Spécification et modélisation Informatique

Notes de Cours

Olivier Pons et Marianne Simonot
CNAM

13 novembre 2018

Table des matières

1 Introduction **5**
- 1.1 Un exemple introductif . . . 6
- 1.2 Les ingrédients des formules . . . 6
- 1.3 Traduction des faits en formules . . . 7
- 1.4 Formaliser, un travail d'abstraction et de précision . . . 8
  - 1.4.1 Les propositions ont été vidées de leur sens concret. . . 8
  - 1.4.2 Les liens logiques entre les propositions ont été précisés. 8
- 1.5 Raisonner sur la formalisation . . . 9
  - 1.5.1 Le point de départ . . . 10
  - 1.5.2 Modèles de la formule 1 . . . 10
  - 1.5.3 Modèles des formules 3 et 1 . . . 11
  - 1.5.4 Modèles des formules 3, 1 et 2 . . . 11
  - 1.5.5 Où l'on déduit que Benoit est coupable . . . 12
  - 1.5.6 Où l'on déduit que B est une fille . . . 12
  - 1.5.7 Différentes notions de vérité . . . 13

2 Langages **15**
- 2.1 Langages formels . . . 15
  - 2.1.1 Construction et raisonnement par récurrence/induction 17
  - 2.1.2 Logique formelle . . . 20

3 Le langage du Calcul des propositions **23**
- 3.1 Construire des formules bien formées . . . 23
  - 3.1.1 L'alphabet . . . 23
  - 3.1.2 La grammaire . . . 24
- 3.2 Le sens des formules . . . 25
  - 3.2.1 Les variables propositionnelles . . . 26
  - 3.2.2 Les formules complexes . . . 26
  - 3.2.3 Tables de vérité des connecteurs . . . 27
  - 3.2.4 Calculer la valeur de vérité d'une formule . . . 29
- 3.3 Modéliser . . . 30

3.4 Exercices 30
3.5 Le coté syntaxique et la démonstration 32
3.6 Exercice 40

Chapitre 1

Introduction

Dans les chapitres qui suivent, nous allons apprendre des langages.

Ces langages permettent de décrire au moyen d'un petit nombre de concepts un grand nombre de choses, en particulier des choses qui ont trait à l'informatique. On peut citer, par exemple, les spécifications de programme, les programmes eux mêmes ou encore les tables des bases de données.

Ces langages sont dit **formel**. Ils s'oppose en cela aux langues que l'on écrit ou que l'on parle, dite **naturelles**. Comme une langue naturelle, une langue formelle est formée de phrases, qui sont soumises à des règles de formation (syntaxe, grammaire ...), et qui ont **un sens**. Mais les langues formelles que nous allons étudier ont ceci de particulier qu'elles ne sont pas ambiguës : chaque phrase à un sens unique.

On distinguera parfois aussi les langages de **spécification** et les langages de **programmation**. Avec un langage de programmation, on exprime comment on obtient un résultat. Avec un langage de spécification, on exprime ce que l'on veut obtenir (c'est-à-dire les propriétés du programme que l'on cherche à écrire).

Dans le cadre de la logiques, on s'interesse à des phrase particulières appelées formules. Il est possible de vérifier que ce qu'elles expriment est "vrai" ou "faux". En logique l'utilisation de ces langage permet **raisonner** et de **démontrer**.

1.1 Un exemple introductif

Imaginons un inspecteur logicien qui décide de formaliser logiquement les résultats de son enquète en cours, afin de s'assurer de la rigueur de ses déductions.

Voici l'état d'avancement de l'enquète : on sait qu'un vol a ete commis. Trois malfaiteurs notoires Arthur, Benoit et Charles sont apprehendés. On a pu établir que

1. Nul autre que ces trois-là ne peuvent etre impliqué.
2. Arthur ne travaille jamais sans un complice
3. Charles est innocent

L'inspecteur **formalise** cela en écrivant (avec des **symboles**) les trois **formules** suivantes :

1. $A \vee B \vee C$
2. $A \Rightarrow (B \vee C)$
3. $\neg C$

L'inspecteur a écrit **des formules** qui traduisent formellement les **faits** recueillis durant l'enquète. On a une formule pour chaque fait, mais ceci n'est pas toujours le cas. Ces formules sont des formules d'un langage logique appelé *Calcul des Propositions*.

1.2 Les ingrédients des formules

Dans ces formules, on a des expressions que l'on considère **indecomposable** par exemple : A désigne la phrase : *"Arthur est coupable"*, B désigne la phrase : *"Benoit est coupable"* et C désigne la phrase : *"Charles est coupable"*.

On appelle ces phrases des *propositions* (on dira aussi variables propositionnelles, ou formules atomiques).

On a aussi des liens correspondants aux conjonctions de la langue naturelle ($\wedge$ pour *et*, $\vee$ pour *ou* , $\neg$ pour *non*, $\Rightarrow$ pour *si . . .alors* et $\Leftrightarrow$ pour *si et seulement si*) que l'on appelle des **connecteurs**.

En **logique des propositions**, ce sera toujours le cas : toutes les formules devront être écrites avec ces ingrédients. Une formule est soit un lien reliant des formules plus petites, soit une formule indécomposable, une sorte de boite noire c'est à dire une proposition.

Remarque

L'inspecteur aurait aussi pu utiliser d'autres formalismes ; écrivant par example :

1. $Coup(a) \vee Coup(b) \vee Coup(c)$
2. $Coup(a) \Rightarrow (Coup(b) \vee Coup(c))$
3. $\neg Coup(c)$

Cette representation ressemble beaucoup à la précédente mais les ingredients ne sont pas les mêmes. Ici a, b et c ne representent plus des faits mais des individus (Arthur, Benoit et Charles) et $Coup$ introduit le **moyen de representer une propriété**, ici "est coupable". Le symbole $Coup$ est appelé un **prédicats** et on parle alors de **logique des prédicats**. C'est un formalisme plus puissant que la logique des propositions que nous étudierons au chapitre **??**.

La *logique des Proposition* et la *logique des Prédicats* sont avec la *théorie des Ensembles* les langages que nous étudierons dans ce cours.

Mais dans cette introduction continuons en logique des propositions et voyons comment l'inspecteur à établi sa formalisation.

1.3 Traduction des faits en formules

1. *Charles est innocent* peut être traduit par Charles n'est pas coupable. Cette phrase est décomposable au moyen d'un lien parmi nos connecteurs :

   ¬ Charles est coupable.

   Charles est coupable n'est pas décomposable avec nos connecteurs. C'est donc une proposition indécomposable, une proposition que nous nommons C. On obtient donc la formule $\neg C$.

2. *Nul autre que Arthur, Benoit et Charles ne peuvent être impliqués* peut se paraphraser en français par : Arthur est coupable ou Benoit est coupable ou Charles est coupable à condition de comprendre par *ou* le fait que l'un ou l'autre ou les deux sont coupables.

   C'est exactement ce que signifie en logique le connecteur $\vee$.

3. *Arthur ne travaille jamais sans un complice* peut être paraphrasé par : Si Arthur est coupable , alors Benoit ou Charles aussi. On reconnait ici le lien $\Rightarrow$ reliant Arthur est coupable (A) , à Benoit est coupable ou Charles est coupable qui se décompose à son tour en $B \vee C$. On

obtient donc $A \Rightarrow (B \vee C)$. Remarquons que les parenthèses refletent la hiérarchie des connecteurs issues de notre analyse.

1.4 Formaliser, un travail d'abstraction et de précision

On peut avoir l'impression de ne pas avoir fait grand chose : la traduction formelle ressemble beaucoup à l'énoncé en langue naturelle. Il y a cependant 2 différences essentielles :

1.4.1 Les propositions ont été vidées de leur sens concret.

A, B et C, ne désignent pas les 3 faits *Arthur est coupable, Benoit est coupable et Charles est coupable* tels qu'ils existent dans notre monde, mais n'importe quels faits.

Les formules parlent de 3 faits quelconques. Elles énoncent les seules choses que l'on sait d'eux.

Si, par exemple, l'inspecteur avait choisi de formaliser son problème avec un symbole de proposition supplémentaire I pour désigner *Charles est innocent*, il aurait dû traduire la troisième phrase par : I. Mais alors, le rapport entre la culpabilité et l'innocence aurait dû être explicité, en ajoutant une quatrième formule $I \Leftrightarrow \neg C$ qui dit que l'innocence de charles est le contraire de la culpabilité de Charles.

Puisque A, B et C ont été vidés de leur sens, ces trois formules désignent aussi bien les faits suivants :

1. "A est une fille" ou "B est une fille" ou "C est une fille".
2. "Si A est une fille" alors "B" ou "C" aussi.
3. "C est un garçon".

1.4.2 Les liens logiques entre les propositions ont été précisés.

A l'inverse, on a exprimé les rapports entre les propositions de façon très précise.

En remplacant, par exemple, la conjonction *ou* de la langue française par le connecteur $\vee$, on sait qu'on emploie un ou **inclusif**. Chaque connecteur autorisé à un **sens non ambigu**, parce que ce sens est défini de façon très

précise en fonction de la vérité des propositions qu'il relie (ou de la proposition qu'il nie pour le connecteur $\neg$).

Par exemple, on défini le sens du connecteur le connecteur $\vee$ (en se basant sur le sens usuel du *ou*) :

Une formule de la forme $F \vee G$ est vraie si et seulement si F est vraie, ou si G est vraie ou si les 2 sont vraies.

