fonction de deux variables où la variable x ne sert à rien. On s'en débarrasse par la composition suivante : prec : $\mathbb{N} \to \mathbb{N}$, définie par $\text{prec}(y) = f(P_1^1(y), P_1^1(y)) = f(y,y)$ qui est donc bien récursive primitive.
- La soustraction entière vérifie la relation de récurrence : $x \dot{-} 0 = x$ et $x \dot{-} (y+1) = \text{prec}(x \dot{-} y)$. On pose $g(x) = P_1^1(x) = x$ et $h(x,y,z) = \text{prec}(P_3^3(x,y,z)) = \text{prec}(z)$. Puisque g et h sont récursives primitives, par l'opération de récurrence, on obtient une fonction $f : \mathbb{N}^2 \to \mathbb{N}$ récursive primitive qui vérifie $f(x,y) = x \dot{-} y$.

□

1.3. Remarques

- **Autre définition de** PR**.** On peut aussi définir l'ensemble des fonctions récursives primitives « par le bas ».

$$PR_0 = \{Z, S\} \cup \{P_i^p\}_{p \geqslant 1, 1 \leqslant i \leqslant p}$$

$$PR_{n+1} = PR_n \cup \{\text{fonctions obtenues par opération de composition à partir de } PR_n\}$$
$$\cup \{\text{fonctions obtenues par opération de récurrence à partir de } PR_n\}$$

Et alors $PR = \bigcup_{n\geqslant 0} PR_n$.

- **Variables.** On peut permuter les variables comme on veut. Par exemple, si $f : \mathbb{N}^2 \to \mathbb{N}, (x,y) \mapsto f(x,y)$ est récursive primitive, alors $(x,y) \mapsto f(y,x)$ est aussi récursive primitive. En effet, $f(y,x) = f(P_2^2(x,y), P_1^2(x,y))$. Ainsi, on utilise souvent les variables x, y, z directement et dans l'ordre que l'on veut.
- **Fonctions $f : \mathbb{N}^0 \to \mathbb{N}$.** Les éléments de $\mathbb{N}^p$ sont des p-uplets $(x_1, \dots, x_p)$. Pour $p = 0$, $\mathbb{N}^0$ est constitué d'un seul 0-uplet noté $()$. Ainsi $\mathbb{N}^0 = \{()\}$ et une fonction $f : \mathbb{N}^0 \to \mathbb{N}$ n'a qu'une seule valeur (car il y a un seul élément dans son ensemble de départ), c'est donc une fonction constante.
  Pour l'instant nous avions exclu ce cas $p = 0$. Mais cela peut être utile de l'autoriser et d'ajouter par convention que les fonctions $f : \mathbb{N}^0 \to \mathbb{N}$ sont récursives primitives (elles sont considérées comme des constantes).
  Cette convention permet d'utiliser l'opération de récurrence avec $p = 0$. Cela simplifie la preuve que la fonction prec est récursive primitive (en effet, dans la preuve, la variable x était inutile). On renvoie aux exercices pour montrer que $n!$ est récursive primitive avec cette méthode.

1.4. Algorithmes

Faisons le lien avec le chapitre précédent. Si $f : \mathbb{N}^p \to \mathbb{N}$ est une fonction récursive primitive, alors quel que soit $\mathbf{x} \in \mathbb{N}^p$, on peut calculer $f(\mathbf{x})$ par un algorithme n'utilisant que des boucles « pour » (avec un nombre d'itérations bien contrôlé).
Si on veut démontrer que les fonctions de PR vérifient une certaine propriété, il suffit de la vérifier pour les trois fonctions de base et de montrer que cette propriété est conservée par application des opérations de composition et de récurrence.
Il est clair que $Z(n) = 0$, $S(x) = x + 1$ et $P_i^p(\mathbf{x}) = x_i$ sont calculables facilement. Si f est la composée $g(f_1, \dots, f_q)$ et qu'il existe des algorithmes faciles pour $f_1, \dots, f_q$ et g, alors on calcule $y_1 = f_1(\mathbf{x}), \dots, y_q = f_q(\mathbf{x})$ par ces algorithmes, puis $g(y_1, \dots, y_q)$ qui donne ainsi $f(\mathbf{x})$ par une suite finie d'algorithmes faciles.
Si f est obtenue par opération de récurrence : $f(\mathbf{x}, 0) = g(\mathbf{x})$ est supposé se calculer par un algorithme facile, ce qui permet de calculer $f(\mathbf{x}, 1) = h(\mathbf{x}, 0, f(\mathbf{x}, 0))$ où h se calcule par un algorithme facile. De proche en proche, on calcule $f(\mathbf{x}, 2)$, $f(\mathbf{x}, 3)$,... jusqu'à la valeur voulue.

2. Ensembles et prédicats

2.1. Ensembles

Définition.
Soit $A \subset \mathbb{N}^p$. La *fonction caractéristique* $\chi_A : \mathbb{N}^p \to \mathbb{N}$ est définie par :

$$\chi_A(\mathbf{x}) = \begin{cases} 1 & \text{si } \mathbf{x} \in A \\ 0 & \text{sinon} \end{cases}$$

On la note aussi parfois $\mathbb{1}_A$.

Définition.
Un ensemble $A \subset \mathbb{N}^p$ est un *ensemble récursif primitif* si sa fonction caractéristique χ_A est une fonction récursive primitive.

Par exemple, l'ensemble $A \subset \mathbb{N}$ des entiers pairs est un ensemble récursif primitif. En effet, on peut définir sa fonction caractéristique par la relation de récurrence suivante :

$$\chi_A(0) = 1 \quad \text{et} \quad \chi_A(n+1) = 1 \dot{-} \chi_A(n)$$

Ceci fait de χ_A une fonction récursive primitive.

Proposition 3.
Si A et B sont des ensembles récursifs primitifs de $\mathbb{N}^p$, alors les ensembles $A \cup B$, $A \cap B$ et $\mathbb{N}^p \setminus A$ sont aussi des ensembles récursifs primitifs.

Pour la preuve, on a besoin de la ***fonction signe*** $\mathrm{sgn} : \mathbb{N} \to \mathbb{N}$ définie par :

$$\mathrm{sgn}(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n > 0 \end{cases}$$

Elle est récursive primitive car elle s'écrit aussi $\mathrm{sgn}(n) = 1 \dot{-} (1 \dot{-} n)$.

Démonstration. Les fonctions caractéristiques de ces ensembles s'expriment à l'aide des fonctions récursives primitives χ_A et χ_B :

- $\chi_{A \cap B} = \chi_A \times \chi_B$
- $\chi_{A \cup B} = \mathrm{sgn}(\chi_A + \chi_B)$
- $\chi_{\mathbb{N}^p \setminus A} = 1 \dot{-} \chi_A$

□

Proposition 4.
Si f est une fonction récursive primitive et $A = \{\mathbf{x} \in \mathbb{N}^p \mid f(\mathbf{x}) = 0\}$, alors A est un ensemble récursif primitif.

Démonstration. On a $\chi_A(\mathbf{x}) = 1 \dot{-} f(\mathbf{x})$. □

Proposition 5 (Définition par cas)**.**
Soient $g, h : \mathbb{N}^p \to \mathbb{N}$ des fonctions récursives primitives et $A \subset \mathbb{N}^p$ un ensemble récursif primitif. La fonction $f : \mathbb{N}^p \to \mathbb{N}$ définie par :

$$f(\mathbf{x}) = \begin{cases} g(\mathbf{x}) & \textit{si } \mathbf{x} \in A \\ h(\mathbf{x}) & \textit{sinon} \end{cases}$$

est une fonction récursive primitive.

Démonstration. On peut écrire $f(\mathbf{x}) = g(\mathbf{x}) \times \chi_A(\mathbf{x}) + h(\mathbf{x}) \times \chi_{\mathbb{N}^p \setminus A}(\mathbf{x})$. Elle s'exprime comme somme et produits de fonctions récursives primitives, donc l'est également. □

2.2. Prédicats

Définition.
Un prédicat $P : \mathbb{N}^p \to \{V, F\}$ à p places est un ***prédicat récursif primitif*** si l'ensemble

$$A = \{\mathbf{x} \in \mathbb{N}^p \mid P(\mathbf{x}) \text{ est vrai}\}$$

est un ensemble récursif primitif.
Autrement dit, cela signifie que la fonction caractéristique :

$$\chi_P(\mathbf{x}) = \begin{cases} 1 & \text{si } P(\mathbf{x}) \text{ est vrai} \\ 0 & \text{sinon} \end{cases}$$

est une fonction récursive primitive.

Proposition 6.
Si P et Q sont des prédicats à p places récursifs primitifs, alors les prédicats $\neg P$, $P \wedge Q$, $P \vee Q$, $P \to Q$, $P \leftrightarrow Q$ le sont aussi.

Démonstration. Notons A et B les ensembles récursifs primitifs associés respectivement à P et Q.

- L'ensemble $\mathbb{N}^p \setminus A$ est un ensemble récursif primitif, donc $\neg P$ est un prédicat récursif primitif. (Autre démonstration : $\chi_{\neg P}(\mathbf{x}) = 1 \dot{-} \chi_P(\mathbf{x})$.)

- Le prédicat $P \wedge Q$ a pour ensemble associé $A \cap B$ qui est bien un ensemble récursif primitif (ou bien $\chi_{P\wedge Q}(\mathbf{x}) = \chi_P(\mathbf{x}) \times \chi_Q(\mathbf{x})$).
- Le prédicat $P \vee Q$ a pour ensemble associé $A \cup B$ (ou bien $\chi_{P\vee Q}(\mathbf{x}) = \mathrm{sgn}(\chi_P(\mathbf{x}) + \chi_Q(\mathbf{x}))$).
- Pour les connecteurs $\to$ et $\leftrightarrow$, ils peuvent s'exprimer à l'aide des connecteurs précédents.

□

Proposition 7.
Les prédicats « $x > y$ » et « $x \geqslant y$ » sont récursifs primitifs.

Démonstration. Leurs fonctions caractéristiques s'écrivent :
- $\chi_>(x, y) = \mathrm{sgn}(x \dot{-} y)$
- $\chi_\geqslant(x, y) = \mathrm{sgn}((x+1) \dot{-} y)$

□

2.3. Schémas bornés

Rechercher des éléments définis par un minimum, ou par les quantificateurs $\forall$ ou $\exists$, conserve le caractère récursif primitif à condition qu'ils soient bornés. Ce sera fondamental pour les applications à l'arithmétique que l'on verra juste après.

Schéma du minimum borné
Soit $P(\mathbf{x}, t)$ un prédicat. Fixons $\mathbf{x}$ et y. On peut considérer :

$$\min\{t \leqslant y \mid P(\mathbf{x}, t) \text{ est vrai}\}.$$

Si l'ensemble $\{t \leqslant y \mid P(\mathbf{x}, t) \text{ est vrai}\}$ est vide, on pose par convention $\min = y + 1$.
Si A est un ensemble, on définit de même :

$$\min\{t \leqslant y \mid (\mathbf{x}, t) \in A\}$$

Proposition 8.
Si P est un prédicat récursif primitif, alors la fonction $f(\mathbf{x}, y)$ définie par :

$$f(\mathbf{x}, y) = \min\{t \leqslant y \mid P(\mathbf{x}, t) \textit{ est vrai}\}$$

est une fonction récursive primitive.
Il en est de même pour un ensemble A récursif primitif.

> **Exemple.**
> Considérons la fonction racine carrée entière $\mathbb{N} \to \mathbb{N}$, qui à y associe $\lfloor \sqrt{y} \rfloor$ (par exemple $\lfloor \sqrt{10} \rfloor = 3$).
> En fait, $\lfloor \sqrt{y} \rfloor$ est le plus petit entier dont le carré de son successeur dépasse y, c'est-à-dire :
>
> $$\lfloor \sqrt{y} \rfloor = \min\{t \leqslant y \mid (t+1)^2 > y\}$$
>
> C'est donc une fonction récursive primitive.

Démonstration. Intuitivement, pour $\mathbf{x}$ fixé et y fixé, il y a un nombre fini de valeurs $t \leqslant y$ à tester, c'est donc facile à calculer au sens algorithmique. Fixons $\mathbf{x}$ et trouvons la formule de récurrence.
- Initialisation : $(y = 0)$. Si $P(\mathbf{x}, 0)$ est vrai, on pose $g(\mathbf{x}) = 0$, sinon $g(\mathbf{x}) = 1$ (par la convention $\min = y + 1$).
- Formule de récurrence. On cherche $f(\mathbf{x}, y+1) = \min\{t \leqslant y+1 \mid P(\mathbf{x}, t)\}$. On trouve une formule par disjonction de cas (voir la Proposition 5) :

$$f(\mathbf{x}, y+1) = \begin{cases} f(\mathbf{x}, y) & \text{si } f(\mathbf{x}, y) \leqslant y \\ y+1 & \text{si } f(\mathbf{x}, y) = y+1 \text{ et } P(\mathbf{x}, y+1) \text{ est vrai} \\ y+2 & \text{si } f(\mathbf{x}, y) = y+1 \text{ et } P(\mathbf{x}, y+1) \text{ est faux} \end{cases}$$

□

Quantification bornée

Proposition 9.
Soit $A \subset \mathbb{N}^{p+1}$ un ensemble récursif primitif. Alors les ensembles :
- $B = \{(\mathbf{x}, y) \mid \forall t \leqslant y, (\mathbf{x}, t) \in A\}$
- $C = \{(\mathbf{x}, y) \mid \exists t \leqslant y, (\mathbf{x}, t) \in A\}$

sont aussi des ensembles récursifs primitifs.

Démonstration. Notons $A_t = \{(\mathbf{x}, t) \in A\}$. Alors $\chi_B = \chi_{A_0} \times \chi_{A_1} \times \cdots \times \chi_{A_y} = \prod_{t=0}^{y} \chi_{A_t}$ et $\chi_C = \operatorname{sgn}\left(\sum_{t=0}^{y} \chi_{A_t}\right)$. La somme et le produit bornés se définissent facilement par l'opération de récurrence. □

Cela fonctionne aussi sous la forme de prédicat. Par exemple, « être un carré parfait » est un prédicat récursif primitif. On rappelle que y est un carré parfait s'il existe t tel que $y = t^2$ ($0, 1, 4, 9, 16, 25, \ldots$ sont des carrés parfaits). Ce prédicat se définit par la quantification bornée :

$$\exists t \leqslant y,\ y = t^2.$$

2.4. Arithmétique

Proposition 10.
Les fonctions suivantes sont récursives primitives :
- $q(x, y)$ *le quotient de la division euclidienne de x par y (avec la convention $q(x, 0) = x + 1$).*
- $\pi(n)$ *qui calcule le n-ième nombre premier.*
- exposant(n, i) *qui calcule l'exposant de $\pi(i)$ dans la décomposition de n en facteurs premiers.*

Proposition 11.
L'ensemble $\mathcal{P}$ des nombres premiers est un ensemble récursif primitif.

Nous allons prouver ces deux propositions ensemble.

