CHAP 02 Logique - Calcul propositionnel

▸ Compétences

- Comprendre la notion de proposition et de valeur de vérité
- Maîtriser les connecteurs logiques (négation, conjonction, disjonction, ...)
- Construire et exploiter une table de vérité
- Comprendre la notion de réciproque, de contraposée d'une implication
- Utiliser les propriétés du calcul propositionnel (De Morgan, ...)

▸ Fil rouge *coloration drone*

Avant chaque décollage, la station de contrôle vérifie plusieurs conditions sur le drone : niveau de batterie suffisant, conditions météo favorables, zone de vol dégagée, etc
Comment combiner ces conditions à l'aide de connecteurs logiques pour décider, sans ambiguïté, si le décollage est autorisé ?

I. Introduction

▸ Intro

Dans ce chapitre, on présente quelques rudiments de logique mathématique (appelée aussi **logique formelle**), dont l'étude remonte au moins jusqu'à *Aristote*. La connaissance de ces notions sert à :

— comprendre certains raisonnements mathématiques, même à un niveau très élémentaire : beaucoup d'erreurs, en mathématiques, proviennent souvent d'une ignorance des règles de base de la logique ;

— formuler des instructions algorithmiques, même dans un langage de programmation évolué. Là encore, les maladresses de l'informaticien débutant relèvent souvent d'une méconnaissance de la logique.

▸ Point histoire



Aristote (−384 ; −322, 🇬🇷) n'est pas à proprement parler un mathématicien mais par sa vision philosophique des sciences, il a participé au développement des mathématiques.
Dans ses traités, ***Les Topiques*** et ***Les Analytiques***, Aristote expose des réflexions dans le domaine de la logique et du raisonnement. Il y définit le syllogisme, sorte de schéma logique menant à une conclusion à partir de la conjonction de deux propositions considérées comme vraies.

▸ Remarque

La logique formelle étant binaire (on dit aussi : **booléenne**, du nom du mathématicien anglais George Boole (1815–1864), elle est souvent très éloignée du **bon sens** quotidien, et emploie certains *mots* (comme les mots « et », « ou », etc) dans un sens qui n'est pas toujours celui du langage ordinaire.

▸ Point histoire *(Anecdote)*



George Boole (1815-1864, 🇬🇧) enseignait au *Queen's College* à Cork et un jour qu'il faisait le trajet de 3 km à pied depuis son domicile et qu'il pleuvait, il attrapa une pneumonie. Sa femme, pensant qu'un remède doit ressembler à la cause, le mit au lit et lui jeta des bassines d'eau pendant plusieurs jours !

II. Propositions

II.1. Définition

▸ Définition 2.1

Une proposition est une affirmation à laquelle on peut attribuer une unique **valeur de vérité** parmi les deux valeurs possibles : **VRAI** ou **FAUX** (en abrégé : V ou F).

▸ Exemples 2.1

- P : « 9 est un carré parfait »
- Q : « $2^2 + 3^2 = 5^2$ »
- R : « Il pleut »

▸ **Python 2.1**

En ALGO python, on a les valeurs de vérité `True` et `False`.

```
🐍 Début de la console Python 🐍
>>> 2**2 + 3**2 == 5**2
False
🐍 Fin de la console Python 🐍
```

II.2. Remarque et exemples

▸ **Exemples et remarques 2.2**

Par exemple, la phrase « 4 est un nombre réel positif » (symbolisée mathématiquement par « $4 \geqslant 0$ ») est une proposition VRAIE. Par conséquent la phrase « $4 < 0$ » est une proposition FAUSSE

Les valeurs de vérité V ou F sont souvent notées respectivement 1 et 0.

▸ **Idée**

De façon intuitive, une proposition est tout simplement une phrase qui peut être qualifiée de "*vraie*" ou bien de "*fausse*" à l'exclusion de toute autre possibilité.
C'est ce qu'on appelle le principe du *tiers exclu*, qui signifie, littéralement, qu'il n'y a pas de troisième possibilité. La logique formelle présuppose toujours qu'une proposition est soit vraie soit fausse, indépendamment de notre jugement ou de notre perception. C'est déjà un premier point essentiel, qui différencie le "vrai" et le "faux" de la logique formelle des notions de "*vérité*" et de "*fausseté*" de la vie quotidienne, qui sont souvent versatiles et subjectives.