On représente généralement cela graphiquement par ce que l'on appelle la *table de vérité* du connecteur $\vee$. Elle exprime le sens (vrai ou faux) de la formule $(F \vee G)$ en fonction de celui des *sous formules* Fet F

F	G	(F $\vee$ G)
v	v	v
v	f	v
f	v	v
f	f	f

$\rightsquigarrow$ *F et G sont des sous formule de la formule* $(F \vee G)$

$\rightsquigarrow$ *quand F est vrai et que G est vrai, la formule* $(F \vee G)$ *est vrai*

$\rightsquigarrow$ *quand F est vrai et que G est faux, la formule* $(F \vee G)$ *est vrai*

$\rightsquigarrow$ *quand F est faux et que G est vrai, la formule* $(F \vee G)$ *est vrai*

$\rightsquigarrow$ *quand F est faux et que G est faux, la formule* $(F \vee G)$ *est faux*

1.5 Raisonner sur la formalisation

La formalisation exprime la structure logique, les liens logiques qui unissent les faits de base entre eux.

Grâce à cela, nous allons pouvoir **raisonner** sur notre formalisation et en tirer *des conclusions sur le monde réel*. Nous pourrons, par exemple, **conclure avec certitude** que Benoit est coupable. Nous pourrons aussi connaitre le statut de nos formules : sont elles vraies dans tous les mondes possibles ? , sont elles vraies dans certain monde seulement ? ou encore ne sont elles jamais vraies ?.

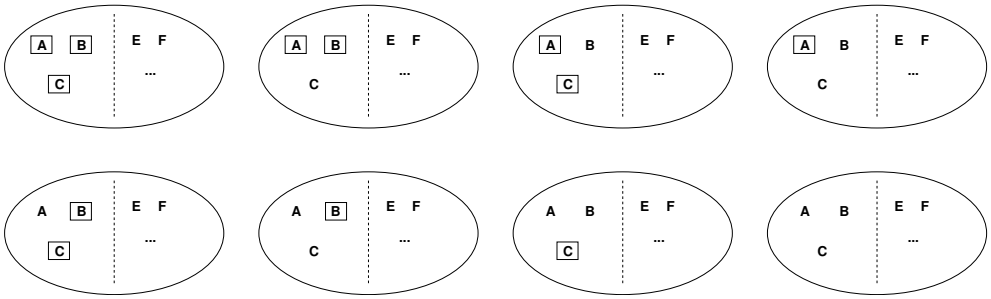
Pour raisonner, nous allons en quelques sortes faire le trajet inverse que celui que nous avons fait jusqu'ici : nous allons déterminer l'ensemble des mondes ou notre formalisation est vraie. Un monde où une formule est vraie s'appelle un *modèle* de la formule.

Les faits qui sont vrais dans tous les modèles de l'ensemble de nos trois formules seront les faits qui se déduisent d'elles.

1.5.1 Le point de départ

Les mondes qui nous intéressent doivent avoir au moins trois faits, un qui tient le rôle de *Arthur est coupable*, un autre pour *Benoit est coupable* et un troisième pour *Charles est coupable*. Les dessins qui suivent épuisent toutes les possibilités :

1. les trois sont coupables,
2. Arthur et Benoit sont coupables
3. Arthur et Charles sont coupables
4. Arthur est le seul coupable
5. Benoit et Charles sont coupables
6. Benoit est le seul coupable
7. Charles est le seul coupable
8. aucun n'est coupable



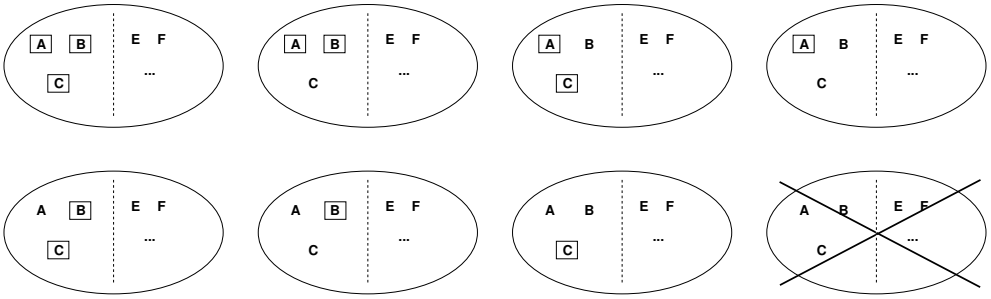
1.5.2 Modèles de la formule 1

La table de vérité du $\vee$:

s	t	(s $\vee$ t)
v	v	v
v	f	v
f	v	v
f	f	f

nous dit que $s \vee t$ n'est faux que lorsque s et t sont faux en même temps. Le seul monde où $A \vee B \vee C$ est faux est donc celui où aucuns des 3 n'est coupable.

Les modèles de cette formule sont donc :



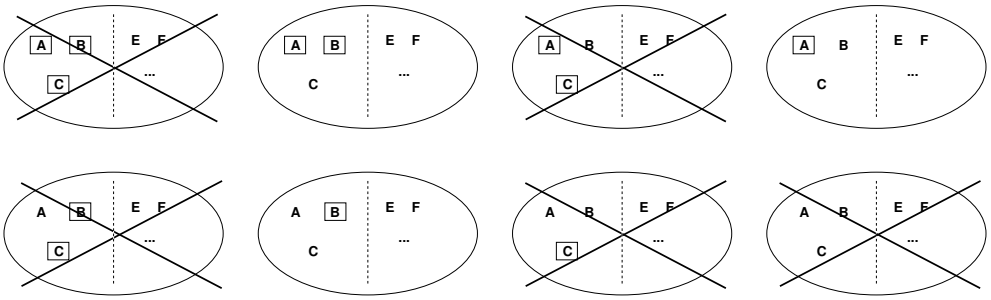
1.5.3 Modèles des formules 3 et 1

Parmi les sept restants, determinons lesquels respectent la contrainte exprimée par la formule 3 : $\neg C$.

La table de vérité de la négation nous dit que $\neg F$ est vraie là où F est fausse.

s	(¬ s)
v	f
f	v

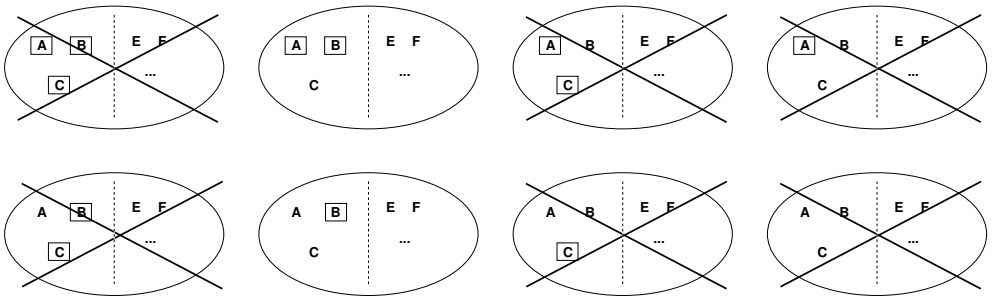
Il nous faut donc ne garder que ceux où C est fausse.



1.5.4 Modèles des formules 3, 1 et 2

Parmi les trois restants, lesquels vérifient la formule $A \Rightarrow (B \vee C)$?
Cette formule modélise le fait que dès que si Arthur est coupable, l'un ou l'autre des 2 autres aussi. Cette formule est donc vraie dans le modèle où Arthur et Benoit sont coupables. Elle est en revanche fausse dans le modèle où seul Arthur est coupable. Elle est vraie dans le modèle ou Benoit est seul coupable : une implication $X \Rightarrow Y$ n'affirme pas que X est vrai, ni que Y est vrai, seulement que X ne peut l'être sans Y. C'est bien ce que dit la table de vérité de l'implication :

s	t	(s ⇒ t)
v	v	v
v	f	f
f	v	v
f	f	v



1.5.5 Où l'on déduit que Benoit est coupable

Il n'y a que deux possibilités pour que nos 3 formules soient vraies ensemble : un monde où Arthur et Benoit sont les 2 coupables, et un monde où Benoit est coupable tout seul.

A chaque fois Benoit est coupable. La culpabilité de Benoit est donc une conséquence logique des trois formules. L'inspecteur peut l'inculper sans crainte et les juges le condamner.

Lorsqu'une formule (par exemple B) est vraie dans tous les modèles réalisant un ensemble de formules, on dit que cette formule est une *conséquence* ou *se déduit* de l'ensemble des formules.

En revanche, personne ne peut affirmer à ce point de l'enquète que A est coupable. Il est possible que Benoit est ai fait le coup tout seul. A ne se déduit pas de nos 3 formules.

1.5.6 Où l'on déduit que B est une fille

L'interprétation de A B et C par Arthur est coupable, Benoit est coupable et Charles est coupable, n'a tenu aucun rôle dans notre raisonnement. Nous avons montré que pour n'importe quelle interprétation de nos propositions A B et C, dès que nos trois formules sont vraies, la proposition B l'est aussi. Autrement dit, en interprétant A B et C par A est une fille , B est une fille, C est une fille, on déduit aussi de nos trois formules que B est une fille.

1.5.7 Différentes notions de vérité

Peut on dire quelque chose de nos trois formules de départ ? Pour chacune d'entre elles, il est possible de construire des modèles qui les rendent vraies et des modèles qui les rendent fausses. Ces formules sont dites *satisfiables* ou *réalisables*.