Démonstration.
- $q(x, y) = \min\{q \leqslant x \mid (q + 1) \times y > x\}$ est donc récursive primitive.
- Le reste $r(x, y) = x \dot{-} (q(x, y) \times y)$ l'est aussi.
- Le prédicat est_divisible(x, y) (qui vaut vrai si y divise x) s'obtient par $r(x, y) = 0$. C'est donc un prédicat récursif primitif.
- Être un nombre premier, c'est être un nombre strictement plus grand que 1 et ne pas avoir d'autres diviseurs que 1 et lui-même. Ainsi $\mathcal{P}$ est un ensemble récursif primitif :
$$\mathcal{P} = \{x \mid x > 1\} \cap \{x \mid \forall y \leqslant x \quad (y = 1 \vee y = x \vee \neg\text{est\_divisible}(x, y))\}$$
De façon équivalente, le prédicat est_premier(n) est récursif primitif.
- La fonction $\pi(n)$ qui renvoie le n-ième nombre premier (noté π_n, avec $\pi_1 = 2$) vérifie la relation de récurrence suivante, qui en fait une fonction récursive primitive :
$$\pi(n + 1) = \min\{t \leqslant \pi(n)! + 1 \mid t > \pi(n) \wedge \text{est\_premier}(t)\}$$
On rappelle qu'il existe toujours un nombre premier entre π_n et $\pi_n! + 1$ (ou même entre m et $2m$).
- L'exposant de π_i dans la décomposition de n en facteurs premiers est défini par :
$$\text{exposant}(n, i) = \min\{k \leqslant n \mid \neg\text{est\_divisible}(n, \pi_i^{k+1})\}$$
C'est donc bien une fonction récursive primitive.

□

Terminons par un contre-exemple célèbre.

Exemple.
La fonction d'Ackermann $A : \mathbb{N}^2 \to \mathbb{N}$ n'est pas récursive primitive. Elle est définie ainsi :
- $A(0,n) = n+1$
- $A(m+1,0) = A(m,1)$
- $A(m+1,n+1) = A(m,A(m+1,n))$

Pour m fixé, la fonction $A_m : \mathbb{N} \to \mathbb{N}, n \mapsto A(m,n)$ est récursive primitive. Mais la fonction $A : \mathbb{N}^2 \to \mathbb{N}$ ne l'est pas à cause de la récurrence imbriquée qui la fait croître plus vite que n'importe quelle fonction récursive primitive.

3. Application au codage de Gödel

Nous allons justifier, dans le monde des entiers, que toutes les opérations que l'on souhaite faire sur les nombres de Gödel se font par des fonctions récursives primitives, c'est-à-dire comme on l'avait présenté avant, que les calculs se font par des algorithmes avec des boucles « pour ».

3.1. Longueur, exposant, concaténation, substitution

On considère un nombre de Gödel $n = \pi_1^{c_1} \times \cdots \times \pi_l^{c_l}$ (avec les $c_i \geqslant 1$).
- On veut d'abord pouvoir retrouver l à partir de n de façon récursive primitive :
$$\text{longueur}(n) = \min\{i \leqslant n \mid \pi_{i+1} \text{ ne divise pas } n\}$$
- On a déjà étudié la fonction exposant(n,i) qui renvoie l'exposant c_i. On sait ainsi décoder un nombre de Gödel par une fonction récursive primitive.
- On a vu au chapitre précédent la formule de la concaténation. Si $n = \prod_{i=1}^{l} \pi_i^{\alpha_i}$ et $m = \prod_{j=1}^{l'} \pi_j^{\beta_j}$ alors :
$$n * m = n \times \prod_{j=1}^{l'} \pi_{j+l}^{\beta_j}$$
Ce sont des produits finis, donc c'est une opération récursive primitive.
- Considérons la fonction substitution(p,v,n) : pour une formule $\mathcal{P}$ (donnée par son nombre de Gödel p), de variable libre x (donnée par son code v), elle renvoie le code de Gödel de $\mathcal{P}(n)$ (c'est-à-dire $\mathcal{P}(n/x)$).
- On commence par extraire chaque code c_i de $\mathcal{P}$, on compare ce code à celui de la variable x, si c'est le même on remplace ce symbole par le terme n en utilisant la concaténation. C'est donc une succession de fonctions récursives primitives, donc la substitution est une fonction récursive primitive.

3.2. Variables, termes, formules

Décider si un entier n représente une variable, un terme, ou une formule valide sont des prédicats récursifs primitifs. On renvoie au chapitre précédent sans s'étendre davantage.
- est_variable(n)
- est_terme(n)
- est_formule(n)

3.3. Axiomes de Peano

On nous donne un entier n (un nombre de Gödel) et on doit décider si la formule qu'il encode est un des axiomes de Peano. Prenons l'exemple du premier axiome de Peano « $\forall x\ S(x) \neq 0$ » que l'on écrirait dans $\mathcal{L}_0$ par « $\forall x \neg Sx = 0$ », on note n_1 son nombre de Gödel.
Pour savoir si n code la formule de cet axiome, il suffit de tester si $n = n_1$. On fait cela pour les sept axiomes.
La difficulté vient des axiomes de récurrence :

$$\mathcal{R}ec_{\mathcal{P}(x),y} : \quad \mathcal{P}(0) \wedge \big[\forall y \quad \mathcal{P}(y) \to \mathcal{P}(S(y))\big] \quad \to \quad \forall x \ \ \mathcal{P}(x)$$

Il s'agit d'un schéma d'axiomes, il y a un axiome pour chaque formule $\mathcal{P}(x)$. Il y a donc à priori une infinité de formules et de variables à tester. Mais heureusement, pour un nombre de Gödel n donné, si x est une variable présente dans la formule de n, alors son code v est nécessairement inférieur à n. De même, si $\mathcal{P}$ est une sous-formule de la formule n, alors son code p vérifie aussi $p \leqslant n$.
On peut ainsi construire un prédicat récursif primitif sous la forme :

$$\begin{aligned}\text{est\_axiome\_recurrence}(n) = \exists \mathcal{P} \text{ de code } p \leqslant n \quad \exists x \text{ de code } v \leqslant n \quad \exists y \text{ de code } w \leqslant n \\ \text{tel que est\_formule}(\mathcal{P}) \text{ et est\_variable}(x) \text{ et est\_variable}(y) \\ \text{et } n \text{ code la formule } \mathcal{R}ec_{\mathcal{P}(x),y}\end{aligned}$$

Ainsi on sait détecter tous les axiomes de Peano de façon récursive primitive.

3.4. Preuves à la Hilbert

Dans les deux premières parties de ce livre, nous avons présenté les preuves avec leurs règles d'inférence dans leur écriture en déduction naturelle. Comme son nom l'indique, les preuves en déduction naturelle ont une écriture proche de l'écriture d'une preuve mathématique. Cependant il y a des hypothèses et des sous-hypothèses, et cette présentation n'est pas adaptée à un traitement algorithmique.
Nous allons voir l'écriture dans le système de Hilbert qui est plus adapté au codage de Gödel.

- Tout d'abord on se passe d'hypothèses. Si on devait prouver $\mathcal{P} \vdash \mathcal{Q}$, on prouverait $\vdash (\mathcal{P} \to \mathcal{Q})$.
- Il n'y a qu'une seule règle d'inférence pour la logique propositionnelle :

$$\mathcal{P}, \mathcal{P} \to \mathcal{Q} \quad \vdash \quad \mathcal{Q} \qquad (\textit{modus ponens})$$

- Par contre on a l'apparition d'axiomes qui sont des formules supposées toujours vraies (quel que soit ce que l'on souhaite démontrer).
- Pour la logique propositionnelle il y a trois axiomes (il existe des variantes, ici la variante de Lukasiewicz) :
  1. $\mathcal{P} \to (\mathcal{Q} \to \mathcal{P})$ (ajout d'hypothèse)
  2. $(\mathcal{P} \to (\mathcal{Q} \to \mathcal{R})) \to ((\mathcal{P} \to \mathcal{Q}) \to (\mathcal{P} \to \mathcal{R}))$ (distribution de l'implication)
  3. $(\neg\mathcal{Q} \to \neg\mathcal{P}) \to (\mathcal{P} \to \mathcal{Q})$ (contraposée)
- Ainsi pour vérifier si une suite $\mathcal{P}_1, \ldots, \mathcal{P}_l$ de formules est bien l'écriture d'une preuve de $\mathcal{Q}$, il suffit que pour tout i :
  — soit $\mathcal{P}_i$ est un des axiomes,
  — soit $\mathcal{P}_i$ est obtenu par *modus ponens* à partir des lignes précédentes, c'est-à-dire qu'il existe $j, k < i$ tels que $\mathcal{P}_k$ soit $\mathcal{P}_j \to \mathcal{P}_i$,
  et enfin on vérifie que $\mathcal{P}_l = \mathcal{Q}$.
- Pour la logique du premier ordre, on ajoute les axiomes suivants :
  — $(\forall x\ \mathcal{P}(x)) \to \mathcal{P}(t)$
  — $\big[\forall x\ (\mathcal{P} \to \mathcal{Q}(x))\big] \to \big[(\mathcal{P} \to \forall x\ \mathcal{Q}(x))\big]$ si $\mathcal{P}$ ne contient pas de variable libre x.
  — $x = x$

— $x = y \to (\mathcal{P}(x) \to \mathcal{P}(y))$

- Et on ajoute une seconde règle d'inférence, qui est une variante de la règle d'introduction de $\forall$:

$$\text{si} \quad \vdash \mathcal{P} \quad \text{alors} \quad \vdash \forall x\ \mathcal{P}$$

3.5. Être une preuve

Ainsi on comprend que vérifier si un super-nombre de Gödel N est une preuve de n se fait facilement à l'aide du système de Hilbert. On rappelle qu'on n'a pas d'hypothèses (pour prouver $\Gamma \vdash \mathcal{Q}$, on prouve $\vdash (\Gamma \to \mathcal{Q})$), que les règles d'inférence sont très simples (*modus ponens* et introduction de $\forall$) et qu'une preuve est juste une suite linéaire de formules.

On construit ainsi un prédicat est_preuve(N, n) récursif primitif qui vérifie si la suite de formules $\mathcal{P}_i$ codée par le super-nombre de Gödel N est bien une preuve valide de la formule codée par le nombre de Gödel n.

La construction de ce prédicat se fait ainsi (pour la logique propositionnelle) :

- extraire de N les expressions $\mathcal{P}_i$, $i = 1, \ldots, l$ et de n l'expression $\mathcal{Q}$,
- vérifier que ces expressions sont des formules valides,
- pour chaque $i = 1, \ldots, l$:
  — vérifier si $\mathcal{P}_i$ est un axiome,
  — ou si $\mathcal{P}_i$ se déduit par *modus ponens* des lignes précédentes (c'est-à-dire qu'il existe $j, k < i$ tels que $\mathcal{P}_k$ soit $\mathcal{P}_j \to \mathcal{P}_i$).
- vérifier si la dernière formule $\mathcal{P}_l$ est bien $\mathcal{Q}$.

Chapitre 16

Fonctions représentables

Il s'agit de justifier que l'arithmétique de Peano permet d'exprimer toutes les fonctions récursives primitives.

Note : Il s'agit d'un chapitre technique. Le lecteur pressé peut juste se concentrer sur le résultat énoncé au début. Cependant, il s'agit aussi d'un chapitre charnière puisqu'on va faire le lien entre la sémantique (le monde des entiers) et la syntaxe (le langage de l'arithmétique de Peano). On y trouve aussi une des prouesses techniques de Gödel : la fonction β. Elle permet de coder une liste quelconque de nombres à l'aide de seulement deux entiers.

1. Un prédicat récursif primitif est représentable

1.1. Prédicat représentable

Pour $p \geqslant 1$ fixé, on note $\mathbf{x} = (x_1, \dots, x_p)$. Si $\mathbf{n} = (n_1, \dots, n_p) \in \mathbb{N}^p$, on note $\overline{\mathbf{n}} = (\overline{n}_1, \dots, \overline{n}_p)$ la suite des numéraux correspondants.

Définition.
Un prédicat $R(\mathbf{x})$ à p places est ***représentable*** s'il existe une formule $\mathcal{P}(\mathbf{x})$ du langage $\mathcal{L}_0$ telle que pour tout $\mathbf{n} \in \mathbb{N}^p$:

- si $R(\mathbf{n})$ est vrai alors : $\quad \mathcal{PA} \vdash \mathcal{P}(\overline{\mathbf{n}})$
- si $R(\mathbf{n})$ est faux alors : $\quad \mathcal{PA} \vdash \neg\mathcal{P}(\overline{\mathbf{n}})$

Remarque : la dénomination complète est « prédicat fortement représentable par la théorie $\mathcal{PA}$ de l'arithmétique de Peano ».
Noter bien le passage d'un entier au numéral ; par exemple si $p = 1$, le prédicat s'évalue en $R(n)$ (avec $n \in \mathbb{N}$), la formule $\mathcal{P}$ s'évalue en $\mathcal{P}(\overline{n})$, avec le numéral $\overline{n} = SSS\dots S0$ représentant n dans $\mathcal{L}_0$.

Exemple.
Le prédicat $R(x)$: « x est un entier pair » est représentable grâce à la formule suivante :

$$\mathcal{P}(x) \ : \quad \exists y \ \ x = \overline{2} \times y$$

La vérification de cette propriété pour un entier précis est un processus fini, qui peut être traduit pas à pas sous forme de preuve au sein de $\mathcal{PA}$.

1.2. Théorème

Voici le théorème qui sera utile pour la suite des chapitres. Il sera appliqué pour le prédicat « est_preuve(N, n) » qui détermine si les formules du super-nombre de Gödel N forment bien une preuve de la formule de nombre de Gödel n.

Théorème 1.
Si R est un prédicat récursif primitif, alors R est représentable.

On rappelle qu'un prédicat est récursif primitif si sa fonction caractéristique est récursive primitive. De même, un prédicat sera représentable si sa fonction caractéristique est représentable. Dans la suite, on va donc définir et étudier cette notion plus générale de fonction représentable, indispensable pour démontrer le théorème ci-dessus.

2. Fonctions représentables

2.1. Définition

Définition.
Une fonction $f : \mathbb{N}^p \to \mathbb{N}$ est ***représentable*** s'il existe une formule $\mathcal{P}(\mathbf{x}, y)$ telle que pour tout $\mathbf{n} = (n_1, \ldots, n_p) \in \mathbb{N}^p$, on ait :

$$\mathcal{PA} \quad \vdash \quad \forall y \quad \left[\mathcal{P}(\overline{\mathbf{n}}, y) \leftrightarrow y = \overline{f(\mathbf{n})}\right]$$

Remarques :

- La terminologie complète est « fonction fortement représentable par la théorie $\mathcal{PA}$ ».
- Encore une fois, cette définition fait le lien entre un entier $k \in \mathbb{N}$ et son numéral $\overline{k}$ dans $\mathcal{L}_0$.
- Il faut comprendre la formule $\mathcal{P}$ comme le graphe de f. C'est une façon équivalente de définir la fonction.

Exemple.
La fonction $f : \mathbb{N} \to \mathbb{N}$ définie par $f(x) = 2x + 1$ est représentable. En effet, considérons la formule :

$$\mathcal{P}(x, y) \ : \ y = 2x + 1$$

Il est donc exact (et démontrable dans l'arithmétique de Peano) que $\mathcal{P}(x, y)$ est vérifiée ssi $y = f(x)$. Autrement dit, quel que soit $n \in \mathbb{N}$, $\mathcal{P}(n, y)$ est vrai ssi $y = f(n)$.
Mais attention, on n'a pas tout à fait construit l'expression dans le langage $\mathcal{L}_0$. En effet, l'entier 2 n'existe pas, il faut l'écrire $\overline{2} = SS0$. La bonne formule de $\mathcal{L}_0$ est donc :

$$\mathcal{P}(x, y) \ : \ y = S(SS0 \times x)$$

Et l'équivalence est, quel que soit $n \in \mathbb{N}$:

$$\forall y \quad \left[\mathcal{P}(\overline{n}, y) \leftrightarrow y = \overline{f(n)}\right]$$

(où $\overline{n} = SS\ldots S0$ est le numéral de l'entier n et $\overline{f(n)} = SS\ldots S0$ est le numéral de l'entier $f(n)$).
C'est donc bien un passage du monde des entiers au langage $\mathcal{L}_0$.

2.2. Définition équivalente

Afin de mieux comprendre la définition de fonction représentable, voici une définition équivalente.