▸ **Remarques 2.1**

- Il existe en mathématiques de nombreuses propositions dont on ignore la valeur de vérité.
- Certaines affirmations, même sémantiquement cohérentes, ne sont pas des propositions. Par exemple, la phrase "tout ce que je dis est faux" n'est pas une proposition, on ne peut pas associer de valeur de vérité à cette affirmation.
- Des phrases impératives ou interrogatives ne sont pas considérées comme des propositions.
- En mathématiques, il existe des propositions que l'on ne démontre pas et qu'on appelle des **axiomes**. Les autres propositions d'une théorie mathématique (celles dont on démontre la vérité) sont appelées théorèmes ou, tout simplement, "propositions" (sous-entendu : propositions vraies).

III. Connecteurs logiques

III.1. Intro

▸ **Cadre**

Étant données des propositions P, Q, R, ..., on peut construire des nouvelles propositions plus complexes en formant des liaisons entre celles-ci à l'aide de signes (qu'on appelle des connecteurs logiques) et de quelques règles logiques. On forme ainsi ce qu'on nomme le "calcul propositionnel" (ou logique des propositions), dont l'usage est fondamental aussi bien en automatique qu'en informatique ou en théorie des langages.

III.2. Négation

▸ **Définition 3.1**

Si P est une proposition, on note $\neg P$ (ou encore $\overline{P}$) (lire « non P », ou « contraire de P ») l'assertion qui est vraie si et seulement si P est fausse.

Licence Ouverte Etalab 2.0

▸ Illustration 3.1

Ce connecteur de négation peut se représenter par une table de correspondance, appelée **table de vérité** :

P	¬P
V	F
F	V

▸ Exemples 3.1

- P : « $x \leqslant 4$ » $\Rightarrow$ ¬P :
- R : « Le soleil brille » $\Rightarrow$ ¬R :
- Q : « Toutes les Ferrari sont rouges » $\Rightarrow$ ¬Q :
- T : « $\pi > 0$ » est VRAIE, donc ¬T est FAUSSE
- Table de vérité de ¬ (¬P)

▸ Python 3.1

En python, l'opérateur de négation est `not`.

```
🐍 Début de la console Python 🐍
>>> not (True) ; not (False) ; not (2**2 + 3**2 == 5**2)
False
True
True
🐍 Fin de la console Python 🐍
```

III.3. Équivalence

▸ Définition 3.2

La proposition P $\Leftrightarrow$ Q (lire "P est équivalent à Q") est la proposition qui est vraie si et seulement P et Q ont la même valeur de vérité.

▸ Illustration 3.2

On a la table de vérité suivante :

P	Q	P $\Leftrightarrow$ Q
V	V	V
V	F	F
F	V	F
F	F	V

▸ Remarque

D'après l'exercice du paragraphe précédent, (¬ (¬P)) $\Leftrightarrow$ P

▸ Propriété 3.1

Pour signifier une équivalence, on utilise souvent les termes "si et seulement si" : P $\Leftrightarrow$ Q se traduit par : "P est vraie" si et seulement si "Q est vraie".
En pratique, P $\Leftrightarrow$ Q peut être compris comme une sorte d'**égalité logique** (résolution d'équations...).

Licence Ouverte Etalab 2.0

▸ Python 3.2

En ALGO python, l'opérateur d'équivalence est PY ==.

```
🐍 Début de la console Python 🐍
>>> True == False ; False == False ; True == True
False
True
True
🐍 Fin de la console Python 🐍
```

III.4. Conjonction

▸ Définition 3.3

Si P et Q sont deux propositions, on note P∧Q (lire "P et Q") la proposition qui est vraie si et seulement si les propositions P et Q sont toutes les deux vraies. Sinon, elle est fausse.

▸ Illustration 3.3

On a la table de vérité suivante :

P	Q	P ∧ Q
V	V	V
V	F	F
F	V	F
F	F	F

▸ Remarque

On peut faire l'analogie entre l'opération ∧ de la logique formelle et le **montage en série** de deux interrupteurs en électricité.