Les formules réalisables dans tous les mondes sont des formules *valides*, aussi appelées *tautologies* (on réserve le mot vrai pour ce cas), et celles qui ne sont vraies dans aucun monde sont des *antilogies*, on dit aussi qu'elles sont *insatisfiables*

Chapitre 2

Langages

Dans ce court chapitre nous allons rapidement préciser la notion de langage et quelques notions genérale

2.1 Langages formels

Pour definir un langage il faut se donner un **ensemble** [1] **de symboles** de base qu'on appelles l'**alphabet**. Il represente le lexique de notre langage et permet de definir ce que l'on appelle les **mots** sur cet alphabet. Les mots sont des **suites finies de symboles** que l'on note généralement par une simple concaténation.

Le nombre de symbole composant un mot définie sa longueur. Lorsque l'on note A l'alphabet on note A^n l'ensemble des mots de longueur n. On note A^* l' Ensemble de tous les mots sur l'alphabet A (c'est la reunion des ensemble de mot de chaque longueur)

$A^* = \underset{n\geq 0}{U} A^n$

Le mot particulier composé d'aucun symbole est noté ϵ.

On considere maintenant les mots et on se donne la possibilité de former de nouveaux mots en concaténant [2] des mots existants [3]. Concaténer un mot avec le mot vide ne change pas le mot initial ; on dit que le mot vide constitue un élément neutre pour l'opération de concaténation. La concatenation est associative (c'est-à-dire que pour construire le mot abc, il est equivalent de construire le mot ab et le concaténer avec le mot c ou construite le bc et de concaténer a et bc.

1. dénombrable
2. *i.e* en les mettant bout à bout
3. L'operation binaire de concaténation définie sur A est étendu à A^*

En mathématique, un ensemble muni d'une operation associative et d'un élément neutre est appelé **un monoid**. L'ensemble A^* muni concatenation est donc un monoïd.

Un **langage** sur l'alphabet A est **une partie de** A^*, c'est-à-dire un ensemble de mots sur A^*. L'ensemble de toutes les parties de A^* est l'ensemble de tout les langages sur A.

Exemple A partir de l'alphabet $A = \{I\}$, qui ne comporte qu'un élément, on peut former les mots I, II, III, ..., $IIIIIIIIII$, ...,

On peut donc énumérer ces mots en comptant le nombre de symbole. Il y a donc autant de mots possibles que d'entier naturels. On dit que A^* est isomorphe à $\mathbb{N}$. On peut donc former autant de langage que de partie de $\mathbb{N}$, c'est-à-dire plus qu'il n'est possible d'énumérer [4]

Exemple A partir de l'alphabet $A = \{x, y, 1, 2, 3+, *, (,)\}$, on peut former les mots $x + 3$, ou $x + y * (4 + x)$ mais aussi $*3 + + + *((($ ou $))(3$.

Clairement, si on veut représenter des expressions arithmétiques, les deux premiers font du sens mais on voudrait pouvoir interdire la formation des deux autres.

Exemple Sur l'alphabet $A = \{function, x, return, +, 1, (,), , toto, \ , \{, \}, \}$, on peut former les mots $function\ toto()return\ x + 1$ mais aussi $xxx\ return\ toto$ ou $))(1$.

Là encore si l'objectif est de representer des programmes en JavaScript, on voudrait interdire les deux dernières expressions.

Exemple A partir de l'alphabet $A = \{matin, \ , fait, ce, il, beau\}$, on peut former le mot $ilfaitbeaucematin$.

Si on veut faire une analogie avec les langues naturelles, l'alphabet constitue ici le lexique de la langue naturelle ($i.e$ l'ensemble des mots du dictionnaire) et les mots (au sens langage formel) sont les phrases de la langue naturelle. La langue elle même est l'ensemble des phrases possibles mais évidement toute combinaison arbitraire de mots du dictionnaire n'est pas acceptable.

Ces derniers exemples font apparaître que la formation de mot au moyen de la simple concaténation n'est pas assez restrictive puisqu'elle permet de former trop de mot. Nous allons maintenant presenter un procédé de construction beaucoup plus restrictif.

4. Le nombre de parties de $\mathbb{N}$ est isomorphe à $\mathbb{R}$ et n'est donc pas dénombrable.

2.1.1 Construction et raisonnement par récurrence/induction

Les entiers Naturels

L'ensemble des entiers est infini on peut néanmoins le décrire de maniere finie en disant qu'un entier "est soit O soit le successeur d'un autre entier".

C'est à dire en se donnant deux symboles l'un O pour representer 0, l'autre S pour representer le successeur d'un entier. Le second est un symbole fonctionnel, il ne represent pas un entier à lui tout seul, c'est en l'appliquant à un entier existant qu'on obtient un nouvel entier.

On formalise cela au moyen d'une **Définition par recurrence** :

Définition 1. *L'ensemble des entiers naturels, $\mathbb{N}$, est le plus petit ensemble de mot construit sur l'alphabet $A = \{O, S, (,)\}$ vérifiant les 2 règles suivantes :*

1. *O est un entier*
2. *si n est un entier $S(n)$ est un entier*

Remarque 1. *On peut se demander si un tel ensemble existe.*

Ici A^* vérifie les règles (puisque $O \in A^*$ et que si $n \in A^*$ alors $S(n) \in A^*$). Donc pour trouver le plus petit il suffit de prendre l'intersection de tous les ensembles vérifiants les règles. Cette intersection contient O donc n'est pas vide.

Plusieurs énoncés découlent de cette definition :

Définition 2. *Principe du raisonnement par récurrence : Soit P une propriété sur les entiers, si*

1. *$P(O)$ est vérifiée* $\rightsquigarrow$ O vérifie bien la propiété P
2. *si pour n'importe quel entier k lorsque $P(k)$ est vérifiée alors $P(S(k))$ est vraie.* $\rightsquigarrow$ Dès qu'un entier k vérifie la propriété P, son successeur $S(k)$ vérifie aussi P

alors $P(n)$ est vérifié pour tout n

Remarque 2. *Une idée de la justification : Remarquons que $\mathbb{N}$ peut être munie d'un ordre structurel (*i.e. *$n \leq m$ si n est un sous-terme de m), et que pour cet ordre toute partie de $\mathbb{N}$ admet un plus petit éléments. Raisonnons alors par l'absurde.*

Pour cela supposons :

— *que $P(O)$ vraie et que si pour n'importe quel entier k , la propriété $P(k)$ est vraie alors $P(S(k))$ est vraie.*

— *que $P(n)$ n'est pas vraie pour tout entier n.*
et montrons qu'il y a contradiction.

Soit $\mathfrak{F}$ l'ensemble des entiers pour lesquels $P(n)$ est fausse : $\mathfrak{F}$ est alors une partie non vide de $\mathbb{N}$.

Or toute partie de $\mathbb{N}$ possède un plus petit élément !

Nommons alors v le plus petit élément de $\mathfrak{F}$. On a $\neg P(v)$

Remarquons que si $P(O)$ est vraie c'est que $\neg(O \in \mathfrak{F}$. Ainsi v n'est pas O, et est donc de la forme $S(v')$ et comme v' est structurellement inferieur à v qui est le plus petit élément de $\mathfrak{F}$, $\neg(v' \in \mathfrak{F}$. Donc $P(v')$ et donc, par 2, $P(S(v'))$ soit $P(v)$ d'ou la contradiction

Définition 3. *Définition de fonction récursive Definir des ensembles inductivement permet aussi de programmer **récursivement** sur ces ensembles.*

*On peut definir des **fonctions récursives** dont la structure va se calquer sur la définition de l'ensemble. Il y aura autant de cas qu'il y a de règles dans la définition de l'ensemble ;*

Les cas correspondant aux règles de base seront des cas non récursifs et ceux correspondant aux règles inductives pourront *avoir des appels recursifs sur l'un des* "sous-élement" *dans la construction de l'élément.*

Exemple Pour definir l'addition de deux entiers (définis inductivement) on peut procéder par récurence sur le premier :

$add(O, n) \Longrightarrow n$ ⇝ *si le premier entier est O, le resultat de l'addition est le second*
$add(S(x), n) \Longrightarrow S(add(x,))$⇝ *si le premier entier est construit en prenant le successeur d'un entier x, le resultat de l'addition est obtenu en prenant le successeur de l'addition de x avec le second argument*

Ce mode de définition de l'ensemble des entiers,peut être étendu pour definir d'autre ensemble. On parle alors de définition **par induction** (ou inductive).

Autres exemples de définitions inductives

Définition 4. *L'ensemble des expressions arithmétiques, est le plus petit ensemble de mot construits sur l'alphabet $A = V \cup \{+, -, *, /(,)\}$ (ou V est un ensemble dénombrable de variable, souvent notées $\{x, y, z \ldots\}$) vérifiant les règles suivantes :*

1. *Les variables (*i.e. les éléments de V) sont des expressions arithmétiques.
2. si e_1 et e_2 sont des expressions arithmétiques $(e_1 + e_2)$ est une expression arithmétique.

3. si e_1 et e_2 sont des expressions arithmétiques $(e_1 - e_2)$ est une expression arithmétique.
4. si e_1 et e_2 sont des expressions arithmétiques $(e_1 * e_2)$ est une expression arithmétique.
5. si e_1 et e_2 sont des expressions arithmétiques (e_1 / e_2) est une expression arithmétique.

Les éléments de cet ensembles sont les mots syntaxiquement valides du langage des expressions arithmétiques.