Définition.
Une fonction $f : \mathbb{N}^p \to \mathbb{N}$ est une fonction ***représentable*** s'il existe une formule $\mathcal{P}(\mathbf{x}, y)$ telle que pour tout $\mathbf{n} \in \mathbb{N}^p$ et en posant $m = f(\mathbf{n})$, on ait :

1. $\mathcal{PA} \vdash \mathcal{P}(\overline{\mathbf{n}}, \overline{m})$ *exactitude*
2. $\mathcal{PA} \vdash \forall y \ \left[\mathcal{P}(\overline{\mathbf{n}}, y) \rightarrow y = \overline{m}\right]$ *unicité*

L'exactitude signifie que la formule $\mathcal{P}$ permet bien de retrouver la valeur m de $f(\mathbf{n})$. L'unicité est indispensable, car une fonction ne peut bien sûr prendre qu'une seule valeur.
Justifions que cette seconde définition est équivalente à la première. Dans la définition originale, on a une équivalence. L'unicité est exactement l'implication $\rightarrow$. Partons de la définition originale, en spécialisant le $\forall$ en $y = \overline{m}$, on obtient exactement $\mathcal{P}(\overline{\mathbf{n}}, \overline{m})$. Réciproquement, en partant de la seconde définition, on a $\mathcal{P}(\overline{\mathbf{n}}, \overline{m})$ ce qui implique que $y = \overline{m} \rightarrow \mathcal{P}(\overline{\mathbf{n}}, y)$. On introduit alors le $\forall y$ pour obtenir l'implication inverse ($\leftarrow$).
On verra en exercice des contre-exemples de formules ne satisfaisant pas l'exactitude ou l'unicité.

2.3. Prédicat représentable

Revenons sur la notion de prédicat représentable.

Proposition 1.
Un prédicat $R(\mathbf{x})$ est un prédicat représentable si et seulement si sa fonction caractéristique $\chi_R(\mathbf{x})$ est une fonction représentable.

On rappelle que par définition $\chi_R(\mathbf{x}) = 1$ si $R(\mathbf{x})$ est vrai, et $\chi_R(\mathbf{x}) = 0$ sinon.
Pour simplifier l'écriture des preuves, ici et souvent dans la suite, on écrira les preuves avec le minimum de variables possibles, par exemple avec $p = 1$. Pour obtenir la preuve générale, il suffirait bien sûr de changer x en $\mathbf{x}$ et n en $\mathbf{n}$. De plus, on écrira à la fois n pour l'entier et son numéral $\overline{n}$ car il n'y a pas d'ambiguïté.
La preuve suivante est assez technique et peut être passée en première lecture.

Démonstration.

- $\Rightarrow$
  Hypothèse : χ_R est une fonction représentable.
  But : R est un prédicat représentable.
  Notons $\mathcal{P}(x, y)$ la formule issue de l'hypothèse et définissons $\mathcal{P}_1(x)$ par $\mathcal{P}(x, 1)$. Dans la formule issue de l'hypothèse « χ_R est une fonction représentable », on pose $y = 1$ et on obtient l'équivalence :
  $$\mathcal{PA} \vdash \left[\mathcal{P}(n, 1) \leftrightarrow 1 = \chi_R(n)\right]$$
  — Si $R(n)$ est vrai, alors $\chi_R(n) = 1$. Donc $1 = \chi_R(n)$ est vrai. Donc, par l'équivalence, $\mathcal{P}_1(n)$ est vrai aussi. Plus précisément, $\mathcal{PA} \vdash (\mathcal{P}_1(n) \leftrightarrow (1 = 1))$. Donc, par définition de l'implication, on obtient : $\mathcal{PA} \vdash \mathcal{P}_1(n)$.
  — Si $R(n)$ est faux, alors $\chi_R(n) = 0$, donc, par l'équivalence, c'est $\neg\mathcal{P}_1(n)$ qui est vrai.
  Ainsi, le prédicat R est bien représentable par la formule $\mathcal{P}_1$.
- $\Leftarrow$
  Hypothèse : R est un prédicat représentable.
  But : χ_R est une fonction représentable.
  L'hypothèse nous donne une formule $\mathcal{P}_0(x)$ associée au prédicat R. On construit alors la formule $\mathcal{P}(x, y)$ pour la fonction χ_R ainsi :
  $$\mathcal{P}(x, y) \ : \quad \big(\mathcal{P}_0(x) \wedge (y = 1)\big) \vee \big(\neg\mathcal{P}_0(x) \wedge (y = 0)\big)$$
  Nous devons vérifier que cette formule fait bien de χ_R une fonction représentable.
  Exactitude. Fixons un entier n et posons $m = \chi_R(n)$, qui vaut donc 0 ou 1.
  — Si $m = 1$, alors $R(n)$ est vrai, donc par hypothèse $\mathcal{PA} \vdash \mathcal{P}_0(n)$. Donc $\mathcal{PA} \vdash \mathcal{P}_0(n) \wedge (m = 1)$, c'est-à-dire $\mathcal{PA} \vdash \mathcal{P}(n, m)$.
  — Si $m = 0$, alors $R(n)$ est faux et on obtient de manière similaire $\mathcal{PA} \vdash \mathcal{P}(n, m)$.

Dans tous les cas, on a bien vérifié la condition d'exactitude.

Unicité. Pour l'unicité, on doit justifier que, quel que soit y : $\mathcal{PA} \vdash \mathcal{P}(n,y) \to y = m$.
— Cas $y \neq 0$ et $y \neq 1$. Alors $\mathcal{P}(n,y)$ est faux, donc l'implication est vérifiée.
— Cas $y = 1$.
  — Si $R(n)$ est vrai, alors $\mathcal{PA} \vdash \mathcal{P}_0(n)$. Donc $\mathcal{PA} \vdash \mathcal{P}_0(n) \wedge (y = 1)$. Ainsi $\mathcal{PA} \vdash \mathcal{P}_0(n) \wedge (y = 1) \wedge (y = \chi_R(n))$, donc $\mathcal{PA} \vdash (\mathcal{P}(n,1) \to y = m)$ (car les deux assertions de part et d'autre de l'implication sont vraies).
  — Si $R(n)$ est faux, alors $\mathcal{PA} \vdash \neg\mathcal{P}_0(n)$ donc $\mathcal{P}(x,y)$ est faux, donc l'implication est vraie.
— Cas $y = 0$. Similaire au cas précédent.

□

2.4. Exemples importants

Le but du chapitre est de prouver que toutes les fonctions récursives primitives sont représentables. La stratégie a déjà été rencontrée : il suffit de le démontrer pour les trois fonctions de base, puis de montrer que cette propriété est stable par les opérations de composition et de récurrence.

Proposition 2.
Les fonctions de base Z, S et les projections P_i^p sont des fonctions représentables.

Démonstration. Il n'y a pas vraiment grand-chose à prouver. Il s'agit juste de donner la formule $\mathcal{P}(\mathbf{x}, y)$ qui convient.

- Prenons l'exemple de la fonction successeur S, définie par $S(n) = n+1$. On pose $\mathcal{P}(x,y) \ : \ y = S(x)$ (qui est une formule de $\mathcal{L}_0$).
  Pour $n \in \mathbb{N}$ fixé, et en notant $m = S(n) = n+1$, on a bien :
  $$\mathcal{P}(\overline{n}, y) \quad \leftrightarrow \quad y = S(\overline{n}) \quad \leftrightarrow \quad y = \overline{S(n)} \quad \leftrightarrow \quad y = \overline{m}$$
  Ces équivalences sont vraies dans tout système logique, pas besoin des axiomes de Peano. Ainsi S est une fonction représentable.
- Pour la fonction zéro Z, la formule est $\mathcal{P}(x,y) \ : \ y = 0$.
- Pour la projection P_i^p, la formule est $\mathcal{P}(\mathbf{x}, y) \ : \ y = x_i$.

□

Proposition 3.
Soient $f_1, \ldots, f_q : \mathbb{N}^p \to \mathbb{N}$ et $g : \mathbb{N}^q \to \mathbb{N}$ des fonctions représentables. Alors $h = g(f_1, \ldots, f_q)$ est aussi représentable.

Ainsi l'opération de composition est stable pour la propriété de représentabilité. Si on prouve qu'il en est de même pour l'opération de récurrence, on aura le résultat souhaité. Cependant, cette dernière stabilité sera beaucoup plus difficile à obtenir et nous occupera le reste du chapitre.
Pour l'instant, on se concentre sur la preuve de la proposition ci-dessus. Comme on l'a déjà dit, on donnera la preuve dans un cadre minimaliste, mais il est intéressant de commenter la formule générale qui fait de h une fonction représentable :

$$\mathcal{P}(\mathbf{x}, y) \ : \ \exists w_1 \ldots \exists w_q \quad \mathcal{P}_0(w_1, \ldots, w_q, y) \wedge \mathcal{P}_1(\mathbf{x}, w_1) \wedge \cdots \wedge \mathcal{P}_q(\mathbf{x}, w_q)$$

où $\mathcal{P}_0$ est la formule associée à la fonction représentable g et $\mathcal{P}_i$ la formule de f_i $(i = 1, \ldots, q)$. Le rôle des variables w_i est très intéressant : il s'agit de mémoriser le résultat intermédiaire du calcul $w_i = f_i(\mathbf{n})$, afin de pouvoir l'injecter dans g pour calculer $h(\mathbf{n}) = g(w_1, \ldots, w_q) = g(f_1(\mathbf{n}), \ldots, f_q(\mathbf{n}))$.

Démonstration. On fait la preuve seulement dans le cas $p = 1$ et $q = 1$. Soit $f_1 : \mathbb{N} \to \mathbb{N}$ une fonction représentable, représentée par la formule $\mathcal{P}_1(x,w)$. Soit $g : \mathbb{N} \to \mathbb{N}$ une fonction représentable, représentée par la formule $\mathcal{P}_0(w,y)$. On veut montrer que la composition $h = g \circ f_1 : \mathbb{N} \to \mathbb{N}$ est représentable.

On définit la formule :

$$\mathcal{P}(x,y) \ : \quad \exists w \ \ \mathcal{P}_0(w,y) \wedge \mathcal{P}_1(x,w)$$

qui va représenter la fonction h. Pour $n \in \mathbb{N}$, posons $k = f_1(n)$ et $m = g(k) = g(f_1(n)) = h(n)$.

- **Exactitude.** Par hypothèse, on a $\mathcal{PA} \vdash \mathcal{P}_0(k,m)$ et $\mathcal{PA} \vdash \mathcal{P}_1(n,k)$. Donc $\mathcal{PA} \vdash \exists w \left(\mathcal{P}_0(w,m) \wedge \mathcal{P}_1(n,w)\right)$ (il suffit de prendre $w = k$). Donc $\mathcal{PA} \vdash \mathcal{P}(n,m)$.
- **Unicité.** $\mathcal{PA}$ prouve par hypothèse les deux implications suivantes :

$$\forall w \ \ \left(\mathcal{P}_1(n,w) \to w = k\right)$$

$$\text{et} \quad \forall y \ \left(\mathcal{P}_0(k,y) \to y = m\right)$$

Ainsi, supposons qu'on ait $\mathcal{P}(n,y)$. Donc il existe w tel que $\mathcal{P}_0(w,y)$ et $\mathcal{P}_1(n,w)$. Comme on a $\mathcal{P}_1(n,w)$, alors $w = k$. On a donc $\mathcal{P}_0(k,y)$, ce qui implique $y = m$. Ainsi on a bien vérifié la condition d'unicité : $\mathcal{P}(n,y) \to y = m$.

□

Proposition 4.
La fonction reste, qui à a et b associe le reste $\text{reste}(b,a)$ de la division euclidienne de b par a, est représentable.

Démonstration. La formule est :

$$\mathcal{P}(a,b,r) \ : \quad \left[a = 0 \wedge b = r\right] \vee \left[\exists q \leqslant b \quad (b = qa + r) \wedge (r < a)\right]$$

On prouve l'exactitude et l'unicité. En effet l'arithmétique de Peano prouve l'existence et l'unicité de la division euclidienne. En fait, ce serait aussi vrai dans l'arithmétique $\mathcal{Q}$ de Robinson, donc sans utiliser la récurrence.
Remarque : « $\exists q \leqslant b \ \ \mathcal{P}$ » signifie « $\exists q \ \ (q \leqslant b) \wedge \mathcal{P}$ ». □

3. La fonction β de Gödel

Le but de cette section est d'étudier la fonction β de Gödel. C'est une construction astucieuse qui permet de stocker une liste quelconque d'entiers $(n_1, \ldots, n_l)$ à l'aide de seulement deux entiers a et b.
Cette construction joue le rôle de mémoire. En effet, pour calculer une suite par récurrence, il faut calculer $u_1, u_2, \ldots, u_{l-1}$ avant de pouvoir calculer u_l.

3.1. Théorème

Théorème 2.
Il existe une fonction $\beta : \mathbb{N}^3 \to \mathbb{N}$ qui est récursive primitive, représentable et telle que, pour tout entier l et toute suite $(n_1, \ldots, n_l)$ d'entiers, il existe $a, b \in \mathbb{N}$ tels que :

$$\beta(a,b,i) = n_i \qquad \text{pour } i = 1, \ldots, l.$$

La fonction β est au final assez simple, car $\beta(a,b,i)$ sera le reste de la division euclidienne de b par $a \times i + 1$. Ce qui est délicat, c'est d'expliquer l'existence de ces a et b. Pour cela, on aura besoin d'un théorème bien connu d'arithmétique.

3.2. Théorème des restes chinois

Théorème 3 (Théorème des restes chinois)**.**
Soient $a_1, \ldots, a_l$ des entiers premiers entre eux deux à deux. Soient $n_1, \ldots, n_l$ des entiers quelconques. Il existe $x \in \mathbb{N}$ tel que pour tout $i = 1, \ldots, l$, on ait :

$$x \equiv n_i \pmod{a_i}$$

Pour deux entiers, cela s'énonce ainsi :

Théorème 4.
Soient a et b des entiers premiers entre eux. Pour tous $\alpha, \beta \in \mathbb{N}$, il existe $x \in \mathbb{N}$ tel que :

$$\begin{cases} x \equiv \alpha \pmod{a} \\ x \equiv \beta \pmod{b} \end{cases}$$

D'un point de vue algébrique, on a même un isomorphisme de groupes :

$$\mathbb{Z}/ab\mathbb{Z} \simeq \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$$

Démonstration. On définit :

$$\phi : \mathbb{Z}/ab\mathbb{Z} \longrightarrow \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$$

par $\phi(\overline{x}^{ab}) = (\overline{x}^{a}, \overline{x}^{b})$ où l'on note $\overline{x}^{m}$ la classe de l'entier x modulo m.

- ϕ est bien définie. En effet, si $\overline{x}^{ab} = \overline{y}^{ab}$ alors $x = y + kab$, donc $\overline{x}^{a} = \overline{y}^{a}$ et $\overline{x}^{b} = \overline{y}^{b}$. Donc $\phi(\overline{x}^{ab}) = \phi(\overline{y}^{ab})$.
- ϕ est un morphisme de groupes : $\phi(\overline{x}^{ab} + \overline{y}^{ab}) = \phi(\overline{x}^{ab}) + \phi(\overline{y}^{ab})$.
- ϕ est injective. Si $\phi(\overline{x}^{ab}) = (\overline{0}^{a}, \overline{0}^{b})$ alors $\overline{x}^{a} = \overline{0}^{a}$ donc $x = k_1 a$, et $\overline{x}^{b} = \overline{0}^{b}$ donc $x = k_2 b$. Ainsi $k_1 a = k_2 b$. En particulier, b divise $k_1 a$. Or, par hypothèse, a et b sont premiers entre eux, donc d'après le lemme de Gauss b divise k_1, donc $k_1 = k_3 b$. Ainsi $x = k_1 a = k_3 ba$. Donc x est un multiple de ab et la classe $\overline{x}^{ab}$ est $\overline{0}^{ab}$. Ainsi l'application ϕ est injective.
- Comme les cardinaux des ensembles de départ et d'arrivée sont égaux (à ab), l'application est bijective.
- Ainsi, pour (α, β) donné, il existe un antécédent x par ϕ, ce qui dit exactement que :

$$\begin{cases} x \equiv \alpha \pmod{a} \\ x \equiv \beta \pmod{b} \end{cases}$$

□

3.3. Preuve

Construction de a et b. On se donne $n_1, \ldots, n_l \in \mathbb{N}$. On choisit $m \in \mathbb{N}$ tel que $m \geqslant l$ et $m! \geqslant n_i$ $(i = 1, \ldots, l)$. On pose $a = m!$. L'autre entier b sera construit à l'aide du théorème des restes chinois.