▸ Exemples 3.2

On sait que Jean n'a pas obtenu son permis de conduire et que Sophie a eu son code.
Soit P : "Jean n'a pas obtenu son permis" et soit Q : "Sophie a eu son code". Alors P ∧ Q est VRAIE

Soient A la proposition : "3 > 5" et B : "2 + 4 = 6". Alors A ∧ B est FAUSSE

▸ Python 3.3

En ALGO python, l'opérateur de conjonction est PY and ou PY &.

```
🐍 Début de la console Python 🐍
>>> True and True
True
>>> True and False
False
>>> False and False
False
>>> A = (3 > 5) ; B = (2+4 == 6) ; A and B
False
>>> True & True
True
>>> True & False
False
>>> False & False
False
🐍 Fin de la console Python 🐍
```

III.5. Disjonction

▸ Définition 3.4

Si P et Q sont deux propositions, on note $P \vee Q$ (lire "P ou Q") la proposition qui est fausse si et seulement si P et Q sont toutes les deux fausses. Sinon, elle est vraie.

▸ Illustration 3.4

On a la table de vérité suivante :

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

▸ Remarque

On peut faire l'analogie entre l'opération $\vee$ de la logique formelle et le **montage en parallèle** de deux interrupteurs en électricité.

▸ Idée 3.1

La table de vérité ci-contre montre que $P \vee Q$ est vraie si et seulement si au moins une des deux propositions P et Q est vraie. Il faut donc distinguer l'expression « P ou Q » employée ici (qui n'exclut pas que les deux termes soient vrais) du « ou » du langage ordinaire qui est souvent utilisé dans un sens *exclusif* qu'il n'a pas en logique. Par exemple, dans la vie quotidienne, si un menu indique : « fromage ou dessert », il est implicitement entendu qu'on ne peut pas prendre les deux. On dit que le « ou » logique est *inclusif*.

▸ Exemples 3.3

- $P \vee (\neg P)$ est VRAIE (soit encore $P \vee (\neg P) \Leftrightarrow \mathcal{V}$: principe du tiers exclu, qui est une *tautologie*).
- $P \wedge (\neg P)$ est FAUSSE (soit encore $P \wedge (\neg P) \Leftrightarrow \mathcal{F}$: principe de la non-contradiction).

On démontre facilement ces résultats à l'aide d'une table de vérité.

▸ Python 3.4

En python, l'opérateur de disjonction est `or` ou bien `|`.

```
♣ Début de la console Python ♣
>>> True or True, True | True
(True, True)
>>> True or False, True | False
(True, True)
>>> False or False, False | False
(False, False)
♣ Fin de la console Python ♣
```

III.6. Lois de Morgan

▸ Théorème 3.1

Soient P et Q deux propositions

— $\neg(P \wedge Q) \Leftrightarrow (\neg P) \vee (\neg Q)$ ou encore $\overline{P \wedge Q} \Leftrightarrow \overline{P} \vee \overline{Q}$

— $\neg(P \vee Q) \Leftrightarrow (\neg P) \wedge (\neg Q)$ ou encore $\overline{P \vee Q} \Leftrightarrow \overline{P} \wedge \overline{Q}$

Licence Ouverte Etalab 2.0

▸ **Remarque**

Ces propriétés sont fondamentales pour simplifier le contraire de propositions composées de *conjonctions* et/ou de *disjonctions*.

▸ **Exemples 3.4**

— Démonstration par table de vérité

P	Q	¬P	¬Q	P ∧ Q	¬ (P ∧ Q)	(¬P) ∨ (¬Q)
V	V	F	F	V	F	F
…						
…						

— « Cet arbre est un hêtre ou un charme »

— « J'ai eu la moyenne en Anglais et en Maths »

— $(x < 0)$ ou $(n = 4)$

— (age $>$ 18) et (age $\leqslant$ 25)

— (n est pair) ou ($n = 7$)

— Le bulletin météorologique affirme : « Demain, il pleuvra et il y aura du vent »

— « Je baisserai les impôts ou j'éradiquerai le chômage »