Par exemple $((x + y) * (x - z)) + y$ est une expression arithmétique mais $x + +(y) -$ qui ne peut pas être formée avec les règles précédentes ne l'est pas.

Remarque 3. *On peut **étendre** la définition précédente. Par exemple en rajoutant à l'alphabet un ensemble constantes C (notées par exemple* $\{1, 2, 3 \ldots\}$*) et la regle "Les Constantes sont des expressions arithmétiques".*

Exemple Comme pour les entiers on peut définir par récurrence des fonctions sur des expressions arithmétique (définis inductivement).

$add(O, n) \Longrightarrow n$ ⇝ *si le premier entier est O, le resultat de l'addition est le second*

$add(S(x), n) \Longrightarrow S(add(x,))$ ⇝ *si le premier entier est construit en prenant le successeur d'un entier x, le resultat de l'addition est obtenu e*
⇝ *le successeur de l'addition de x avec le second argument*

TODO : Ajouter d'autres exemples de definition inductive

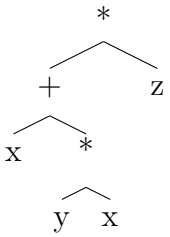
Arbre syntaxique

Comme chaque élément x d'un ensemble defini par induction se construit en appliquant successivement les règles de construction, il est pratique de décrire le procédé qui permet de construire x par un arbre. Cet arbre s'appelle un **arbre syntaxique**.

Les feuilles sont étiquetées par des éléments de base (dont le symbole ne prende pas d'argument) et Les nœuds internes sont étiquetés par des symbols fonctionnels. Le nombre d'argument que de doit prendre un symbol s'appelle l'arité du symbol.

Si dans le processus de construction d'un element x, une règle fait intervenir un symbol d'arité k, le nœud associé aura k descendants (des sous-éléments).

Exemple Par exemple l'élement $(x + (y * x)) * z)$ de l'ensemble des expressions arithmétiques correspond à l'arbre syntaxique suivant :



TODO *ajouter un paragraphe sur les syntaxes abstraite/concrete, les grammaires, BNF, lexeur/parseurs,peg. faire le lien vers du code en annexe*

Sémantique

Les mots d'un langage formel sont constitués de symbole mais n'ont aucun sens en soit ; Pour exprimer le sens des mots (suite de symbol) il faut les interpréter dans un ensemble particulier.

TODO exemple de la sémantique des expression arithmétiques,

2.1.2 Logique formelle

Par la suite, nous allons introduire différents formalismes logique que nous étendrons progressivement.

Pour chacun d'eux on définira formellement un **langage** dont les mot seront appelés **formules**. A chaque fois la grammaire (règles de formation des mots) des formules sera décrite par une définition inductive.

Les formules sont constituée de symbole et n'ont aucun sens en soit ; On se donnera les moyens d'exprimer le sens des formules en les interpretant dans un ensemble particulier. On introduira aussi des système de déduction, c'est-à-dire des ensembles de règles de transformation (syntaxiques) des formules. Ces règles se baseront uniquement sur la forme (structure) des formules. On aura par exemple une règle disant que pout toute formule Φ et Ψ si l'on sait démontrer que Φ implique Ψ (on écrira $\Phi \Rightarrow \Psi$), et que l'on sait démontrer Φ alors on sait démontrer ψ.

Pour chaque formalisme on aura donc toujours deux façons de voir les choses

1. Le coté sémantique avec des **interprétations** permettant de donner du sens aux formules et définir les formules "vrai" (On verra plusieurs notion de vérité, le terme vrai sera par la suite limité au formule vrai dans toute interpretation)

2. Le coté syntaxique avec des **systèmes de déduction**, c'est-à-dire des systèmes de regle de transformation permettant de construire des **preuves** de formules. Cela permet de définir les formules prouvables.

Remarque 4. *Pour définir un langage on utilise aussi un langage ; On l'appel le* ***meta-langage*** *(pour nous ici le français et les mathématiques usuelles).*

Lorsqu'on définie un formalisme logique il est important de s'assurer qu'il vérifie un certain nombre de propriétés. L'etude de ces propriétés s'appelle la **méta-thérorie**.

L'etablissement de la méta-thérorie concerne le concepteur de langage (donc le logicien ou le mathématicien mais aussi l'informaticien) et dépasse le cadre de ce cours mais la connaissance de certaines propriétés est aussi fondamentale pour l'utilisateur des formalismes.

Les propriétés fondamentales sont

— La **cohérence** : une thèorie est dite **contradictoire** si toute formule est valide/prouvable. Sinon la logique est dite cohérente.
Notons qu'une théorie contradictoire n'est pas vraiment utile puisqu'on peut y prouver tout et son contraire (c'est-à-dire une formule et sa négation) !

— La **correction** (soundness) : toute formule prouvable est elle « vrai » ?

La encore un système de déduction permettant de prouver des formules non valides est a éviter !

— La **complétude** : toute formule « vrai » est elle « prouvable » ?

L'absence de complétude exprime juste qu'il existe des formules « vrai » non démontrable dans le formalisme. Cela pause une limite théorique mais n' empêche pas l'utilisation de formalisme dans la pratique.

— La **décidabilité** : existe-t'il un algorithme pour décider si une formule est « vrai » ou non ?
Attention, il s'agit d'un algorithme prenant en entrée une formule et renvoyant "vrai" ou "non vrai". Le fait de ne pas avoir la décidabilité ne dit pas qu'on ne sais jamais décider si on formule donnée est « vrai » ou non, il dit simplement que l'on n'a pas de moyen **générique** de le faire. On a donc recours à des heuristiques.

Même lors qu'on dispose d'un algorithme de décidabilité, son utilisation pratique depend de sa complexité. Si le temps de réponse croît exponentiellement avec la taille des données il devient vite inutilisable.

Chapitre 3

Le langage du Calcul des propositions

Dans ce chapitre, nous allons définir précisément comment écrire des formules (la syntaxe et la grammaire) du **Calcul des propositions**, quel est leur sens (la sémantique), c'est-à-dire quel est leur statut par rapport à la vérité, puis nous introduirons différents formalismes de raisonnement pour faire des preuves formelles sur ces formules.

3.1 Construire des formules bien formées

Les formules du calcul des propositions sont des mots qui respectent un certain nombre de règles.

3.1.1 L'alphabet

Les formules sont des mots qui sont faits en utilisant un jeu de symboles précis. C'est l'alphabet du Calcul des propositions.

Cet alphabet est le suivant :

1. un ensemble de *connecteurs* : $\{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$
2. un ensemble de symbole de ponctuation : les parenthèses ouvrantes et fermantes
3. un ensemble de *variables propositionnelles* aussi appelées propositions atomiques et généralement écrites comme des lettres (ou des suites de lettre et de chiffres) comme cela se fait pour les identifiants dans les langages de programmation.

Par exemple : $XX\Rightarrow\neg\vee A(\wedge BB($ est un mot fabriqués à partir de l'alphabet du calcul des propositions, $XX\vee(A\wedge XX)$ aussi alors que $XX!Y\forall\vee$ ne l'est pas : ! et $\forall$ ne sont pas des symboles de l'alphabet.

3.1.2 La grammaire

Connaitre l'alphabet avec lequel on peut former des formules ne suffit pas. $XX\Rightarrow\neg\vee A(\wedge BB($ n'est pas une formule, alors que $(XX\vee(A\wedge XX))$ en est une.

Il faut aussi avoir des règles, une grammaire, qui précisent les règles de formation des mots.

La grammaire du Calcul des propositions se résume aux deux règles suivantes :

1. Les variables propositionnelles sont des formules.
2. Si A et B sont des formules alors $(A\vee B)$, $(A\wedge B)$, $(A \Rightarrow B)$, $(A \Leftrightarrow B)$, $\neg A$ et (A) sont des formules.

La première règle nous permet d'affirmer que XX, A BB sont des formules, puisque ce sont des variables propositionnelles.

De cela plus la deuxième règle, on obtient que $(A\wedge XX)$ est une formule. Par une nouvelle application de la règle 2 on obtient alors que $(XX\vee(A\wedge XX))$ en est une.

En revanche , aucune suite d'applications des règles 1 et 2 ne permet d'établir que $XX \Rightarrow \neg\vee A(\wedge BB($ est une formule.

Remarque 5. *L'appartenance à l'ensemble des formules est décidable (ie il existe un algorithme permettant de décider si un mot quelconque est une formule)*

Ces règles nous disent qu'une formule est soit une variable propositionnelle, soit un connecteur binaire $\wedge, \vee, \Rightarrow, \Leftrightarrow$ appliqué à deux formules, soit le connecteur $\neg$ appliqué à une formule.

Comme nous l'avons vu au chapitre precedent, mathématiquement, on resume par la **definition inductive** :

Définition 5. *L'ensemble des formules de la logique propositionnelle, est le plus petit ensemble de mot construits sur l'alphabet $A = V \cup \{\wedge, \vee, \neg, \Rightarrow\} \cup \{(,)\}$ (ou V est un ensemble dénombrable de variable propositionnelle, souvent notées $\{A, B, C \ldots\}$) vérifiant les règles suivantes :*

1. *Les variables propositionnelles sont des formules.*

2. *Si A et B sont des formules alors $(A \vee B)$, $(A \wedge B)$, $(A \Rightarrow B)$, $(A \Leftrightarrow B)$ et $\neg A$ sont des formules.*

La encore le fait que l'ensemble soit défini par induction nous permet de raisonner par induction pour démontrer des propriétés des formules.