- Les entiers $ai+1$ sont premiers entre eux deux à deux. En effet, si p est un nombre premier qui divise $ai+1$ et $aj+1$ (avec $i < j$), alors par différence $p|a(j-i)$. Donc $p|a$ ou $p|(j-i)$. Or $j - i < l \leqslant m$ et $a = m!$ donc dans tous les cas, $p \leqslant m$. Cela entraîne $p|m!$ (c'est-à-dire $p|a$), mais comme $p|ai+1$ alors $p|1$. Contradiction.
- On a $ai + 1 > n_i$ car $n_i \leqslant m! = a$ $(i = 1, \ldots, l)$.
- Par le théorème des restes chinois, il existe $b \in \mathbb{N}$ tel que :

$$b \equiv n_i \pmod{ai+1} \quad \text{quel que soit } i = 1, \ldots, l$$

Construction de β. Cela justifie la construction de β : $\beta(a, b, i)$ est le reste de la division euclidienne de b par $ai + 1$. En effet, ce reste est caractérisé par $b \equiv n_i \pmod{ai+1}$ et $n_i < ai + 1$ comme ci-dessus.

β est récursive primitive. La première partie nous assure que ce codage existe même s'il est un peu compliqué. Cependant, le décodage reste simple :

$$\beta(a, b, i) = \text{reste}(b, ai + 1)$$

Ce reste est bien donné par une fonction récursive primitive. En effet :

$$\text{reste}(b, a) = \min\{r \leqslant b \mid \exists q \leqslant b \quad b = qa + r\}$$

La condition « $\exists q \leqslant b \quad b = qa + r$ » est bien un prédicat récursif primitif car la quantification est bornée. Donc le reste l'est aussi car il est obtenu par minimisation bornée. La borne sur le minimum n'est pas

naturelle, car en général $r < a$ dans une division euclidienne, mais cette borne b permet d'englober le cas exceptionnel $a = 0$.

β est représentable. La formule à considérer est :

$$\mathcal{B}(a,b,i,r) \ : \quad \big(\exists q \leqslant b \quad b = q(ai+1)+r\big) \wedge \big(r < ai+1\big)$$

Autrement dit, r est défini comme le reste de la division euclidienne de b par $ai+1$. On a vu que la fonction reste est représentable.
Abus de notation : a, b, i, r ne sont pas des variables du langage $\mathcal{L}_0$, elles devraient être remplacées par $x_1, x_2, \ldots$, mais les formules deviendraient moins lisibles.

Note. L'intérêt du codage par la fonction β est de n'utiliser que des additions et des multiplications, qui sont les seules opérations autorisées par le langage $\mathcal{L}_0$.
Sinon, pour coder une suite d'entiers, on aurait pu choisir le codage $2^{n_1}3^{n_2}5^{n_3}\ldots$ avec les nombres premiers. Mais dans $\mathcal{L}_0$, on n'a pas encore le droit de calculer des puissances : par exemple, pour calculer x^n, on calcule successivement $x, x^2, x^3 \ldots$ jusqu'à x^n. On a donc besoin de l'opération de récurrence, autrement dit, il faut pouvoir mettre en mémoire les résultats intermédiaires. Mais on ne sait pas encore que l'opérateur de récurrence préserve la représentabilité, puisque c'est exactement ce que l'on souhaite démontrer.
Une fois qu'on aura la représentabilité des fonctions récursives primitives grâce à la fonction β, on pourra utiliser des listes (finies) de nombres pour le codage de Gödel dans le langage $\mathcal{L}_0$.

4. Les fonctions récursives primitives sont représentables

Attaquons-nous à l'objectif final du chapitre.

Théorème 5.
Toute fonction $f : \mathbb{N}^p \to \mathbb{N}$ qui est récursive primitive est représentable.

4.1. Opération de récurrence

Pour prouver que les fonctions récursives primitives sont représentables, il reste donc à démontrer la stabilité de la représentabilité par l'opération de récurrence.

Proposition 5.
Soient $g : \mathbb{N}^p \to \mathbb{N}$ et $h : \mathbb{N}^{p+2} \to \mathbb{N}$ deux fonctions représentables. La fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$ définie par l'initialisation g et la relation de récurrence h est une fonction représentable.

Cette proposition entraîne le théorème ci-dessus, car on a déjà montré que les trois fonctions de base Z, S, P_i^p étaient représentables et que la composition préservait la représentabilité.

4.2. La formule

On va faire la preuve dans le cas $p = 1$ afin d'avoir l'écriture la plus simple possible.

- $g : \mathbb{N} \to \mathbb{N}$ représentable par $\mathcal{P}_g(x,u)$.
- $h : \mathbb{N}^3 \to \mathbb{N}$ représentable par $\mathcal{P}_h(x,y,u,v)$.
- $f : \mathbb{N}^2 \to \mathbb{N}$ définie par $f(x,0) = g(x)$ et $f(x,y+1) = h(x,y,f(x,y))$.

Le but est de trouver une formule qui représente f. Cette formule $\mathcal{P}_f(x,y,z)$ est : « il existe une suite finie (encodée par des entiers a et b), dont le premier terme est $z_0 = g(x)$, chaque transition respecte la relation de récurrence $z_{i+1} = h(x,i,z_i)$ et dont le dernier terme est $z_y = z$ ».

Il faut transcrire cette phrase dans $\mathcal{L}_0$. Pour cela on utilise la fonction β de Gödel. On va faire une modification mineure afin d'avoir une bonne indexation : on suppose maintenant que $\beta(a,b,i)$ encode une suite $n_0, n_1, n_2, \ldots$ (les indices partent de 0). On a vu que la fonction β est représentable, on note $\mathcal{B}(a,b,i,r)$ la formule correspondante. On rappelle que la fonction β a pour rôle de mémoriser une liste de nombres qui seront ici les termes z_i de la suite.
Voici la formule $\mathcal{P}_f$ dans le langage $\mathcal{L}_0$:

$$\mathcal{P}_f(x,y,z) \ : \quad \exists a\ \exists b \quad \text{Init}(a,b,x) \wedge \text{Trans}(a,b,x,y) \wedge \mathcal{B}(a,b,y,z)$$

On fixe x (cette variable x ne sert en fait à rien), on pose $z_0 = g(x)$, $z_1 = h(x,0,z_0)$, $z_2 = h(x,1,z_1), \ldots$ et le but est de calculer z_y afin de vérifier que c'est bien la valeur $z = f(x,y)$.
Avec $\mathcal{B}$ la formule associée à la fonction β de Gödel, les trois parties de la formule $\mathcal{P}_f$ sont définies ainsi :

- $\text{Init}(a,b,x)$: $\quad \exists u \quad \mathcal{B}(a,b,0,u) \wedge \mathcal{P}_g(x,u)$
  Ainsi on a $u = \beta(a,b,0)$ (le premier terme dans la mémoire) et en même temps $u = g(x)$, donc on a $z_0 = g(x) = \beta(a,b,0)$.
- $\text{Trans}(a,b,x,y)$: $\quad \forall i < y \quad \exists u\ \exists v \quad \mathcal{B}(a,b,i,u) \wedge \mathcal{B}(a,b,i+1,v) \wedge \mathcal{P}_h(x,i,u,v)$
  Les conjonctions donnent : $u = \beta(a,b,i)$ et $v = \beta(a,b,i+1)$ et $v = h(x,i,u)$. Ainsi si $z_i = \beta(a,b,i) = u$ alors $z_{i+1} = h(x,i,z_i) = h(x,i,u) = v = \beta(a,b,i+1)$. Ainsi par le principe de récurrence on a $z_i = \beta(a,b,i)$ pour tout $i \leqslant y$.
- $\mathcal{B}(a,b,y,z)$ signifie que $z = \beta(a,b,y)$ c'est-à-dire $z = z_y$.

4.3. Exactitude

On considère n (pour la variable x), l (pour la variable y) et $m = f(n,l)$ (pour la variable z). L'exactitude signifie que $\mathcal{PA} \vdash \mathcal{P}_f(n,l,m)$.
Cette fois on note $z_0 = g(n)$ et $z_{i+1} = h(n,i,z_i)$ pour $i = 0, \ldots, l-1$. But final : montrer que $m = z_l$.
On sait qu'il existe a, b qui encodent la suite et vérifient $\beta(a,b,i) = z_i$ pour $i = 0, \ldots, l$.

- On a $\beta(a,b,0) = z_0$, on a donc $\mathcal{PA} \vdash \mathcal{B}(a,b,0,z_0) \wedge \mathcal{P}_g(n,z_0)$. Donc en posant $u = z_0$, on a $\mathcal{PA} \vdash \text{Init}(a,b,n)$.
- Ensuite pour z_{i+1}, on pose $u = z_i$ et $v = z_{i+1}$, on a :
  $$\mathcal{B}(a,b,i,z_i) \wedge \mathcal{B}(a,b,i+1,z_{i+1}) \wedge \mathcal{P}_h(n,i,z_i,z_{i+1})$$
  Ceci pour $i = 0, \ldots, l-1$, donc $\mathcal{PA} \vdash \text{Trans}(a,b,n,l)$.
- Enfin $z_l = h(n,l-1,z_{l-1})$ donc $z_l = \beta(a,b,l) = f(n,l) = m$ et ainsi on a $\mathcal{B}(a,b,l,m)$.

4.4. Unicité

Il s'agit de démontrer que :

$$\mathcal{PA} \quad \vdash \quad \forall z \ \ \mathcal{P}_f(n,l,z) \to z = m$$

c'est-à-dire $z = z_l$.
Par récurrence sur l :

- Au rang 0 : La formule $\mathcal{P}_f(n,0,z)$ (avec $m = f(n,0) = g(n) = z_0$) se réduit à :
  $$\exists a\ \exists b \quad \text{Init}(a,b,n) \wedge \mathcal{B}(a,b,0,z)$$
  (car la condition « $\forall i < 0$ » est vide, donc le prédicat Trans est vrai). Donc $z_0 = \beta(a,b,0) = z$, d'où $z = m$ comme voulu.
- Hérédité. Fixons i avec $0 \leqslant i < l$ et supposons :
  $$\forall z \quad \mathcal{P}_f(n,i,z) \to z = z_i$$
  Soit z quelconque. Supposons $\mathcal{P}_f(n,i+1,z)$ vraie. Dans la formule $\mathcal{P}_f(n,i+1,z)$ est incluse la sous-formule $\mathcal{P}_f(n,i,v)$ donc par hypothèse de récurrence $v = z_i$. Mais on a en plus :
  $$\mathcal{B}(a,b,i,u) \wedge \mathcal{B}(a,b,i+1,v) \wedge \mathcal{P}_h(n,i,u,v) \wedge \mathcal{B}(a,b,i+1,z)$$

ce qui prouve dans l'ordre : $u = \beta(a,b,i) = z_i$, $v = \beta(a,b,i+1)$, $v = h(n,i,u) = h(n,i,z_i) = z_{i+1}$ et $z = \beta(a,b,i+1)$. On en déduit donc que $z = z_{i+1}$.

- Par le principe de récurrence (dans $\mathcal{PA}$), l'assertion est vraie pour tous les $i \leqslant l$, donc l'unicité est démontrée.

Remarque. Il est possible de tout faire dans l'arithmétique $\mathcal{Q}$ de Robinson, car en fait notre récurrence se limite à un nombre fini d'étapes (car l est fixé).
Cela termine la preuve que, pour les fonctions récursives primitives, l'opération de récurrence préserve la propriété d'être une fonction représentable.

4.5. Application aux prédicats

On savait que les trois fonctions récursives primitives de base étaient représentables, et que l'opération de composition préservait la représentabilité. On vient de terminer la preuve difficile que pour l'opération de récurrence on a aussi la stabilité de la représentabilité. Par définition de ce qu'est une fonction récursive primitive, on en déduit le Théorème 5 « Toutes les fonctions récursives primitives sont représentables. »
En particulier, on sait qu'un prédicat est représentable ssi sa fonction caractéristique l'est. Donc on a le Théorème 1 annoncé au tout début « Tout prédicat récursif primitif est représentable. »

Chapitre 17

Diagonalisation

Nous rassemblons tous les outils construits à présent afin de démontrer l'étape cruciale de l'incomplétude : le lemme de diagonalisation.

1. Résumé des épisodes précédents

L'objectif final n'est maintenant plus très loin : démontrer le théorème d'incomplétude. Nous sommes ici dans un chapitre encore technique avec un lemme crucial qui nécessite tous les chapitres précédents. Avant de nous plonger dans les détails, revenons brièvement sur toutes les notions utiles.

1.1. Arithmétique de Peano

- L'arithmétique de Peano est basée sur le langage $\mathcal{L}_0 = (0, S, +, \times)$, dans lequel on construit une théorie $\mathcal{PA}$ (les axiomes de Peano).
- Les entiers naturels $\mathbb{N}$ sont un modèle de cette théorie : les axiomes de Peano sont satisfaits sur $\mathbb{N}$.
- Pour $n \in \mathbb{N}$, son numéral $\overline{n} = SS\ldots S0$ est son écriture dans $\mathcal{L}_0$ (il y a exactement n symboles S).

1.2. Codage de Gödel

- À toute formule de $\mathcal{L}_0$, on associe un entier dans $\mathbb{N}$: son nombre de Gödel. On le note aussi $n = \ulcorner \mathcal{P} \urcorner$.
- Dans $\mathcal{L}_0$ son numéral est donc $\overline{n} = \overline{\ulcorner \mathcal{P} \urcorner}$, que l'on notera souvent aussi par $\ulcorner \mathcal{P} \urcorner$ (c'est un peu abusif, mais il ne peut pas y avoir de confusion dans les formules).
- On a déjà donné un aperçu de ce qu'est la diagonalisation d'une formule lors de la preuve informelle de l'incomplétude, l'idée est de considérer $\mathcal{G} = \mathcal{P}(\ulcorner \mathcal{G} \urcorner)$.
- Le but du chapitre est de formaliser de façon rigoureuse cette formule $\mathcal{G}$.

1.3. Fonctions récursives primitives

- On rappelle que les fonctions récursives primitives sont des fonctions $f : \mathbb{N}^p \to \mathbb{N}$ facilement calculables.
- Le point important pour ce chapitre est que la fonction substitution$(n, v, k) : \mathbb{N}^3 \to \mathbb{N}$ est récursive primitive, n désigne le nombre de Gödel d'une formule $\mathcal{P}$, v le code d'une variable x, et k est la valeur que l'on souhaite appliquer à x. La fonction renvoie donc $\ulcorner \mathcal{P}(k) \urcorner \in \mathbb{N}$.

1.4. Fonctions représentables

- Toute fonction récursive primitive est représentable dans $\mathcal{PA}$. On peut alors traduire tous nos concepts mathématiques dans $\mathcal{L}_0$.
- Pour $f : \mathbb{N} \to \mathbb{N}$ récursive primitive, cela signifie qu'il existe une formule $\mathcal{P}(x, y)$ de $\mathcal{L}_0$ telle que pour tout $n \in \mathbb{N}$, on ait une preuve dans $\mathcal{PA}$ de l'équivalence :
$$\forall y \quad \mathcal{P}(\overline{n}, y) \quad \leftrightarrow \quad y = \overline{f(n)}$$
- Autrement dit, on caractérise la valeur $f(n)$ de façon unique dans $\mathcal{L}_0$ grâce à $\mathcal{P}$.