IV. Propriétés

IV.1. Premières propriétés des connecteurs

▸ **Propriétés 4.1**

Soient P, Q et R des propositions :

— $P \wedge Q \Leftrightarrow Q \wedge P$ et $P \vee Q \Leftrightarrow Q \vee P$ — Commutativité

— $P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$ — Double distributivité
$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$

— $P \wedge (Q \wedge R) \Leftrightarrow (P \wedge Q) \wedge R \Leftrightarrow P \wedge Q \wedge R$ — Associativé
$P \vee (Q \vee R) \Leftrightarrow (P \vee Q) \vee R \Leftrightarrow P \vee Q \vee R$

▸ **Remarque - Principe de la dualité**

Les règles de logique, sauf l'involution de la négation, fournissent chacune une paire de propriétés, telle que la deuxième est obtenue de la première en permutant $\wedge$ et $\vee$ ainsi que V et F.
C'est ce qu'on appelle le principe de dualité : à chaque fois qu'on établit une propriété, on établit aussi, du même coup, une propriété duale.

▸ **Propriétés 4.2**

Soit $\mathcal{V}$ une proposition vraie, et $\mathcal{F}$ une proposition fausse. Alors, pour toute proposition P :

— $(P \wedge \mathcal{V}) \Leftrightarrow P$,

— $(P \wedge \mathcal{F}) \Leftrightarrow \mathcal{F}$,

— $(P \vee \mathcal{V}) \Leftrightarrow \mathcal{V}$,

— $(P \vee \mathcal{F}) \Leftrightarrow P$.

On dit que $\mathcal{V}$ est l'élément neutre du connecteur $\wedge$, et l'élément absorbant du connecteur $\vee$.
De même $\mathcal{F}$ qui est l'élément neutre du connecteur $\vee$, et l'élément absorbant du connecteur $\wedge$.

Licence Ouverte Etalab 2.0

IV.2. Implication

▸ Propriété 4.3

La proposition $(\neg P \vee Q)$ [lire "non P ou Q"] est aussi notée $(P \Rightarrow Q)$ [lire "P implique Q" ou encore "Si P est vraie alors Q est vraie"]. Le symbole "$\Rightarrow$" est appelé implication.

▸ Illustration 4.1

Table de vérité :

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

P	Q	$P \Rightarrow Q$
V	V	V
V	F	F
F	V	V
F	F	V

Autrement dit la proposition $P \Rightarrow Q$ est la proposition qui n'est fausse que lorsque P est vraie et Q est fausse.

▸ Vocabulaire 4.1

Dans l'implication $P \Rightarrow Q$, P est une condition **suffisante** de Q et Q est une condition **nécessaire** de P.

▸ Python 4.1

En NSI python, l'implication n'est pas implémentée (enfin si... mais c'est « contre-intuitif »), on peut la « programmer ».

```
🐍 Début de la console Python 🐍
>>> P = True ; Q = True ; print(f"{P} => {Q} : {not P or Q}")
True => True : True
>>> P = True ; Q = False ; print(f"{P} => {Q} : {not P or Q}")
True => False : False
>>> P = False ; Q = True ; print(f"{P} => {Q} : {not P or Q}")
False => True : True
>>> P = True ; Q = True ; print(f"{P} => {Q} : {not P or Q}")
True => True : True
🐍 Fin de la console Python 🐍
```

IV.3. Réciproque et négation d'une implication

▸ Définition 4.1

L'implication "$Q \Rightarrow P$" est la réciproque de l'implication "$P \Rightarrow Q$".

▸ Propriété 4.4

En sachant que "$Q \Rightarrow P \Leftrightarrow (\neg P \vee Q)$", on obtient directement que "$\overline{Q \Rightarrow P} \Leftrightarrow (P \wedge \neg Q)$

IV.4. Contrapposée d'une implication

▸ Définition 4.2

La contraposée de l'implication $P \Rightarrow Q$ est l'implication $\neg Q \Rightarrow \neg P$

▸ Remarque 4.1

- Il ne pas confondre contrapposée et réciproque !
- L'équivalence $(P \Leftrightarrow Q)$ est la conjonction des deux implications $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$