Pour montrer une propriété P sur les formules, il suffira de montrer :

1. qu'elle est vérifiée par les variables proportionnelles
2. si elle est vérifiée par une formule F alors elle est vérifié par la formule $\neg F$.
3. si elle est vérifié par deux formule F et G F alors elle est vérifié par la formule $(F \wedge G)$, par $(F \vee G)$, $(F \Rightarrow G)$.

On peux par exemple montrer le *(méta-)théorème* suivant :

Theorem 3.1.1. *« toute formule contient autant de parenthèses ouvrantes "(" que de parenthèses fermantes ")" ».*

Démonstration. Soit $P(\_)$ la propriété exprimant qu'« une formule passée en argument a autant de parenthèses ouvrantes "(" que de parenthèses fermantes ")" ».

Montrons que l'on a $P(F)$ pour toute formule F :

1. si F est une proposition, il n'y a pas de parenthèse donc, on a $P(F)$
2. si F est une formule de la forme $\neg G$, si on a $P(G)$ alors ajoute $\neg$, n'ajoute pas de parenthèse et on a donc $P(F)$
3. si F est une formule de la forme $(F_1 \wedge F_2)$, et que l'on a $P(F_1)$ et $P(F_2)$, alors la construction de $(F_1 \wedge F_2)$ ajoute une parenthèses ouvrantes "(" que de parenthèses fermantes ")", et donc F a toujours autant de parenthèses ouvrantes que de parenthèses fermantes. On bien $P(F)$.
4. même chose pour $\vee$, $\Rightarrow$ et $\Leftrightarrow$

□

Nous ne ferons pas beaucoup de *(méta-)théorie* dans ce cours mais sachez que c'est aussi par induction que l'on montre par exemple la propriété de sureté d'un systeme de déduction.

3.2 Le sens des formules

Le sens d'une formule est en quelques sortes l'ensemble des mondes qui la rendent « *vraie* ».

Si tous les mondes la rendent vraie, cette formule est *valide*. On dit aussi que c'est une *tautologie*.

Si certains mondes seulement la rendent vraies, cette formule est *satisfiable*. On dit aussi *réalisable*.

Si aucun monde la rend vraie, cette formule est *insatisfiable*, on dit aussi que c'est une *antilogie*.

3.2.1 Les variables propositionnelles

Les variables propositionnelles ne sont que satisfiables (réalisables). Une variable donnée, A peut toujours être interprétée par *un fait vrai* , par exemple « Zola est écrivain », ou par *un fait faux* : « Zola est cinéaste. »[1]

La seul chose qui nous interesse pour les variables propositionelle, c'est la valeur de vérité de celles qui apparaissent dans nos formules. Choisir des valeurs de vérité pour ces variables propositionnelles, c'est *choisir un monde* particulier.

Ce choix d'un monde correspond à definir une **fonction** de l'ensemble des variables propositionnelles vers l'ensemble des booléens $\mathbb{B} = \{vrai, faux\}$. On appelle cette fonction **une valuation** et note souvent $\rho : V \to \mathbb{B}$. On peut aussi representer cette fonction par **son graphe**, c'est-à-dire un ensemble de couple (variable,valeur), on parle alors parfois **d'environnement**.

3.2.2 Les formules complexes

La valeur de vérité d'une formule complexe où apparait des connecteurs, dépend de la valeur de vérité des variables propositionnelles et du sens du connecteur.

Par exemple : $A \wedge B$ est vraie dans un monde ou A est vraie et B est vraie. En interprétant A par « Zola est écrivain » et B par « Flaubert est écrivain », cette formule est vraie. En interprétant A par « Zola est écrivain » et B par « Flaubert est cineaste », cette formule est fausse. Ainsi, $(A \wedge B)$ est une formule satisfiable mais pas valide.

L'ensemble des mondes qui rendent vraies cette formule est l'ensemble des mondes qui rendent vraies A et B à la fois.

Une formule réalisable (satisfiable) décrit (caractérise, réalise) un **ensemble de monde restreint** : ceux qui la rendent vraies.

1. Les langues naturelle contienne aussi des phrase qui ne peuvent etre ni vrai ni fausse. Par exemple la phase : « Cette phrase est fausse ». On se méfiera de ces phrases autoréférente et on évitera de les representer par une proposition

Le connecteur $\wedge$ est défini en lui donnant un sens ; c'est-à-dire que à interpretation fixée des formules F et G qu'il relie, on sait déterminer la valeur de vérité de la formule $F \wedge G$.

La definition d'un **connecteur binaire** ($\wedge$, $\vee \Rightarrow$) est donc une **fonction** prenant **en entrée deux valeurs de vérité** (issues de l'interpretation des deux formules gauche et droite) et **renvoie une nouvelle valeur de vérité** (celle de la formule composée en les combinant avec le connecteur).

La définition de chaque connecteur est donnée par la table de vérité de chacuns des connecteurs.

En revanche $A \vee \neg A$ est valide : quelquesoit le sens que l'on donne à A, cette formule est vraie. L'ensemble des mondes qui la rendent vraie est l'ensemble de tous les mondes possibles. Cette formule n'impliquent aucune contrainte sur l'univers, elle decrit tous les mondes.

Pour finir $A \wedge \neg A$ est une formule insalifiable : Aucun monde ne peut rendre vraies à la fois une chose et son contraire. Elle ne parle d'aucun monde.

3.2.3 Tables de vérité des connecteurs

Une formule de la forme $\neg s$ est vraie si et seulement si s est fausse

s	($\neg$ s)
v	f
f	v

Cette table décrit le graphe d'une fonction qu'on pourrait écrire en pseudo code

```
function EvalNot(v){
  if (v == true)     {return false}
  if (v == false)    {return true}
}
```

Une formule de la forme $s \wedge t$ est vraie si et seulement si s est vraie, et t est vraie.

s	t	(s $\wedge$ t)
v	v	v
v	f	f
f	v	f
f	f	f

Cette table décrit le graphe d'une fonction qu'on pourrait écrire en pseudo code

```
function EvalEt(v1,v2){
  if (v1 == true && v2 == true)     {return true}
  if (v1 == true && v2 == false)    {return false}
  if (v1 == false && v2 == true)    {return false}
  if (v1 == false && v2 == false)   {return false}
}
```

qui peut évidement se simplifier en

```
function EvalEt(v1,v2){
  if (v1 == true && v2 == true){
     return true
  }else{
     return false
  }
}
```

Une formule de la forme $s \vee t$ est vraie si et seulement si s est vraie, ou t est vraie ou les 2.

s	t	(s ∨ t)
v	v	v
v	f	v
f	v	v
f	f	f

Cette table décrit le graphe d'une fonction qu'on pourrait écrire en pseudo code

```
function EvalOu(v1,v2){
  if (v1 == true && v2 == true)     {return true}
  if (v1 == true && v2 == false)    {return true}
  if (v1 == false && v2 == true)    {return true}
  if (v1 == false && v2 == false)   {return false}
}
```

qui peut évidement se simplifier en

```
function EvalEt(v1,v2){
  if (v1 == false && v2 == false){
     return false
  }else{
     return true
  }
}
```

Une formule de la forme $s \Rightarrow t$ est fausse si et seulement si s est vraie et t est fausse.

s	t	(s ⇒ t)
v	v	v
v	f	f
f	v	v
f	f	v

Cette table décrit le graphe d'une fonction qu'on pourrait écrire en pseudo code

```
function EvalOu(v1,v2){
  if (v1 == true && v2 == true)     {return true}
  if (v1 == true && v2 == false)    {return false}
  if (v1 == false && v2 == true)    {return true}
  if (v1 == false && v2 == false)   {return true}
}
```

qui peut évidement se simplifier en

```
function EvalEt(v1,v2){
  if (v1 == true && v2 == false){
     return false
  }else{
     return true
  }
}
```

Une formule de la forme $s \Leftrightarrow t$ est vraie si et seulement si s et t sont vraies en même temps , ou bien fausses en même temps.

s	t	(s ⇔ t)
v	v	v
v	f	f
f	v	f
f	f	v

3.2.4 Calculer la valeur de vérité d'une formule

À partir de la table de vérité des connecteurs, il est possible de calculer la valeur de vérité d'une formule toute entière :
exemple :

i_1	s	t	(s ∨ t)	(t ∧ s)	(s ∨ t) ⇒ (t ∧ s)
i_1	v	v	v	v	v
i_2	v	f	v	f	f
i_3	f	v	v	f	f
i_4	f	f	f	f	v

Chaque ligne correspond à une interprétation possible. Si on obtient vrai a chaque ligne, la formule est une tautologie, Si on obtient faux a chaque ligne, la formule est insatisfiable c'est une antilogie, Sinon, la formule est satisfiable ou réalisable.

3.3 Modéliser

Retournons à ce qui nous occupe : nous voulons utiliser ce langage, pour spécifier des problèmes, des systèmes informatiques .... Spécifier quelque chose , c'est décrire les mondes où ce quelque chose est vrai.

C'est donc écrire des formules decrivant ces mondes.

Chacune de ces formules n'ont aucune raison d'être valide. Bien au contraire, leur rôle est de restreindre l'ensemble des mondes possibles, à ceux où ce dont on veut rendre compte est vrai. Les formules constituant notre spécification seront donc par essence réalisables.