2. Lemme de diagonalisation

Attaquons-nous à la clé de voûte de la preuve du théorème d'incomplétude.

2.1. Énoncé

Lemme 1 (de diagonalisation)**.**
Pour toute formule $\mathcal{P}(x)$ (où x est la seule variable libre), il existe une formule fermée $\mathcal{G}$ telle que :

$$\boxed{\mathcal{PA} \quad \vdash \quad \left(\mathcal{G} \quad \leftrightarrow \quad \mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})\right)}$$

- Ainsi $\mathcal{G}$ apparaît deux fois dans l'énoncé, une fois en tant que formule logique, l'autre via son nombre de Gödel. La formule $\mathcal{G}$ vérifie donc une propriété sur elle-même.
- On peut considérer le lemme de diagonalisation comme un théorème de point fixe : au lieu d'obtenir $f(x) = x$, on a $\mathcal{P}(\ulcorner \mathcal{G} \urcorner) \leftrightarrow \mathcal{G}$.
- L'équivalence est la façon syntaxiquement correcte de dire que $\mathcal{G}$ et $\mathcal{P}(\ulcorner \mathcal{G} \urcorner)$ sont des énoncés équivalents (mais ce ne sont pas des formules égales dans $\mathcal{L}_0$). Par contre on verra dans la preuve ci-dessous que $\mathcal{G}$ est exactement défini par $\mathcal{G} = \mathcal{H}(\ulcorner \mathcal{H} \urcorner)$.

2.2. Preuve

Diagonalisation sur les entiers

On va construire une fonction diag : $\mathbb{N} \to \mathbb{N}$ récursive primitive.

- Pour $\mathcal{Q}(x)$, on sait lui associer son nombre de Gödel $n = \ulcorner \mathcal{Q} \urcorner \in \mathbb{N}$. On va s'intéresser à l'opération inverse.
- Pour $n \in \mathbb{N}$, on sait décider si cet entier correspond à une formule correctement formée $\mathcal{Q}(x)$ de $\mathcal{L}_0$, ayant x comme seule variable libre. Si ce n'est pas le cas, on pose $\text{diag}(n) = 0$.
- Si n représente la formule $\mathcal{Q}(x)$, alors on peut évaluer $\mathcal{Q}$ en n, c'est-à-dire écrire la substitution $\mathcal{Q}(\overline{n})$. C'est une formule de $\mathcal{L}_0$, qui a pour nombre de Gödel $\ulcorner \mathcal{Q}(\overline{n}) \urcorner$. On définit alors l'entier :
$$\text{diag}(n) = \ulcorner \mathcal{Q}(\overline{n}) \urcorner$$
C'est-à-dire $\text{diag}(n) = \ulcorner \mathcal{Q}(\ulcorner \mathcal{Q} \urcorner) \urcorner$.
- Résumons la construction par un schéma :
$$n \in \mathbb{N} \xrightarrow{\text{inversion du nb de Gödel}} \mathcal{Q} \in \mathcal{L}_0 \xrightarrow{\text{substitution}} \mathcal{Q}(\overline{n}) \in \mathcal{L}_0 \xrightarrow{\text{nb de Gödel}} \text{diag}(n) = \ulcorner \mathcal{Q}(\overline{n}) \urcorner \in \mathbb{N}$$
- Pour une formule $\mathcal{Q}(x)$ de nombre de Gödel $n = \ulcorner \mathcal{Q} \urcorner$, le calcul de $\mathcal{Q}(k)$ se fait par la fonction substitution$(n, \ulcorner x \urcorner, k)$ qui, pour la formule $\mathcal{Q}$ de nombre de Gödel n, remplace les occurrences de la variable x par $\overline{k}$, puis renvoie le nombre de Gödel de la formule obtenue, c'est-à-dire $\ulcorner \mathcal{Q}(\overline{k}) \urcorner$.
Remarque : dans toute la suite la variable à substituer sera toujours la variable x.

- Décider si un entier n désigne une formule $\mathcal{Q}(x)$ correctement formée ayant x pour seule variable libre est une opération récursive primitive. La substitution l'est aussi (voir les chapitres « Codage de Gödel » et « Fonctions récursives primitives »). Ainsi la fonction $\text{diag} : \mathbb{N} \to \mathbb{N}$ est aussi une fonction récursive primitive.

Diagonalisation dans le langage

- Toute fonction récursive primitive est représentable (voir le chapitre « Fonctions représentables »), donc pour notre fonction $\text{diag} : \mathbb{N} \to \mathbb{N}$, il existe une formule $\mathcal{D}\text{iag}(x, y)$ de $\mathcal{L}_0$ telle que, quel que soit $n \in \mathbb{N}$ on ait :
$$\mathcal{PA} \quad \vdash \quad \Big(\forall y \;\; \mathcal{D}\text{iag}(\overline{n}, y) \leftrightarrow y = \overline{\text{diag}(n)}\Big)$$

Formule $\mathcal{G}$

- On fixe la formule $\mathcal{P}(x)$ de l'énoncé et on définit une nouvelle formule $\mathcal{H}$ par :
$$\mathcal{H}(x) \;:\quad \exists y \; \mathcal{D}\text{iag}(x, y) \wedge \mathcal{P}(y)$$
Notons p le nombre de Gödel de cette dernière formule : $p = \ulcorner \mathcal{H} \urcorner$.
- On définit la formule de Gödel par :
$$\mathcal{G} = \mathcal{H}(\overline{p})$$
Ainsi par définition, $\mathcal{G}$ désigne « $\mathcal{H}(\overline{p})$ » et donc aussi « $\exists y \; \mathcal{D}\text{iag}(\overline{p}, y) \wedge \mathcal{P}(y)$ ».
- De plus, $\text{diag}(p) = \ulcorner \mathcal{G} \urcorner$. En effet, reprenons le schéma ci-dessus, cette fois pour l'entier p :
$$p \in \mathbb{N} \xrightarrow{\text{inversion du nb de Gödel}} \mathcal{H}(x) \in \mathcal{L}_0 \xrightarrow{\text{substitution}} \mathcal{H}(\overline{p}) \in \mathcal{L}_0 \xrightarrow{\text{nb de Gödel}} \text{diag}(p) = \ulcorner \mathcal{H}(\overline{p}) \urcorner = \ulcorner \mathcal{G} \urcorner \in \mathbb{N}$$

Fin de la preuve

Rappelons que le but est de montrer :
$$\mathcal{PA} \quad \vdash \quad \Big(\mathcal{G} \leftrightarrow \mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})\Big)$$
On se place donc dans la théorie $\mathcal{PA}$. On rappelle que $p = \ulcorner \mathcal{H} \urcorner$, $\mathcal{G} = \mathcal{H}(\overline{p})$ et $\text{diag}(p) = \ulcorner \mathcal{G} \urcorner$.

- **Sens $\to$**

  Supposons $\mathcal{G}$
  donc $\mathcal{H}(\overline{p})$ par définition de $\mathcal{G}$
  donc $\exists y \quad \mathcal{D}\text{iag}(\overline{p}, y) \wedge \mathcal{P}(y)$ par définition de $\mathcal{H}$
  donc $\exists y \quad (y = \overline{\text{diag}(p)}) \wedge (\mathcal{P}(y))$ par la représentabilité de diag par $\mathcal{D}$iag dans $\mathcal{PA}$
  donc $\mathcal{P}(\overline{\text{diag}(p)})$ car l'égalité force la valeur de y
  donc $\mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})$

  Ainsi $\mathcal{G} \to \mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})$.
- **Sens $\leftarrow$**
  Supposons $\mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})$. Il s'agit d'en déduire $\mathcal{G} = \mathcal{H}(\overline{p})$, c'est-à-dire :
$$\exists y \quad \mathcal{D}\text{iag}(\overline{p}, y) \wedge \mathcal{P}(y)$$
  On va montrer que $y = \overline{\ulcorner \mathcal{G} \urcorner}$ convient.
  — D'une part, $y = \overline{\ulcorner \mathcal{G} \urcorner} = \overline{\text{diag}(p)}$, donc $\mathcal{D}\text{iag}(\overline{p}, y)$ est vérifié par la représentabilité.
  — De plus, $\mathcal{P}(y) = \mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})$ est notre hypothèse.
  — Donc, pour ce y, on a bien $\mathcal{D}\text{iag}(\overline{p}, y) \wedge \mathcal{P}(y)$.

Par introduction de $\exists$, on a :

$$\mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner}) \quad \to \quad \big[\exists y \quad \mathcal{D}\mathrm{iag}(\overline{p}, y) \wedge \mathcal{P}(y)\big]$$

ce qui est exactement $\mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner}) \to \mathcal{G}$.
On a terminé la preuve du lemme de diagonalisation, on a montré :

$$\mathcal{PA} \quad \vdash \quad \Big(\mathcal{G} \leftrightarrow \mathcal{P}(\overline{\ulcorner \mathcal{G} \urcorner})\Big)$$

3. Théorème de Tarski

3.1. Motivation

Le reste de ce chapitre n'est pas utile pour la preuve du théorème d'incomplétude. Cependant, c'est une excellente application du lemme de diagonalisation et c'est aussi l'occasion de souligner une nouvelle fois la différence entre le vrai et le démontrable. Le théorème de Tarski est une version puissante du paradoxe du menteur. Je dis la phrase suivante :

« Je suis un menteur. »

- Si je suis effectivement un menteur, alors la phrase est vraie, donc je dis la vérité, donc je ne suis pas un menteur. Contradiction.
- Si je ne suis pas un menteur, alors je dis la vérité, donc d'après ma phrase je suis un menteur. Contradiction à nouveau.

Une variante est :

« Cette phrase est fausse. »

3.2. La vérité n'est pas définissable

De façon informelle, le théorème de Tarski (1933) dit qu'il n'existe pas de formule $\mathcal{V}(x)$ qui détecte les formules vraies, c'est-à-dire que $\mathcal{V}(x)$ serait une formule telle que $\mathcal{V}(\overline{n})$ est vraie exactement lorsque la formule fermée $\mathcal{P}$ de nombre de Gödel n est vraie.
Dans notre contexte, une formule fermée $\mathcal{P}$ est vraie si $\vDash_{\mathbb{N}} \mathcal{P}$, c'est-à-dire si elle est satisfaite dans le modèle standard des entiers naturels $\mathbb{N}$.
Voici l'énoncé précis.

Théorème 1 (Tarski)**.**
Il n'existe pas de formule $\mathcal{V}(x)$ de $\mathcal{L}_0$ telle que pour toute formule fermée $\mathcal{P}$ de $\mathcal{L}_0$ on ait :

$$\vDash_{\mathbb{N}} \mathcal{P} \quad \textit{si et seulement si} \quad \vDash_{\mathbb{N}} \mathcal{V}(\overline{\ulcorner \mathcal{P} \urcorner})$$

Une reformulation est « la vérité n'est pas définissable ». En effet, si on note A l'ensemble des entiers $n = \ulcorner \mathcal{P} \urcorner$ tels que $\vDash_{\mathbb{N}} \mathcal{P}$ (c'est-à-dire les nombres de Gödel des formules vraies), alors il n'existe pas de formule $\mathcal{V}(x)$ telle que $\vDash_{\mathbb{N}} \mathcal{V}(\overline{n})$ pour tous les $n \in A$, et uniquement ceux-là. On renvoie au paragraphe « Ensembles définissables » du chapitre « Arithmétique de Peano » pour cette notion de « définissable ».
Par contre, on verra dans le chapitre suivant que « être démontrable » est définissable.

3.3. Preuve

Par l'absurde, supposons qu'une telle formule $\mathcal{V}(x)$ existe. On applique le lemme de diagonalisation à $\neg\mathcal{V}(x)$ (à la place de $\mathcal{P}(x)$ dans l'énoncé du lemme). Ainsi il existe une formule fermée $\mathcal{G}$ telle que :

$$\mathcal{PA} \quad \vdash \quad \Big(\mathcal{G} \leftrightarrow \neg\mathcal{V}(\overline{\ulcorner \mathcal{G} \urcorner})\Big)$$

Cette phrase $\mathcal{G}$ est notre version du menteur qui parle de lui-même. On se place sur la structure $\mathbb{N}$ pour la théorie $\mathcal{PA}$ sur $\mathcal{L}_0$. Par le théorème de validité, si $\mathcal{PA} \vdash \mathcal{Q}$ alors $\vDash_{\mathbb{N}} \mathcal{Q}$: tout ce qui est prouvable est

vrai. Ainsi l'équivalence ci-dessus est vraie :

$$\mathcal{G} \leftrightarrow \neg\mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$$

Vérifions que la phrase $\mathcal{G}$ contredit l'équivalence du Théorème 1 voulue par Tarski (où $\mathcal{G}$ joue le rôle de la phrase $\mathcal{P}$).

- Si $\mathcal{G}$ est vraie, c'est-à-dire $\vDash_{\mathbb{N}} \mathcal{G}$, alors par l'équivalence du lemme de diagonalisation on a aussi $\vDash_{\mathbb{N}} \neg\mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$, c'est-à-dire que $\mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$ est faux, donc on n'a pas $\vDash_{\mathbb{N}} \mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$. Ainsi dans ce cas, $\mathcal{V}$ ne détecte pas que $\mathcal{G}$ est vraie.
- Si $\mathcal{G}$ est fausse (c'est-à-dire on n'a pas $\vDash_{\mathbb{N}} \mathcal{G}$), alors par l'équivalence du lemme de diagonalisation $\neg\mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$ est faux, donc $\mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$ est vrai. Ainsi on a $\vDash_{\mathbb{N}} \mathcal{V}(\overline{\ulcorner\mathcal{G}\urcorner})$. Par conséquent dans ce cas, $\mathcal{V}$ ne détecte pas que $\mathcal{G}$ est fausse.
- Ainsi, sans savoir si $\mathcal{G}$ est vraie ou fausse, $\mathcal{G}$ est quand même un contre-exemple à l'équivalence recherchée.

Chapitre 18
Preuve du théorème d'incomplétude

Nous rassemblons tous les outils obtenus afin de prouver le premier théorème d'incomplétude de Gödel.

Remarque : C'est le bon moment pour relire le chapitre « Un aperçu du théorème d'incomplétude » et aussi le chapitre précédent « Diagonalisation ».

1. L'incomplétude de l'arithmétique de Peano

1.1. Énoncé

Théorème 1.
Si $\mathcal{PA}$ est cohérente, alors $\mathcal{PA}$ est incomplète, c'est-à-dire qu'il existe une formule fermée $\mathcal{G}$ telle que :

$$\mathcal{PA} \nvdash \mathcal{G} \qquad \text{et} \qquad \mathcal{PA} \nvdash \neg\mathcal{G}$$

Ainsi il existe des propositions vraies (dans le modèle standard $\mathbb{N}$) mais non démontrables.

- On rappelle que $\mathcal{PA}$ désigne la théorie de l'arithmétique de Peano, c'est-à-dire les axiomes usuels associés aux entiers naturels.
- L'incomplétude signifie qu'il existe une formule fermée $\mathcal{G}$ qui n'est pas démontrable à partir des axiomes de Peano, et telle que $\neg\mathcal{G}$ n'est pas non plus démontrable.
- Comme $\mathcal{G}$ est une formule fermée, alors dans toute interprétation, par exemple le modèle $\mathbb{N}$ ou un autre modèle, exactement l'une des deux possibilités est vérifiée : soit $\mathcal{G}$ est vraie, soit $\neg\mathcal{G}$ est vraie. L'une des deux est vraie et les deux sont indémontrables.
- Pour le modèle standard $\mathbb{N}$, et pour tout autre modèle, il existe donc des propositions vraies (dans ce modèle) et non démontrables (dans $\mathcal{PA}$).

1.2. Cohérence, incomplétude

Rappels sur la cohérence.

- Une théorie T est ***incohérente*** s'il existe une formule $\mathcal{Q}$ telle que :

  $$T \vdash \mathcal{Q} \qquad \text{et} \qquad T \vdash \neg\mathcal{Q}.$$

  Elle est dite ***cohérente*** sinon.
- De façon équivalente, T est incohérente si $T \vdash \bot$ et alors T prouve n'importe quelle formule (et sa négation), par la règle d'explosion. Une théorie incohérente n'a donc aucun intérêt.
- Si T admet un modèle (c'est-à-dire $\vDash_{\mathcal{S}} T$ pour une certaine structure $\mathcal{S}$), alors T est cohérente via le théorème de validité.

Rappels sur la complétude.

- Une théorie T est ***incomplète*** s'il existe une formule fermée $\mathcal{Q}$ telle que :
$$T \nvdash \mathcal{Q} \qquad \text{et} \qquad T \nvdash \neg\mathcal{Q}.$$
- Elle est dite ***complète*** sinon, c'est-à-dire que pour toute formule fermée $\mathcal{Q}$ on a :
$$T \vdash \mathcal{Q} \qquad \text{ou} \qquad T \vdash \neg\mathcal{Q}.$$

Dans l'énoncé du théorème de Gödel, on part d'une théorie cohérente et on prouve qu'elle est incomplète.

1.3. Hypothèse de cohérence

Quand on avait énoncé le théorème d'incomplétude dans le chapitre « Un aperçu du théorème d'incomplétude », on n'avait pas énoncé l'hypothèse « $\mathcal{PA}$ est cohérente ». Pourquoi ? Il s'agit d'une des subtilités de l'énoncé.
Si on se place dans un monde où les entiers naturels $\mathbb{N}$ existent (par exemple dans $\mathcal{ZFC}$), alors $\mathbb{N}$ est un modèle de $\mathcal{PA}$ (tout est fait pour : les axiomes de $\mathcal{PA}$ correspondent aux propriétés des entiers), donc $\mathcal{PA}$ est cohérente, ce qui rend l'hypothèse inutile.
Mais si on souhaite rester dans le cadre strict du langage $\mathcal{L}_0$ et de la théorie $\mathcal{PA}$, alors l'existence de $\mathbb{N}$ n'est pas acquise, et par conséquent, la cohérence de $\mathcal{PA}$ non plus.
Ce point est d'ailleurs irrésoluble : le second théorème d'incomplétude affirmera qu'une théorie (suffisamment riche) ne peut pas prouver sa propre cohérence.

2. ω-cohérence, être une preuve, être démontrable

2.1. ω-cohérence

On va prouver le théorème de Gödel avec une hypothèse un peu plus forte : l'ω-cohérence (c'est celle de l'article original de Gödel). On discutera un peu plus tard de la façon de s'en passer.

Définition.
Une théorie $\mathcal{T}$ est ***ω-incohérente*** s'il existe une formule $\mathcal{Q}(x)$ (ayant x pour seule variable libre) telle que :
$$\mathcal{T} \quad \vdash \quad \exists x\ \mathcal{Q}(x)$$
$$\text{et pour chaque } n \in \mathbb{N}, \qquad \mathcal{T} \quad \vdash \quad \neg\mathcal{Q}(\overline{n})$$
Sinon elle est dite ***ω-cohérente***.

- C'est donc une situation d'incohérence spécifique aux entiers naturels : on a $\neg\mathcal{Q}(0)$, $\neg\mathcal{Q}(1)$, $\neg\mathcal{Q}(2)$, etc., et pourtant, il existe x tel que $\mathcal{Q}(x)$.
- Dans une théorie ω-cohérente, une telle formule n'existe pas.
- Cette définition sort du langage $\mathcal{L}_0$ car elle fait intervenir les entiers naturels $n = 0$, $n = 1$, …
- Si $\mathcal{T}$ admet $\mathbb{N}$ comme modèle, alors $\mathcal{T}$ est ω-cohérente. Et donc, si on considère que $\mathbb{N}$ est un modèle de $\mathcal{PA}$, alors $\mathcal{PA}$ est ω-cohérente.

Voici le lien avec la cohérence classique :

- $\mathcal{T}$ ω-cohérente implique $\mathcal{T}$ cohérente.
- $\mathcal{T}$ incohérente implique $\mathcal{T}$ ω-incohérente.

Justification : si $\mathcal{T}$ est incohérente, alors $\mathcal{T}$ prouve n'importe quoi, donc prouve les formules de la définition de la ω-incohérence. L'autre sens s'obtient par contraposition.
Ces deux notions ne sont pas équivalentes, mais on verra un peu plus loin l'astuce de Rosser qui permet de remplacer l'hypothèse « ω-cohérente » de Gödel par l'hypothèse plus faible de « cohérente ».

2.2. Être une preuve, être démontrable

Nous revenons sur la distinction fondamentale entre vérifier qu'on a une preuve et décider si une formule est démontrable.

- On rappelle que est_preuve(N, n), qui décide si la suite des formules de super-nombre de Gödel N constitue une preuve de la formule dont le nombre de Gödel est n, est un prédicat récursif primitif.
- Donc par la représentabilité, il existe une formule correspondante dans $\mathcal{L}_0$:

$$\mathcal{E}\text{st-preuve}(y, x)$$

- Par contre, est_démontrable(n) n'est pas un prédicat récursif primitif, car la recherche d'une preuve doit se faire sur un ensemble d'entiers N non borné.
- Mais ce n'est pas grave car dans $\mathcal{L}_0$ il est tout à fait légitime de définir la formule :

$$\mathcal{E}\text{st-démontrable}(x) \ : \quad \exists y \ \ \mathcal{E}\text{st-preuve}(y, x)$$

  Noter que pour obtenir cette formule de $\mathcal{L}_0$, on est passé par l'intermédiaire de la représentabilité de la relation « être une preuve ».

Corollaire 1.
L'ensemble des formules démontrables est définissable.

En effet, par définition, une formule $\mathcal{P}$ est démontrable (dans $\mathcal{PA}$) si et seulement si $\mathcal{E}$st-démontrable$(\ulcorner\mathcal{P}\urcorner)$ est vraie (sur $\mathbb{N}$).
C'est à mettre en opposition avec la propriété « être vraie » qui n'est pas définissable (voir le théorème de Tarski du chapitre « Diagonalisation »).
Retenez donc ces deux différences cruciales :

- « être une preuve » et « être démontrable » sont des prédicats fondamentalement différents, l'un est facilement calculable, l'autre non.
- « être démontrable » et « être vraie » sont aussi deux concepts fondamentalement différents, l'un est définissable, l'autre pas.

3. Preuve du théorème d'incomplétude pour $\mathcal{PA}$

Voici l'énoncé que l'on va prouver :

Théorème 2.
Si $\mathcal{PA}$ est ω-cohérente, alors $\mathcal{PA}$ est incomplète.

3.1. La formule à diagonaliser

On note $\mathcal{P}(x)$ la formule suivante :

$$\mathcal{P}(x) \ : \quad \neg\mathcal{E}\text{st-démontrable}(x)$$

C'est bien une formule de $\mathcal{L}_0$ d'après ce que l'on a dit précédemment.
On a donc :

$$\mathcal{E}\text{st-démontrable}(x) \ : \quad \exists y \ \ \mathcal{E}\text{st-preuve}(y, x)$$

donc

$$\neg\mathcal{E}\text{st-démontrable}(x) \ : \quad \forall y \ \ \neg\mathcal{E}\text{st-preuve}(y, x)$$

Par le lemme de diagonalisation, il existe une formule fermée $\mathcal{G}$, dite ***formule de Gödel***, telle que :

$$\mathcal{PA} \quad \vdash \quad \big(\mathcal{G} \leftrightarrow \mathcal{P}(\ulcorner\mathcal{G}\urcorner)\big)$$

(On note $\ulcorner\mathcal{G}\urcorner$ à la fois pour le nombre de Gödel de $\mathcal{G}$, mais aussi pour son numéral.)

Ainsi le lemme de diagonalisation fournit dans $\mathcal{PA}$ l'équivalence :

$$\mathcal{G} \leftrightarrow \neg\exists y \ \ \mathcal{E}\text{st-preuve}(y, \ulcorner\mathcal{G}\urcorner)$$

c'est-à-dire :

$$\neg\mathcal{G} \leftrightarrow \exists y \ \ \mathcal{E}\text{st-preuve}(y, \ulcorner\mathcal{G}\urcorner)$$

Autrement dit :

$$\mathcal{G} \leftrightarrow \neg\mathcal{E}\text{st-démontrable}(\ulcorner\mathcal{G}\urcorner)$$

Ainsi $\mathcal{G}$ signifie « Je ne suis pas démontrable ».
On va maintenant montrer que cette formule $\mathcal{G}$ n'est pas démontrable, et que $\neg\mathcal{G}$ ne l'est pas non plus.

3.2. Preuve de $\mathcal{PA} \nvdash \mathcal{G}$

Par l'absurde, si $\mathcal{PA} \vdash \mathcal{G}$, alors il existe une preuve de $\mathcal{G}$ à partir des axiomes de $\mathcal{PA}$. On note N le super-nombre de Gödel associé aux formules qui constituent cette preuve.
On a donc est_preuve$(N, \ulcorner\mathcal{G}\urcorner)$ (en tant que prédicat récursif primitif) donc :

$$\begin{array}{lll} & \mathcal{PA} \vdash & \mathcal{E}\text{st-preuve}(\overline{N}, \ulcorner\mathcal{G}\urcorner) \\ \text{donc} & \mathcal{PA} \vdash & \exists y \ \ \mathcal{E}\text{st-preuve}(y, \ulcorner\mathcal{G}\urcorner) \\ \text{donc} & \mathcal{PA} \vdash & \mathcal{E}\text{st-démontrable}(\ulcorner\mathcal{G}\urcorner) \\ \text{donc} & \mathcal{PA} \vdash & \neg\mathcal{G} \quad \text{par l'équivalence de la diagonalisation.} \end{array}$$

Conclusion : si $\mathcal{PA} \vdash \mathcal{G}$ alors $\mathcal{PA} \vdash \neg\mathcal{G}$, ce qui implique que $\mathcal{PA}$ est incohérente. Or on a supposé $\mathcal{PA}$ ω-cohérente, donc cohérente (cette hypothèse simple suffit pour cette première étape). On obtient une contradiction.

3.3. Preuve de $\mathcal{PA} \nvdash \neg\mathcal{G}$

Supposons par l'absurde que $\mathcal{PA} \vdash \neg\mathcal{G}$, alors par l'équivalence de la diagonalisation on a :

$$\mathcal{PA} \quad \vdash \quad \exists y \ \ \mathcal{E}\text{st-preuve}(y, \ulcorner\mathcal{G}\urcorner)$$

Si on définit $\mathcal{Q}(y)$ par $\mathcal{E}$st-preuve$(y, \ulcorner\mathcal{G}\urcorner)$ cela s'écrit :

$$\mathcal{PA} \quad \vdash \quad \exists y \ \ \mathcal{Q}(y)$$

D'un autre côté, comme on suppose $\mathcal{PA} \vdash \neg\mathcal{G}$ et $\mathcal{PA}$ est cohérente (car ω-cohérente), alors $\mathcal{PA} \nvdash \mathcal{G}$ (ce qu'on a en fait déjà prouvé dans la section précédente).
Cela signifie que pour chaque entier $N = 0, 1, 2, \ldots$, il est faux que N soit une preuve de $\ulcorner\mathcal{G}\urcorner$, autrement dit, il est vrai que ¬est_preuve$(N, \ulcorner\mathcal{G}\urcorner)$. Par la (forte) représentabilité des prédicats récursifs primitifs, on a :

$$\mathcal{PA} \quad \vdash \quad \neg\mathcal{E}\text{st-preuve}(\overline{N}, \ulcorner\mathcal{G}\urcorner)$$

Cela signifie que pour $N = 0, 1, 2, \ldots$, on a

$$\mathcal{PA} \quad \vdash \quad \neg\mathcal{Q}(\overline{N})$$

Cela implique que $\mathcal{PA}$ est ω-incohérente, en contradiction avec notre hypothèse.

3.4. $\mathcal{G}$ est-elle vraie ?

Nous avons prouvé $\mathcal{PA} \nvdash \mathcal{G}$ et $\mathcal{PA} \nvdash \neg\mathcal{G}$. On se place dans un modèle de $\mathcal{PA}$, alors :

- soit la phrase $\mathcal{G}$ est vraie et elle n'est pas démontrable,
- soit la phrase $\neg\mathcal{G}$ est vraie et elle n'est pas non plus démontrable.

Dans tous les cas, il existe des vérités non démontrables !

Plaçons-nous dans la situation classique où $\mathbb{N}$ est considéré comme modèle de $\mathcal{PA}$ (par exemple on se place dans $\mathcal{ZFC}$). D'une part $\mathcal{PA}$ est ω-cohérente (donc cohérente). D'autre part $\mathcal{G}$ est vraie ! En effet $\mathcal{G}$ n'est pas démontrable, ce qui signifie que la formule $\neg\mathcal{E}$st-démontrable($\ulcorner\mathcal{G}\urcorner$) est vraie (dans $\mathbb{N}$). Donc, par l'équivalence de la diagonalisation, $\mathcal{G}$ est vraie.
Encore une fois, cela suppose l'hypothèse que $\mathbb{N}$ est un modèle de $\mathcal{PA}$ (qui sort du cadre de l'arithmétique de Peano). C'est pourquoi Gödel énonce son théorème avec une hypothèse de cohérence et sans notion de vérité afin de rester dans le cadre strict de $\mathcal{PA}$.

3.5. L'astuce de Rosser

Rosser (1936) remplace l'hypothèse d'ω-cohérence de Gödel en la simple hypothèse « $\mathcal{PA}$ est cohérente ».
Au lieu d'utiliser la phrase « Je ne suis pas démontrable » dans la preuve, il utilise la phrase : « Si je suis démontrable alors il existe une preuve plus courte de ma négation ».
Formellement la phrase $\mathcal{P}(x)$ à laquelle il applique le lemme de diagonalisation est :

$$\mathcal{P}(x) : \qquad \forall y \ \ \mathcal{E}\text{st-preuve}(y,x) \ \rightarrow \ \exists z \leqslant y \ \mathcal{E}\text{st-preuve}(z, \text{neg}(x))$$

Si $x = \ulcorner\mathcal{Q}\urcorner$, on a noté $\text{neg}(x) = \ulcorner\neg\mathcal{Q}\urcorner$.
Remarquer la quantification bornée « $\exists z \leqslant y$ », qui évite d'avoir à introduire la ω-cohérence et d'énumérer $N = 0, 1, 2, \ldots$ jusqu'à l'infini.

4. Le théorème d'incomplétude général

4.1. Énoncé

Théorème 3.
Soit $\mathcal{T}$ une théorie cohérente, ayant des axiomes récursifs et qui contient $\mathcal{PA}$. Alors $\mathcal{T}$ est incomplète.