Aucun outil formel ne pourra jamais permettre de prouver qu'une spécification rend bien compte de ce qu'elle est censée modéliser. La correction du passage de l'informel au formel ne pourra jamais etre objet de preuve. D'où la necéssité absolue de maitriser parfaitement ce langage pour être capable de dire ce que l'on veut dire.

En revanche, une fois que l'on tient une spécification d'un problème, il est possible de raisonner dessus de façon rigoureuse. On peut déduire formellement des propriétés. On pourra démontrer que des proprétés sont des conséquences logiques des formules constituant notre spécification, en montrant que ces propriétés sont vraies dans tous les modèles de notre spécification. On peut même montrer (mais il faut un langage plus puissant que la logique des proposition) qu'un programme vérifie bien ses spécifications.

3.4 Exercices

Exercice 1. *On juge un homme accusé d'avoir trempé dans un cambriolage. Le procureur et l'avocat parlent tour à tour :*

Le procureur : "Si l'accusé est coupable, alors il a un complice".
L'avocat : "C'est faux !"

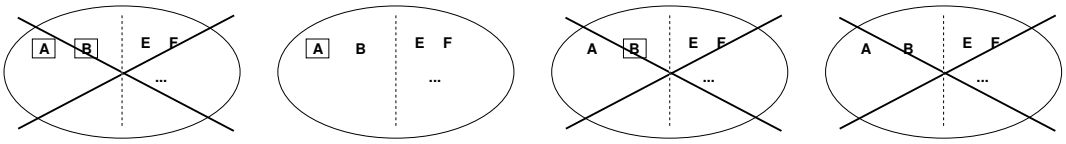
Pourquoi est ce la pire des choses que puisse dire l'avocat ?

Le procureur dit en fait : l'accusé n'a pas agit seul. Donc l'avocat dit : si

il à agit seul. Donc il dit qu'il est coupable.

Formalisation :
Le procureur dit : $A \Rightarrow B$
L'avocat dit : $\neg(A \Rightarrow B)$

Or , les modèles de $\neg(A \Rightarrow B)$ *sont les mondes qui rendent faux* $(A \Rightarrow B)$. *La seule façon de rendre faux* $A \Rightarrow B$ *est d'avoir A vrai et B faux, c'est à dire A coupable et B innocent*

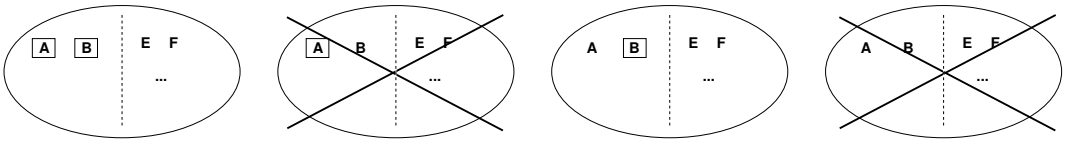


Exercice 2. *Supposons que les affirmations suivantes soient vraies :*

1. *J'aime Elisabeth ou j'aime Jeanne*
2. *Si j'aime Elisabeth , alors j'aime Jeanne*

Que peut on en déduire ?

Formalisation :
(1) $A \vee B$
(2) $A \Rightarrow B$



Dans tous les modèles de ces 2 formules B est vraie. J'aime Elizabeth est donc une conséquence de ces 2 formules.

Exercice 3. *Durant une enquète on fait le raisonnement suivant :*

1. *Si l'accusé est coupable, il etait a Paris au moment du crime*
2. *Or il était a Marseille*
3. *Donc il n'est pas coupable*

Peut on vraiment déduire qu'il n'est pas coupable ? Justifier votre réponse par une formalisation et des modèles.

Exercice 4. *Supposons qu'on me demande : "Est il vrai que si tu aimes Eve alors tu aimes Marguerite aussi" ? et que je reponde : "Si c'est vrai, alors j'aime Eve".*

Qu'en déduisez vous ?

Pour que mon affirmation soit vraie il faut soit que si j'aime E alors j'aime M et que j'aime Eve, soit que si j'aime Eve alors j'aime M soit faux c'est a dire que j'aime E et pas M. J'aime donc E dans tous les cas

Formalisation :
Notons
— *E : "j'aime Eve"*
— *M : "j'aime Margerite"*
$(E \Rightarrow M) \Rightarrow E$

3.5 Le coté syntaxique et la démonstration

L'interpretation permet donc de verifier la véracité d'une formule ou de verifier qu'une formule est conséquence d'une autre. Mais dans la vie courante (ou en mathématique), quand on veut convaincre de la véracité de quelque chose on procède rarement en énumérant toutes les mondes possibles et en vérifiant que cela est vrai dans tous les mondes.

Par exemple, si l'on sais que :

1. « si j'ai mangé alors je n'ai pas faim »
2. » si je n'ai pas faim de dors bien »
3. « j'ai mangé »

qu'on peut modeliser par

1. $M \Rightarrow F$
2. $F \Rightarrow D$
3. M

Pour affirmer qu'alors « je dors bien » vous pouvez utiliser une methode sémantique ; c'est-à-dire montrer

$$M \Rightarrow F, F \Rightarrow D, M \models D$$

, ou ce qui est équivalent (FIXME ref-theorem) que

$$(((M \Rightarrow F) \wedge ((F \Rightarrow D) \wedge M)) \Rightarrow D)$$

est une tautologie en faisant la table de vérité.

Mais si vous voulez convaincre votre grand mère il y a plus de chance que vous disiez quelque chose, comme « j'ai mangé, donc (par l'affirmation 1) je n'est pas faim, donc (par l'affirmation 2) je dors bien. ».

C'est-à-dire, de M et $M \Rightarrow F$ on déduit F, de F et $F \Rightarrow D$ on déduit D

On a ici appliqué deux fois la même figure (*règle*) de raisonnement qui exprime que quelque soit les formules ϕ et ψ, de ϕ et $\phi \Rightarrow \psi$ on déduit ψ.

C'est pour exprimer cette façon de raisonner/déduire, on parle alors de raisonnement déductif, *à partir de règles* qu'est apparu le concept de preuve formelle et la notion de *théorie de la preuve*.

Dans cette approche, on manipule les formules en se basant leur **forme** (la **syntaxe**) afin de construire *une preuve* formelle en utilisant **des règles d'inferences** (de raisonnement) .

L'ensemble des manipulations possibles est décrit par *un système de déduction*

Un système de déduction est définie par les données suivante :

1. Des axiomes
2. Des règles

Les axiomes sont les briques de base du raisonnement ; c'est à partir d'eux qu'au moyen des regles on construit de nouveaux resultats.

Il y a deux façons d'utiliser ces systèmes, en avant (*forward proof*) ou en arrière *backward proof*.

- Dans **une preuve en avant** on part des axiomes et on les transforme à partir des règles pour obtenir de nouveaux résultats (pas forcément interessant). c'est aussi comme cela que l'on procède pour verifier une preuve donnée.
- Dans **une preuve en arrière** on part d'une formule [2] que l'on cherche a prouver, on l'appelle le **but**. Si le but à prouver correspond a un axiome, on a fini, sinon on applique une règle pour transformer le **but** en un ensemble de nouveaux buts, les**sous-but**. Si l'ensemble est vide le **but** précedent est prouvé, sinon on poursuit (récursivement) le même processus sur les sous-buts restants.

Dans tout les cas, faire la preuve c'est construire un arbre (de preuve) dont les feuilles sont des axiomes et les noeuds sont des règles

Il existe différent systèmes de déduction, notament :

- Les systèmes à la Frede / Hilbert
- La déduction Naturelle

2. ou une formule dans un certain contexte

— La calcul des séquents

— ...

Après une très courte presentation du premier nous nous focaliserons sur la **Déduction Naturelle**

La déduction à la Frede / Hilbert

Elle ce caractérise par un grand nombre d'axiomes logiques exprimant les principales propriétés de la logique. On combine ces axiome au moyen de quelques règles, notamment **la règle de modus ponens**, pour dériver de nouveaux théorèmes.

Les systèmes *à la Frede / Hilbert*, sont les premiers systèmes déductifs formels, avant la déduction naturelle ou du calcul des séquents de Genzen.

On y introduit une **Familles d'axiomes**. c'est-à-dire des axiomes paramètrés ou les paramètres Φ , Ψ et θ peuvent être remplacés par n'importe quelle formule.

— Familles d'axiomes (pour toutes formules Φ , Ψ et θ)
  — Pour l'implication [3]
    — K : $\Phi \Rightarrow (\Psi \Rightarrow \Phi)$
    — $S : (\Phi \Rightarrow (\Psi \Rightarrow \theta)) \Rightarrow ((\Phi \Rightarrow \Psi) \Rightarrow (\Phi \Rightarrow \theta))$
  — Pour la conjonction
    — $\Phi \Rightarrow (\Psi \Rightarrow \Phi \wedge \Psi)$
    — $\Phi \wedge \Psi \Rightarrow \Phi$ et $\Phi \wedge \Psi \Rightarrow \Psi$
  — Pour la disjonction
    — $\Phi \Rightarrow \Phi \vee \Psi$ et $\Psi \Rightarrow \Phi \vee \Psi$
    — $\Phi \vee \Psi \Rightarrow ((\Phi \Rightarrow \theta) \Rightarrow ((\Psi \Rightarrow \theta) \Rightarrow \theta))$ (par cas)
  — $\perp$ et négation
    — $\perp \Rightarrow \Phi$
    — $\Phi => ((\neg\Phi \Rightarrow \perp)$ et $\neg\Phi \Rightarrow (\Phi \Rightarrow \perp)$

Puis des **règles de déduction**

— $\dfrac{F \Rightarrow G \quad F}{G}\ MP$

— $\dfrac{}{A}\ Ax$ ou A est une instance d'un des schéma d'axiome précédent

Essayons maintenant d'utiliser ce système pour prouver un resultat qui parait trivial : $p \Rightarrow p$:

FIXME : découper et expliquer la preuve.