Autrement dit, dans n'importe quelle théorie suffisamment riche, il existe des formules vraies mais non démontrables.
Comme l'arithmétique de Peano est incomplète, on pourrait croire qu'il suffit d'ajouter une formule indémontrable (comme $\mathcal{G}$) à ses axiomes afin de la rendre immédiatement démontrable dans cette nouvelle théorie. Mais le théorème de Gödel prouve que cette nouvelle théorie sera aussi incomplète : il existe une nouvelle formule indémontrable. Il n'y a donc aucun moyen de s'en sortir : il existe toujours des formules vraies et non démontrables.
Revenons sur les hypothèses. On a déjà rappelé ce que signifie « $\mathcal{T}$ cohérente ». L'hypothèse « $\mathcal{T}$ contient $\mathcal{PA}$ » signifie exactement « $\mathcal{PA} \subset \mathcal{T}$ », c'est-à-dire que les axiomes de Peano font partie des axiomes de $\mathcal{T}$.
On dit de façon informelle que $\mathcal{T}$ est « suffisamment riche ». C'est cependant une exigence assez faible en mathématiques que de pouvoir faire des calculs sur les entiers.
Il reste l'hypothèse « les axiomes de $\mathcal{T}$ sont récursifs » que l'on va expliquer ci-dessous.

4.2. Fonctions récursives : la thèse de Church

Nous allons expliquer, sans la définir complètement, la notion de « récursive » qui est une version plus large de « récursive primitive ».
Définition informelle. Une fonction $f : \mathbb{N}^p \rightarrow \mathbb{N}$ est une fonction ***récursive*** si elle est calculable.
C'est donc un ensemble plus grand de fonctions que les fonctions récursives primitives, qui étaient les fonctions « facilement calculables ».
Il existe bien sûr une définition précise de ce qu'est une fonction récursive. C'est une variante plus technique de la notion de fonction récursive primitive avec, en plus, des opérations non bornées. Par

exemple, la fonction d'Ackermann (voir le chapitre « Fonctions récursives primitives ») est une fonction récursive, mais pas récursive primitive.
La « thèse de Church » est l'affirmation heuristique selon laquelle toutes les fonctions calculables sont récursives (et réciproquement).
Le résultat qui nous intéresse est que toute fonction récursive est représentable (comme pour les fonctions récursives primitives).
Terminons nos définitions :

- Un ensemble $A \subset \mathbb{N}$ est un ***ensemble récursif*** (resp. récursif primitif) si sa fonction caractéristique $\chi_A : \mathbb{N} \to \mathbb{N}$ est une fonction récursive (resp. fonction récursive primitive).
- Une théorie $\mathcal{T}$ a des ***axiomes récursifs*** (resp. récursifs primitifs) si $A = \{\ulcorner P \urcorner \mid P \text{ est un axiome de } \mathcal{T}\}$, l'ensemble des nombres de Gödel des axiomes de $\mathcal{T}$, est un ensemble récursif (resp. récursif primitif).

Exiger que l'ensemble des axiomes d'une théorie soit récursif signifie juste qu'on sait calculer si une formule donnée est un axiome ou pas. C'est une exigence extrêmement raisonnable : on doit pouvoir reconnaître les axiomes de notre théorie !

4.3. Preuve du théorème d'incomplétude

Comme on n'a pas donné la définition exacte d'une fonction récursive, on remplace cette hypothèse par l'hypothèse un peu plus forte : « les axiomes de $\mathcal{T}$ sont récursifs primitifs », ce qui fait que l'on sait calculer (facilement) si une formule donnée est un axiome ou pas.
Cela permet donc de construire la fonction prédicat est_preuve$(N, n) : \mathbb{N}^2 \to \mathbb{N}$ récursive primitive, puis de la représenter par une formule de $\mathcal{L}_0$, $\mathcal{E}$st-preuve(y, x). On remplace aussi l'hypothèse de cohérence par celle de ω-cohérence.
La démonstration est alors exactement la même que pour la théorie $\mathcal{PA}$: on construit une formule $\mathcal{G}$ par diagonalisation et on prouve que ni $\mathcal{G}$, ni $\neg\mathcal{G}$ ne sont démontrables.
Si $\mathbb{N}$ est un modèle de la théorie $\mathcal{T}$, alors $\mathcal{G}$ est vraie, bien que non démontrable.

Retour sur la complétude.
On sait par le théorème de complétude que si une formule est vraie dans tous les modèles, alors elle est démontrable. Ce n'est pas du tout une contradiction avec le théorème d'incomplétude, cela signifie juste ici qu'il existe des modèles dans lesquels $\mathcal{G}$ est fausse. Ces modèles sont appelés *modèles non standards* de $\mathcal{PA}$: ils respectent les axiomes de Peano mais ne sont pas égaux à $\mathbb{N}$ (par exemple, ils possèdent des entiers infiniment grands).

Le second théorème d'incomplétude

Chapitre 19

Nous clôturons ce livre avec le second théorème d'incomplétude de Gödel.

1. Le second théorème de Gödel

1.1. Énoncé

Théorème 1.
Soit $\mathcal{T}$ une théorie cohérente, contenant $\mathcal{PA}$, ayant des axiomes récursifs. Alors $\mathcal{T}$ ne peut pas prouver sa propre cohérence.

Si l'on note $\mathcal{E}$st-cohérente($\mathcal{T}$) le prédicat « être une théorie cohérente », la conclusion du théorème est donc :

$$\boxed{\mathcal{T} \quad \nvdash \quad \mathcal{E}\text{st-cohérente}(\mathcal{T})}$$

Noter que les hypothèses sont les mêmes que celles du premier théorème.

1.2. Cohérence

On rappelle qu'une théorie est ***incohérente*** si elle prouve n'importe quelle formule, en particulier la formule absurde « $0 = 1$ », ce qui peut s'exprimer dans le langage $\mathcal{L}_0$ par $\mathcal{E}$st-démontrable($\ulcorner 0 = 1 \urcorner$), où on note bien sûr $1 = S(0)$. Ainsi on définit la formule :

$$\mathcal{E}\text{st-cohérente}(\mathcal{T}) \ : \quad \neg\mathcal{E}\text{st-démontrable}(\ulcorner 0 = 1 \urcorner)$$

On peut ainsi exprimer la cohérence par une formule de $\mathcal{L}_0$.

2. Preuve

La preuve est une conséquence assez directe du premier théorème si on accepte sa version formalisée.

2.1. Formalisation du premier théorème d'incomplétude

On suppose que $\mathcal{T}$ est une théorie qui contient $\mathcal{PA}$, on suppose aussi que les axiomes de $\mathcal{T}$ sont récursifs, et on s'intéresse seulement à l'hypothèse de cohérence.
On rappelle que le lemme de diagonalisation fournit une formule $\mathcal{G}$ telle que :

$$\mathcal{T} \quad \vdash \quad \big(\mathcal{G} \ \leftrightarrow \ \neg\mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner)\big)$$

Et l'énoncé du premier théorème d'incomplétude est alors :

$\mathcal{T}$ cohérente implique $\mathcal{G}$ non démontrable.

Mais par le lemme de diagonalisation, $\mathcal{G}$ signifie exactement « $\mathcal{G}$ non démontrable », donc l'énoncé du premier théorème est aussi :

$\mathcal{T}$ cohérente implique $\mathcal{G}$.

Voici la formalisation du premier théorème d'incomplétude :

$$\boxed{\mathcal{T} \quad \vdash \quad \big(\mathcal{E}\text{st-cohérente}(\mathcal{T}) \to \mathcal{G}\big)}$$

C'est exactement l'énoncé précédent qui a été *internalisé*. L'énoncé « $\mathcal{T}$ cohérente implique $\mathcal{G}$ » est prouvable (ce qu'on a vu dans les chapitres précédents) mais ici on admet qu'on pourrait en écrire une preuve dans la théorie $\mathcal{T}$. On reviendra sur cette formalisation dans la dernière section.

2.2. Preuve du second théorème d'incomplétude

On admet la version formalisée du premier théorème afin de prouver le second. En cela, on suit la même démarche que Gödel quand il énonce le second théorème et esquisse la preuve à la fin de son article fondateur sur ses théorèmes d'incomplétude.
On a aussi besoin d'admettre le principe appelé « nécessité de la preuve » :

$$\text{si} \quad \mathcal{T} \vdash \mathcal{Q} \quad \text{alors} \quad \mathcal{T} \vdash \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner)$$

Cela signifie que si une théorie prouve une formule, alors elle sait qu'elle peut prouver cette formule !

Prouvons maintenant qu'une théorie cohérente ne peut pas prouver sa propre cohérence.
On part d'une théorie quelconque $\mathcal{T}$ et on suppose qu'elle prouve sa propre cohérence :

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-cohérente}(\mathcal{T})$$

Par *modus ponens* appliqué à la formalisation du premier théorème d'incomplétude, on a alors :

$$\begin{aligned} & \mathcal{T} \quad \vdash \quad \mathcal{G} \\ \text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \quad \text{par la nécessité de la preuve} \\ \text{donc} \quad & \mathcal{T} \quad \vdash \quad \neg\mathcal{G} \quad \text{par l'équivalence de la diagonalisation} \end{aligned}$$

Bilan :

$$\mathcal{T} \quad \vdash \quad \mathcal{G} \qquad \text{et} \qquad \mathcal{T} \quad \vdash \quad \neg\mathcal{G}$$

Donc $\mathcal{T}$ est incohérente.

Ainsi, par contraposition, on a bien démontré le second théorème :

Si $\mathcal{T}$ est cohérente alors $\mathcal{T} \nvdash \mathcal{E}\text{st-cohérente}(\mathcal{T})$.

2.3. Retour sur la formalisation

La preuve expliquée ci-dessus est l'esquisse originale de Gödel (1931). Il faudra attendre les travaux de Hilbert et Bernays (1934-1939), puis Löb (1955) pour obtenir une preuve détaillée. Cette preuve repose sur trois conditions, appelées « conditions de Hilbert-Bernays-Löb » :

- **C1** - *Nécessité de la preuve.*

$$\text{Si } \mathcal{T} \quad \vdash \quad \mathcal{Q}, \quad \text{alors } \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner)$$

- **C2** - *Modus ponens des preuves.*

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{P} \to \mathcal{Q} \urcorner) \to \big(\mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{P} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner)\big)$$

- **C3** - *Introspection de la preuve.*

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner) \urcorner)$$

On admet que ces conditions sont vérifiées. (Elles sont très techniques à démontrer, il faudra attendre 1955 pour en achever la formalisation.) On peut maintenant donner une preuve rigoureuse de la formalisation du premier théorème d'incomplétude. On part de l'équivalence que la diagonalisation fournit :

$$\begin{aligned}
& \mathcal{T} \quad \vdash \quad \mathcal{G} \leftrightarrow \neg \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \\
\text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{G} \to \neg \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \quad \text{on n'a gardé que l'implication directe} \\
\text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}\big(\ulcorner \mathcal{G} \to \neg \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \urcorner\big) \quad \text{par (C1)} \\
\text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \neg \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \urcorner) \quad \text{par (C2)}
\end{aligned}$$

Mais par (C3), on a toujours :

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \urcorner)$$

En posant :

$$\mathcal{Q} : \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner)$$

on obtient une double conclusion :

$$\begin{aligned}
& \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \urcorner) \wedge \mathcal{E}\text{st-démontrable}(\ulcorner \neg \mathcal{Q} \urcorner) \\
\text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{Q} \wedge \neg \mathcal{Q} \urcorner) \\
\text{donc} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \mathcal{E}\text{st-démontrable}(\ulcorner 0 = 1 \urcorner) \\
\text{ainsi} \quad & \mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner) \to \neg \mathcal{E}\text{st-cohérente}(\mathcal{T})
\end{aligned}$$

Par contraposition :

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-cohérente}(\mathcal{T}) \to \neg \mathcal{E}\text{st-démontrable}(\ulcorner \mathcal{G} \urcorner)$$

Mais par diagonalisation, la conclusion est exactement $\mathcal{G}$, donc :

$$\mathcal{T} \quad \vdash \quad \mathcal{E}\text{st-cohérente}(\mathcal{T}) \to \mathcal{G}$$

Ce qui est la formalisation souhaitée du premier théorème d'incomplétude.

Annexe

La vie de Gödel

Kurt Gödel est né à Brno en 1906 (Autriche-Hongrie, aujourd'hui en Tchéquie) et est mort à Princeton (États-Unis) en 1978. Il commence ses études à Vienne en 1923, où il s'intéresse à la physique, aux mathématiques, mais aussi à la philosophie. Il termine sa thèse en 1929 sous la direction de Hans Hahn, sur des problèmes posés par David Hilbert concernant les fondements des mathématiques.

Le programme de Hilbert consistait à prouver la cohérence et la complétude des mathématiques. Gödel y répond d'abord par son théorème de complétude : « Ce qui est vrai dans absolument tous les mondes mathématiques possibles est démontrable (et réciproquement). »

Cependant, en 1930, il obtient son résultat majeur qui fixe les limites des théories mathématiques : les théorèmes d'incomplétude. « Pour toute théorie assez riche pour décrire le monde des entiers naturels, il existe des formules vraies pour ces entiers, mais non démontrables. » Le paradoxe avec le résultat précédent n'est qu'apparent, car ici on ne s'intéresse qu'au monde spécifique des entiers naturels. Ces résultats sont publiés dans l'article « Sur les propositions formellement indécidables » en 1931.

Ces travaux ont tout de suite eu un retentissement international et surprennent et déstabilisent la communauté mathématique, à commencer par Hilbert, dont le credo était : « Nous devons savoir, nous saurons ». Hilbert est d'abord incrédule, puis reconnaît la validité de la preuve de Gödel et s'attelle ensuite à en diffuser les résultats. En effet, la communauté mathématique avait toujours pensé que tout ce qui est vrai devait pouvoir être démontré. Avec les théorèmes de Gödel, cela devient impossible ! C'est le résultat le plus important en logique du siècle.

Gödel a eu l'occasion d'aller présenter ses travaux dans le monde entier, en particulier aux États-Unis, où il a rencontré Albert Einstein dès 1933. Dans les années 1930, l'Autriche devient fasciste jusqu'à célébrer son rattachement au régime nazi (l'Anschluss, mars 1938). Gödel n'est pas juif, mais il a travaillé avec de nombreux professeurs juifs et possède un esprit indépendant. Cela suffit pour qu'il se retrouve mis à l'écart de l'université.

Aussi, en décembre 1939, il quitte Vienne avec sa femme Adèle pour les États-Unis. Il ne part pas vers l'Ouest pour traverser l'Atlantique, mais vers l'Est : à l'aide du Transsibérien il traverse la Russie, puis passe par le Japon, traverse le Pacifique pour arriver à San Francisco, et arrive en mars 1940 à Princeton (sur la côte Est des États-Unis) où Einstein l'attend. Tout cela à travers des pays en guerre !

Voici une autre anecdote célèbre : pour se faire naturaliser américain, Gödel passe devant un juge avec Einstein comme témoin. À cette occasion, Gödel avait étudié la constitution américaine et, avant d'entrer dans la salle d'audience, il dit à Einstein qu'il y a trouvé une faille logique. Heureusement, Einstein empêche Gödel d'en parler devant le juge !

Gödel et Einstein restent amis jusqu'à la mort de ce dernier en 1955. Tous les jours, ils marchent et

discutent ensemble sur leur chemin vers Princeton. Gödel continue ses travaux et se tourne vers la philosophie et la théorie de la relativité.

Toute sa vie, Gödel a été obsédé par sa santé. De plus en plus hypocondriaque, il craint un empoisonnement accidentel puis intentionnel, et s'enferme dans sa paranoïa. Il ne côtoie plus qu'Einstein et son épouse Adèle (qui devait d'ailleurs goûter tous ses plats). Lorsque sa femme est hospitalisée, il ne s'alimente plus et meurt en 1978 à l'âge de 71 ans.

Lectures

Voici quelques références commentées pour varier les points de vue et approfondir le sujet.

- *forall x : An Introduction to Formal Logic* de P. D. Magnus, T. Button, R. Trueman, R. Zach.
  forallx.openlogicproject.org
  Une bonne introduction à la logique formelle qui couvre à la fois la logique Vrai/Faux, la logique de premier ordre et les preuves dans ces deux systèmes.
  Les chapitres sont courts, bien motivés et adaptés aux étudiants de première ou de deuxième année, n'ayant jamais eu de cours de logique (voire même de mathématiques).
  Le livre est écrit en anglais, avec quand même beaucoup de texte et des exemples issus du langage courant. Cela fait une ressource accessible aux étudiants des filières non mathématiques.
- *Introduction à la logique – Théorie de la démonstration et des modèles*, R. David, K. Nour, C. Raffalli, édition Dunod.
  Livre qui met l'accent sur les démonstrations formelles en logique propositionnelle et en logique du premier ordre. Les premiers chapitres sont accessibles aux étudiants de licences.
- *Logique mathématique*, tomes 1 et 2, de R. Cori et D. Lascar, édition Dunod, traduit en anglais sous le titre *Mathematical Logic*.
  Il s'agit d'ouvrages de référence plutôt destinés aux étudiants de master. Le premier tome couvre la logique propositionnelle et la logique du premier ordre (et bien plus), le second tome couvre les théorèmes d'incomplétude (et bien plus). Le livre est complet et très bien écrit, adapté à ceux qui ont déjà suivi un cours de logique et ont un bagage mathématique sérieux.
- *A Mathematical Introduction to Logic*, de H. Enderton, édition Academic Press. C'est un ouvrage de référence pour les étudiants de fin de licence et de master. Il couvre la logique propositionnelle, la logique du premier ordre, les théorèmes d'incomplétude et la logique de second ordre.
- *Gödel Without (Too Many) Tears*, de P. Smith :
  logicmatters.net/igt
  C'est un livre consacré uniquement aux théorèmes d'incomplétude de Gödel. L'auteur est professeur de logique et de philosophie et adopte un style plus littéraire, destiné à un public non mathématicien, avec de nombreuses explications conceptuelles. Dans cette version courte il n'y a pas de preuves, elles sont reportées dans la version étendue : *An Introduction to Gödel's Theorems*.
- *Aux frontières des mathématiques - Kurt Gödel et l'incomplétude*, de Jean-Paul Delahaye (édition Dunod) raconte à la fois la vie de Gôdel et ses découvertes. *Les démons de Gödel : Logique et folie*, de Pierre Cassou-Noguès (édition du Seuil) explore les tourments de Gödel.

Tables

Connecteurs

Symbole	Nom	Définition
$\neg$	non	$\neg P$ est vrai ssi P est faux
$\wedge$	et	$P \wedge Q$ est vrai ssi P et Q sont vrais
$\vee$	ou	$P \vee Q$ est vrai ssi P ou Q est vrai
$\rightarrow$	implication	$(\neg P) \vee Q$
$\leftrightarrow$	équivalence	$(P \rightarrow Q) \wedge (Q \rightarrow P)$

Équivalences logiques usuelles de la logique Vrai/Faux

Nom	Règle	
Idempotence	$\mathcal{P} \wedge \mathcal{P} \equiv \mathcal{P}$	$\mathcal{P} \vee \mathcal{P} \equiv \mathcal{P}$
Commutativité	$\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P}$	$\mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P}$
Associativité	$(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R})$	$(\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R})$
Distributivité	$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$	$\mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R})$
Absorption	$\mathcal{P} \wedge (\mathcal{P} \vee \mathcal{Q}) \equiv \mathcal{P}$	$\mathcal{P} \vee (\mathcal{P} \wedge \mathcal{Q}) \equiv \mathcal{P}$
Lois de De Morgan	$\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv (\neg\mathcal{P}) \vee (\neg\mathcal{Q})$	$\neg(\mathcal{P} \vee \mathcal{Q}) \equiv (\neg\mathcal{P}) \wedge (\neg\mathcal{Q})$
Contraposée	$\mathcal{P} \rightarrow \mathcal{Q} \equiv (\neg\mathcal{Q}) \rightarrow (\neg\mathcal{P})$	

Règles d'inférence de la logique Vrai/Faux

Nom	Règle
Réutilisation	$\mathcal{P} \quad \vdash \quad \mathcal{P}$
Élimination de $\wedge$	$\mathcal{P} \wedge \mathcal{Q} \quad \vdash \quad \mathcal{P}$ $\mathcal{P} \wedge \mathcal{Q} \quad \vdash \quad \mathcal{Q}$
Introduction de $\wedge$	$\mathcal{P},\ \mathcal{Q} \quad \vdash \quad \mathcal{P} \wedge \mathcal{Q}$
Élimination de la double négation	$\neg\neg\mathcal{P} \quad \vdash \quad \mathcal{P}$
Introduction de $\vee$	$\mathcal{P} \quad \vdash \quad \mathcal{P} \vee \mathcal{Q}$ $\mathcal{Q} \quad \vdash \quad \mathcal{P} \vee \mathcal{Q}$
Élimination de $\vee$	$\mathcal{P} \vee \mathcal{Q},\ \mathcal{P} \to \mathcal{R},\ \mathcal{Q} \to \mathcal{R} \quad \vdash \quad \mathcal{R}$
Implication (modus ponens)	$\mathcal{P},\ \mathcal{P} \to \mathcal{Q} \quad \vdash \quad \mathcal{Q}$
Introduction de $\to$	Si $\mathcal{P} \vdash \mathcal{Q}$ alors $\vdash \mathcal{P} \to \mathcal{Q}$
Explosion (ex falso)	$\bot \quad \vdash \quad \mathcal{P}$
Équivalence	$\mathcal{P} \leftrightarrow \mathcal{Q} \quad \vdash \quad \mathcal{P} \to \mathcal{Q}$ $\mathcal{P} \leftrightarrow \mathcal{Q} \quad \vdash \quad \mathcal{Q} \to \mathcal{P}$ $\mathcal{P} \to \mathcal{Q},\ \mathcal{Q} \to \mathcal{P} \quad \vdash \quad \mathcal{P} \leftrightarrow \mathcal{Q}$
Définition de la négation	$\neg\mathcal{P}$ est définie par $\mathcal{P} \to \bot$
Introduction de la double négation	$\mathcal{P} \quad \vdash \quad \neg\neg\mathcal{P}$
Tiers exclu	$\vdash \quad \mathcal{P} \vee \neg\mathcal{P}$
Modus tollens	$\mathcal{P} \to \mathcal{Q},\ \neg\mathcal{Q} \quad \vdash \quad \neg\mathcal{P}$
Lois de De Morgan	$\neg(\mathcal{P} \wedge \mathcal{Q}) \quad \vdash \quad \neg\mathcal{P} \vee \neg\mathcal{Q}$ …
Raisonnement par l'absurde	$\neg\mathcal{P} \to \bot \quad \vdash \quad \mathcal{P}$
Contraposée	$\mathcal{P} \to \mathcal{Q} \quad \vdash \quad \neg\mathcal{Q} \to \neg\mathcal{P}$ $\neg\mathcal{Q} \to \neg\mathcal{P} \quad \vdash \quad \mathcal{P} \to \mathcal{Q}$

Connecteurs binaires

Symbole	Réalisation	Nom	Valeurs
$\bot$	$P \wedge \neg P$	toujours faux / contradiction	**0000**
$\wedge$	$P \wedge Q$	ET/ conjonction	**0001**
	$P \wedge \neg Q$	P et (non Q)	**0010**
	P	projection gauche	**0011**
	$\neg P \wedge Q$	(non P) et Q	**0100**
	Q	projection droite	**0101**
$\oplus$	$(P \vee Q) \wedge \neg(P \wedge Q)$	XOR / ou exclusif	**0110**
$\vee$	$P \vee Q$	OU/ disjonction	**0111**
$\downarrow$	$\neg(P \vee Q)$	NOR	**1000**
$\leftrightarrow$	$P \leftrightarrow Q$	équivalence	**1001**
	$\neg Q$	négation de Q	**1010**
	$Q \rightarrow P$	Q implique P	**1011**
	$\neg P$	négation de P	**1100**
$\rightarrow$	$P \rightarrow Q$	P implique Q	**1101**
$\uparrow$	$\neg(P \wedge Q)$	NAND / Sheffer	**1110**
$\top$	$P \vee \neg P$	toujours vrai / tautologie	**1111**

La dernière colonne indique les 4 valeurs de la fonction prises lorsque (P,Q) vaut successivement $(\mathbf{0},\mathbf{0})$, $(\mathbf{0},\mathbf{1})$, $(\mathbf{1},\mathbf{0})$, $(\mathbf{1},\mathbf{1})$.

Équivalences logiques usuelles de la logique de premier ordre

Nom	Règle
Négation de $\forall$	$\neg(\forall x\ \mathcal{P}(x)) \quad\equiv\quad \exists x\ \neg\mathcal{P}(x)$
Négation de $\exists$	$\neg(\exists x\ \mathcal{P}(x)) \quad\equiv\quad \forall x\ \neg\mathcal{P}(x)$
Permutation de $\forall$	$\forall x\,\forall y\ \mathcal{P}(x,y) \quad\equiv\quad \forall y\,\forall x\ \mathcal{P}(x,y)$
Permutation de $\exists$	$\exists x\,\exists y\ \mathcal{P}(x,y) \quad\equiv\quad \exists y\,\exists x\ \mathcal{P}(x,y)$
Distribution de $\forall$ sur $\wedge$	$\forall x\,(\mathcal{P}(x) \wedge \mathcal{Q}(x)) \quad\equiv\quad (\forall x\,\mathcal{P}(x)) \wedge (\forall x\,\mathcal{Q}(x))$
Distribution de $\exists$ sur $\vee$	$\exists x\,(\mathcal{P}(x) \vee \mathcal{Q}(x)) \quad\equiv\quad (\exists x\,\mathcal{P}(x)) \vee (\exists x\,\mathcal{Q}(x))$
Sortie de $\forall$	$\forall x\,(\mathcal{P}(x) \rightarrow \mathcal{Q}) \quad\equiv\quad (\exists x\,\mathcal{P}(x)) \rightarrow \mathcal{Q}$ (si x n'apparaît pas dans $\mathcal{Q}$)
Renommage d'une variable liée	$\forall x\,\mathcal{P}(x) \quad\equiv\quad \forall y\,\mathcal{P}[y/x]$ (si y est une variable nouvelle)

Règles d'inférence de la logique de premier ordre

Nom	Règle
Élimination de $\forall$	$\forall x\, \mathcal{P}(x) \quad \vdash \quad \mathcal{P}(c)$ (c constante ou terme clos quelconque)
Introduction de $\forall$	$\mathcal{P}(c) \quad \vdash \quad \forall x\, \mathcal{P}(x)$ (c arbitraire : n'apparaît dans aucune hypothèse ouverte)
Introduction de $\exists$	$\mathcal{P}(c) \quad \vdash \quad \exists x\, \mathcal{P}(x)$ (c constante ou terme clos quelconque)
Élimination de $\exists$	$\exists x\, \mathcal{P}(x),\ [\, \mathcal{P}(c) \vdash \mathcal{Q}\,] \quad \vdash \quad \mathcal{Q}$ (c est une constante nouvelle, n'apparaissant nulle part ailleurs)
Négation de $\forall$	$\neg(\forall x\, \mathcal{P}(x)) \quad \vdash \quad \exists x\, \neg\mathcal{P}(x)$
Négation de $\exists$	$\neg(\exists x\, \mathcal{P}(x)) \quad \vdash \quad \forall x\, \neg\mathcal{P}(x)$
Renommage	$\forall x\, \mathcal{P}(x) \quad \vdash \quad \forall y\, \mathcal{P}(y)$ $\exists x\, \mathcal{P}(x) \quad \vdash \quad \exists y\, \mathcal{P}(y)$
Permutation de deux $\forall$	$\forall x\, \forall y\, \mathcal{P}(x,y) \quad \vdash \quad \forall y\, \forall x\, \mathcal{P}(x,y)$
Permutation de deux $\exists$	$\exists x\, \exists y\, \mathcal{P}(x,y) \quad \vdash \quad \exists y\, \exists x\, \mathcal{P}(x,y)$
Interversion de $\exists$ *et* $\forall$	$\exists x\, \forall y\, \mathcal{P}(x,y) \quad \vdash \quad \forall y\, \exists x\, \mathcal{P}(x,y)$
Commutation de $\forall$ *et* $\wedge$	$\forall x(\mathcal{P}(x) \wedge \mathcal{Q}(x)) \quad \vdash \quad (\forall x\, \mathcal{P}(x)) \wedge (\forall x\, \mathcal{Q}(x))$ $(\forall x\, \mathcal{P}(x)) \wedge (\forall x\, \mathcal{Q}(x)) \quad \vdash \quad \forall x(\mathcal{P}(x) \wedge \mathcal{Q}(x))$
Commutation de $\exists$ *et* $\vee$	$\exists x(\mathcal{P}(x) \vee \mathcal{Q}(x)) \quad \vdash \quad (\exists x\, \mathcal{P}(x)) \vee (\exists x\, \mathcal{Q}(x))$ $(\exists x\, \mathcal{P}(x)) \vee (\exists x\, \mathcal{Q}(x)) \quad \vdash \quad \exists x(\mathcal{P}(x) \vee \mathcal{Q}(x))$

Axiomes de Peano

- $\mathcal{A}_1 : \quad \forall x \quad S(x) \neq 0$
- $\mathcal{A}_2 : \quad \forall x \quad (x = 0) \vee (\exists y \quad x = S(y))$
- $\mathcal{A}_3 : \quad \forall x \ \ \forall y \quad (S(x) = S(y)) \to (x = y)$
- $\mathcal{A}_4 : \quad \forall x \quad x + 0 = x$
- $\mathcal{A}_5 : \quad \forall x \ \ \forall y \quad x + S(y) = S(x + y)$
- $\mathcal{A}_6 : \quad \forall x \quad x \times 0 = 0$
- $\mathcal{A}_7 : \quad \forall x \ \ \forall y \quad x \times S(y) = (x \times y) + x$

Et pour chaque formule $\mathcal{P}$ ayant x comme variable libre :

$$\mathcal{R}ec_{\mathcal{P}(x)} : \quad \mathcal{P}(0) \ \wedge \ \big[\forall x \quad \mathcal{P}(x) \to \mathcal{P}(S(x))\big] \quad \to \quad \forall x \ \ \mathcal{P}(x)$$

Lexique français-anglais

Ce lexique répertorie les termes techniques fondamentaux avec leurs équivalents en anglais.

- affectation : *assignment*, *variable assignment*
- codage de Gödel : *Gödel coding*, *Gödel numbering*
- conséquence logique : *logical consequence*, *semantic consequence*
- constante fraîche : *fresh constant*, *new constant*
- déduction naturelle : *natural deduction*
- domaine : *domain*, *domain of discourse*, *universe*
- ensemble définissable : *definable set*
- équivalence logique : *logical equivalence*
- explosion : *explosion*, *ex falso quodlibet*
- fonction récursive primitive : *primitive recursive function*
- formule : *formula*, *well-formed formula (wff)*
- formule atomique : *atomic formula*
- formule fermée, formule close, phrase : *closed formula*, *sentence*
- hypothèse : *assumption*, *hypothesis*
- logique propositionnelle : *propositional logic*
- logique du premier ordre : *first-order logic*
- modèle : *model*
- portée (d'un quantificateur) : *scope*
- règle d'inférence : *rule of inference*
- table de vérité : *truth table*
- tautologie : *tautology*
- témoin de Henkin : *Henkin witness*
- terme clos : *closed term*
- théorie cohérente/incohérente : *consistent/inconsistent theory*
- théorie complète : *complete theory*
- théorème de compacité : *Compactness Theorem*
- théorème de complétude : *Completeness Theorem*
- théorème d'incomplétude : *Incompleteness Theorem*
- théorème de validité : *Soundness Theorem*
- tiers exclu : *law of excluded middle*
- variable libre : *free variable*
- variable liée : *bound variable*

Remerciements

Je remercie Stéphanie Bodin et Michel Bodin pour leurs relectures.

Index