3. Les nom K et S viennent d'un autre formalisme, *la logique combinatoire* `https://fr.wikipedia.org/wiki/Logique_combinatoire`

$$\dfrac{\dfrac{\dfrac{}{(p\Rightarrow((p\Rightarrow p)\Rightarrow p))\Rightarrow((p\Rightarrow(p\Rightarrow p))\Rightarrow(p\Rightarrow p))}\,Ax \quad \dfrac{}{(p\Rightarrow((p\Rightarrow p)\Rightarrow p)}\,Ax}{p\Rightarrow(p\Rightarrow p))\Rightarrow(p\Rightarrow p)}\,MP \quad \dfrac{}{p\Rightarrow(p\Rightarrow p)}\,Ax}{p\Rightarrow p}\,MP$$

Il apparait immédiatement que faire des preuves dans un tel système est extrêmement pénible et inefficient. Par contre un tel système est utile au niveau méta-théorique

On peut par exemple prouver :

— La **correction** : toute formule démontrable dans le système de Hilbert est une formule valide. On note cela :

$\vdash_H F$ implique $\models F$

— La **complétude** : toute formule valide est démontrable dans le système de Hilbert. On note cela :

$\models F$ implique $\vdash_H F$

La déduction naturelle)

Nous introduisons maintenant la déduction naturelle. Nous avons choisie de la presenter en considérant des formules dans un certain contexte lui meme exprimé par un ensemble de formule.

Pour cela on introduit la notion de **jugement**.

On note $\Gamma \vdash \phi$, pour exprimer, "On peut démontrer ϕ à partir des formule de Γ".

On appel Γ le contexte (ou environnement), c'est un ensemble d'hypothèses qui caractérise (de facons syntaxique) le monde dans lequel on fait la preuve de ϕ.

Il peut être vide lorsque l'on prouve une formule sans hypothese complementaire.

On aura qu'un seul axiome qui exprimera que qu'un resultat qui est déjà dans le contexte est prouvé.

On aura un ensemble es règles de la forme

$$\dfrac{ensemble\ de\ premisses}{conclusion}\ \textsc{Nom de regle}$$

dans lesquelles prémisses et conclusion : sont des jugements.

On construira alors (des arbres) de preuves en combinant les règles.

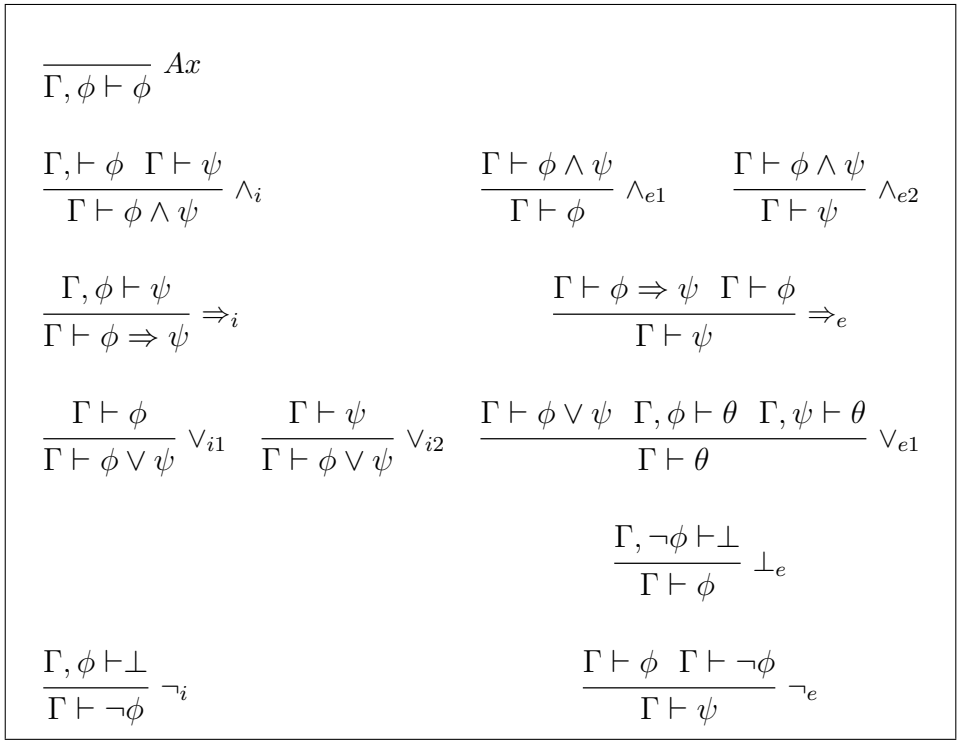


FIGURE 3.1 – règles de la déduction naturelle

La figure 3.1 liste les règles qui sont ensuite expliquées une à une dans le paragraphes suivants.

Explication/Justification des règles de déduction naturelle

Axiome

$$Ax \quad \frac{}{\Gamma, \phi \vdash \phi}\ Ax$$

Elle exprime simplement que si la formule cherchée est déjà dans les hypothèses alors on en a une preuve.

Règles d'introduction Les règles d'introduction doivent leur nom au fait que lors d'une application en avant (de haut en bas), elle créent **une** formule avec un connecteur de plus (il a été introduit).

$$\wedge_i \quad \frac{\Gamma, \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge_i$$

De haut en bas : Si sous les hypothèses Γ (ensemble de formule), on sait prouver un formule ϕ et sous les hypothèses Γ on sait prouver un formule ψ alors sous les hypothèses Γ on sait prouver $\phi \wedge \psi$.
De bas en haut : Pour prouver $\phi \wedge \psi$ sous les hypothèses Γ, il suffit de prouver d'un coté ϕ sous les hypothèses Γ, de l'autre ψ sous les hypothèses Γ,

$\Rightarrow_i$ $\dfrac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \Rightarrow \psi} \Rightarrow_i$

De haut en bas : Si sous les hypothèses Γ et ϕ on sait prouver ψ alors sous les hypothèses Γ (sans ϕ) on sait prouver ψ. Autrement dit si **en supposant** ϕ on sait prouver ψ alors sans cette hypothèses on sait prouver $\phi \Rightarrow \psi$.

De bas en haut : Pour prouver $\phi \Rightarrow \psi$ sous les hypothèses Γ, **supposons** ϕ (*i.e.* ajoutons le à l'ensemble γ d'hypothèses) et prouvons ψ.

$\vee_{i1}$ **et** $\vee_{i2}$ $\dfrac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi} \vee_{i1}$ $\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi} \vee_{i2}$

De haut en bas : Si sous les hypothèses Γ on sait prouver la formule ϕ (respectivement ψ) alors sous les hypothèses Γ on sait prouver la formule$\phi \vee \psi$.
De bas en haut : Pour prouver $\phi \vee \psi$ sous les hypothèses Γ, il suffit de prouver soit ϕ soit ψ sous les hypothèses *Gamma*.

$\neg_i$ $\dfrac{\Gamma, \phi \vdash \bot}{\Gamma \vdash \neg\phi} \neg_i$

De haut en bas : Si les hypothèses Γ et ϕ mènent à une contradiction (*i.e.* on sait prouver la formule $\bot$) alors sous les hypothèses Γ on sait prouver la formule$\neg\phi$.

De bas en haut : Pour prouver $\neg\phi$ sous les hypothèses Γ, il suffit de prouver qu'ajouter ϕ aux hypothèses *Gamma* conduit à une contradiction (*i.e.* on sait prouver la formule $\bot$).

Règles d'élimination .

Les règles d'élimination doivent leur nom au fait que lors d'une application en avant (de haut en bas), elle créent une formule avec un connecteur de moins (il a été éliminé).

$\wedge_{e1}$ **et** $\wedge_{e2}$ $\dfrac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi} \wedge_{e1}$ $\dfrac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi} \wedge_{e2}$

De haut en bas : Si sous les hypothèses Γ on sait prouver la formule $\phi \Rightarrow \psi$ et que sous les hypothèses Γ on sait prouver la formule ϕ alors sous les hypothèses Γ on sait prouver la formule ψ.

De bas en haut : Pour prouver une formule ψ sous les hypothèses Γ, il suffit de prouver formule ($\phi \wedge \psi$.

$$\Rightarrow_e \quad \frac{\Gamma \vdash \phi \Rightarrow \psi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi} \Rightarrow_e$$

C'est de nouveau le *modus ponens*.

De haut en bas : Si sous les hypothèses Γ on sait prouver la formule$\phi \wedge \psi$ alors sous les hypothèses Γ on sait prouver la formule ϕ (respectivement ψ).

De bas en haut : Pour prouver une formule ψ sous les hypothèses Γ, il suffit de prouver d'un coté la formule $\phi \Rightarrow \psi$ sous les hypotheses Γ, de l'aute la formule ϕ sous les hypothèses Γ.

$$\vee_{e1} \quad \frac{\Gamma \vdash \phi \vee \psi \quad \Gamma, \phi \vdash \theta \quad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta} \vee_{e1}$$

Cette règle exprime l'idée qu'il y a 2 cas , décrit par ϕ et ψ et que comme dans chacun des 2 cas on sait prouver θ alors on sait prouver θ dans tout les cas.

De haut en bas : Si sous les hypothèses Γ on sait prouver la formule $\phi \vee \psi$, et en ajoutant ϕ aux hypothèses on sait prouver la formule θ et en ajoutant ψ aux hypothèses on sait prouver la formule θ, alors sous les seules hypothèses Γ on sait prouver la formule θ.

De bas en haut : Pour prouver une formule θ sous les hypothèses Γ, il suffit de prouver la formules $\phi \vee \psi$ sous les hypotheses Γ et de prouver d'un coté θ après avoir ajouter ϕ aux hypothèses Γ, de l'autre θ après avoir ajouter ψ aux hypothèses Γ.

$$\perp_e \quad \frac{\Gamma, \neg\phi \vdash \perp}{\Gamma \vdash \phi} \perp_e$$

Cette règle exprime le raisonnement par l'absurde.

De haut en bas : Si en supposant $\neg\phi$ (*i.e.* en l'ajoutant aux hypothèse Γ) on arrive a une contradiction (*i.e.* une preuve de $\perp$), cela prouve ϕ.

De bas en haut : Pour prouver une formule ϕ, il suffit de supposer $\neg\phi$ (*i.e.* de l'ajouter aux hypothèse Γ) et de montrer qu'on a une contradiction (*i.e.* une preuve de $\bot$).

$$\neg_e \quad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \neg\phi}{\Gamma \vdash \psi}\ \neg_e$$

Cette règle exprime que si on prouver un resultat et son contraire, on peut en déduire n'importe quoi.

De haut en bas : Si sous les hypothèses Γ) on peut prouver une formule ϕ et que sous ces mêmes hypothèses on peut prouver $\neg\phi$ alors on peut prouver n'importe quelle formule ψ.

De bas en haut : Pour prouver une formule ψ sousles hypothèses Γ , il suffit de prouver une formule ψ et sa negation sous ces même hypothèses.

Exemples de preuve en déduction naturelle

Outils : `http://cedric.cnam.fr/~pons/ENSIIE/demo1.html`

$\vdash (A \Rightarrow (B \Rightarrow A))$ FIXME
write the proof
ENDFIXME

$\vdash (A \wedge B) \Rightarrow B \wedge A$ FIXME
write the proof
ENDFIXME

$\vdash (A \vee B) \Rightarrow B \vee A$ FIXME
write the proof
ENDFIXME

$\vdash (A \Rightarrow \bot) \Rightarrow \neg A$

$$\dfrac{\dfrac{\dfrac{}{(A\Rightarrow\bot), A\vdash A}\,Ax \qquad \dfrac{}{(A\Rightarrow\bot), A\vdash (A\Rightarrow\bot)}\,Ax}{(A\Rightarrow\bot), A\vdash \bot}\Rightarrow_e}{(A\Rightarrow\bot)\vdash \neg A}\,\neg_i}{\vdash (A\Rightarrow\bot)\Rightarrow\neg A}\,\Rightarrow_i$$

On montre de même : $\vdash \neg A \Rightarrow (A \Rightarrow \bot)$

Donc $\neg\phi$ abréviation pour $\phi \Rightarrow \bot$.

$\vdash (A \vee \neg A)$ Avec les règles que nous avons donnée, $\vdash (A \vee \neg A)$ n'est pas un théorème de base. Sa preuve peut évidemment etre construite mais n'est pas intuitive.

FIXME
écrire la preuve
ENDFIXME

hautre notation :

$$\begin{array}{ll} \vdash (A \Rightarrow \bot) \Rightarrow \neg A & \Rightarrow_i \\ (A \Rightarrow \bot) \vdash \neg A & \neg_i \\ (A \Rightarrow \bot), A \vdash \bot & \Rightarrow_e (1)(2) \\ (1)(A \Rightarrow \bot), A \vdash A & Ax \\ (2)(A \Rightarrow \bot), A \vdash A \Rightarrow \bot & Ax \end{array}$$

Règles dérivées :

— règle de coupure

$$\frac{\Gamma; A \vdash B \quad \Gamma \vdash A}{\Gamma \vdash B}\ coupure$$

— preuve :

$$\frac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow_i \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow_e$$

— ou

$$\begin{array}{ll} \Gamma \vdash B & \Rightarrow_e (1)(2) \\ (1)\Gamma \vdash A \Rightarrow B & \Rightarrow_i \\ \Gamma, A \vdash B & \text{prémisse gauche} \\ (2)\Gamma \vdash A & \text{prémisse droite} \end{array}$$

3.6 Exercice

Un Rapport (évidemment très neutre ;-) sur la reforme des retraites prévoit que :

1. Si la reforme n'est pas équitable, elle devra est promue par un intense matraquage de "communication".
2. si le Medef s'en mêle, la reforme ne sera pas équitable.

3. Si la reforme est équitable, il n'y aura pas de mouvement social.

La reforme est proposée peut après. Il y a un mouvement social.

Les médias en déduisent que le Medef s'en est mêlé tandis que l'opinion publique pense avoir subit une intense campagne de "matraquage communicatif".

1. Modéliser le problème
2. Montrer que l'opinion publique a raison. (preuve en déduction naturelle).
3. quelles sont les hypothèses utiles
4. Montrer par des valeur de vérités bien choisie que les médias se trompent.

Solution

1. on introduit les propositions atomiques suivantes :
   — E : "la reforme est équitable",
   — C : "il y a une intense campagne de matraquage de communication"
   — M : "le Medef s'en mèle"
   — S : "il y a un mouvement social".
   Le rapport énnonce donc :
   (a) $\neg E \Rightarrow C$
   (b) $M \Rightarrow \neg E$
   (c) $E \Rightarrow \neg S$
   Le fait qu'il y ait un mouvement social s'énonce :
   S
   La conclusion des médias est :

$$(\neg E \Rightarrow C), (M \Rightarrow \neg E), (E \Rightarrow \neg S), S \vdash M$$

   ou ce qui est équivalent :

$$\vdash (\neg E \Rightarrow C) \Rightarrow (M \Rightarrow \neg E) \Rightarrow (E \Rightarrow \neg S) \Rightarrow S \Rightarrow M$$

   La conclusion de l'opinion publique est :

$$(\neg E \Rightarrow C), (M \Rightarrow \neg E), (E \Rightarrow \neg S), S \vdash C$$

   ou ce qui est équivalent :

$$\vdash (\neg E \Rightarrow C) \Rightarrow (M \Rightarrow \neg E) \Rightarrow (E \Rightarrow \neg S) \Rightarrow S \Rightarrow C$$

2. Le "script de preuve" (ie. la suite de regle a appliquer) est la suivante :

```
(~E=>C)=>(M=>~E)=>(E=>~S)=>S=>C
IntroImp
IntroImp
IntroImp
IntroImp

ElimImp ~E
 Axiom

 IntroNoT
 elimNot S
   Axiom

   ElimImp E
     Axiom

     Axiom
```

3. La preuve n'ulise pas le fait que $M \rightarrow E$ les autres sont utiles.
4. On peut construire la table de vérité de la formule

$$(\neg E \Rightarrow C) \Rightarrow (M \Rightarrow \neg E) \Rightarrow (E \Rightarrow \neg S) \Rightarrow S \Rightarrow M$$

mais ce n'est pas nécessaire on cherche a montrer que cette formule n'est pas une tautologie. ie qu'il existe un choix v de valeurs de vérité tel que $v[\neg E \Rightarrow C] = v[M \Rightarrow \neg E] = v[E \Rightarrow \neg S] = v[S] = 1$ et $v[M] = 0$
On a donc les contrainte $v[M] = 0$ (ce qui entraîne $v[M \Rightarrow \neg E] = 1$) $v[S] = 1$, $v[\neg E \Rightarrow C] = v[E \Rightarrow \neg S] = 1$
Pour avoir $v[E \Rightarrow \neg S] = 1$ avec $v[S] = 1$ il faut avoir $v[E] = 0$. Comme on doit aussi avoir $v[\neg E \Rightarrow C] = 1$ in doit avoir $v[C] = 1$
En résumé si $(v[E], v[C], v[M], v[S]) = (0, 1, 0, 1)$ les quatre hypothèses sont vrai mais la conclusion des média est fausse.

Logique propositionnelle (preuve constructive)

— Preuve constructive : n'utilise pas la règle du Tiers Exclu.
— La déduction naturelle constructive est correcte (mais pas complète) pour la logique propositionnelle.

— Exemple de tautologie non démontrable en déduction naturelle constructive :
: formule de Peirce : $(((A \Rightarrow B) \Rightarrow A) \Rightarrow A)$
— Problème :
"Montrer qu'il existe deux nombre irrationnels a et b telque a^b est rationnel".

— Démonstration :

Considérons $\sqrt{3}^{\sqrt{2}}$

— si $\sqrt{3}^{\sqrt{2}} \in \mathbb{Q}$ on pose $a = \sqrt{3}$ et $b = \sqrt{2}$

— si $\sqrt{3}^{\sqrt{2}} \neg \in \mathbb{Q}$ on pose $a = \sqrt{3}^{\sqrt{2}}$ et $b = \sqrt{2}$ et donc $a^b = 3 \in \mathbb{Q}$
Donc a et b existent, mais quelles sont leurs valeurs ?
Notion de témoin.