Benjamin COLLAS
benjamin.collas@telaxo.com

Mi-101 *Mathematica* 2004-2005

# Leçon 1 : Introduction à la logique

**Objectifs :**
- manipuler les propositions booléennes
- établir une table de vérité afin de définir un de base ou caratériser une classe de formules équivalentes
- transformer une formule en formules équivalentes
- utiliser les raisonnements par contraposé et par l'absurde

Des compléments seront faits concernant les formules propositionnelles comportant des quantificateurs.

## Connecteurs propositionnels booléens

### Les connecteurs propositionnels élémentaires

Les connecteurs propositionnels *Mathematica* de base sont les suivants

```
And  Not
Or
```

Ils sont définis par leur table de vérité. Celles-ci ont été créées afin de modéliser au mieux les raisonnements mathématiques.

| | |
|---|---|
| True | False |
| False | False |

**And**

| | |
|---|---|
| True | True |
| True | False |

**Or**

| | |
|---|---|
| False | True |

**Not**

### Exercices :

En utilisant l'aide, trouvez le fonctionnement de ces fonctions ainsi que quelques exemples.
Donnez une définition d'une fonction booléenne.



### Tables de vérités

Afin de caractériser un connecteur booléen, il suffit de connaitre l'ensemble de ses valeurs pour chacun des couples pris dans l'espace de départ (*{True, False}×{True, False}* pour une proposition à deux arguments).
Les résultats se disposent usuellement sous forme de tableau, appelé *table de vérité* de la formule propositionnelle.

### La commande Outer

Nous commencerons par l'étude de la commande *Outer* dans le cas de deux arguments.

**Exercices :**

Utilisez l'aide pour trouver la syntaxe de la commande. Repérez bien la nature des objets mis en oeuvre.
A l'aide de *liste1*, *liste2* et des trois commandes suivantes, expérimentez la commande *Outer*.

```
liste1 = {1, 2, 3, 4};
liste2 = {a, b, c};
liste3 = {x, y, z, t, u};

Outer[Plus, liste1, liste2]
Outer[Times, liste1, liste2]
Outer[List, liste1, liste2]
Outer[List, liste1, liste2, liste3]
```

**Exercice :**

A l'aide de *Outer* et de la liste *ref* suivante, retrouvez les tables de vérité des connecteurs propositionnels *And* et *Or*.

```
ref = {True, False};
```

**Exercice :**

Après avoir lu les tables de vérité précédentes, que pouvez dire de la phrase suivante "jeune homme sérieux cherche location ou/et collocation pas chère".
Quelle erreur est ici commise ? Le "ou" utilisé ici est-il en concordance avec le "ou" mathématique ?
Etudiez la fonction booléenne *Xor*. Commentaires ?

**Exercice :**

Ecrire la table de vérité des formules propositionnelles suivantes *non(non A)*, *(non B) ou (non A)*.

## Les formules propositionnelles

En composant les différents connecteurs propositionnels de base avec des variables propositionnelles, on obtient une formule booléenne plus complexe que nous nommerons *formule propositionnelle*.
On peut alors les transformer en formules propositionnelles équivalentes en utilisant les règles de transformations usuelles ( ou encore chercher l'expression de leur négation).

## ■ Caractérisation

La caractérisation d'une formule propositionnelle se fait grâce à sa table de vérité

### ■ Exercice

Trouver une formule équivalente pour *Non (A et B).*
*Indication : on pourra s'appuyer sur les commandes suivantes*

```
formule1[a_, b_] := Or[Not[a], Not[b]]
Outer[formule1, ref, ref] // TableForm
```

```
formule2[a_, b_] := Not[And[a, b]]
Outer[formule2, ref, ref] // TableForm
```

### ■ Exercice

Trouver une formule équivalente pour *Non (A ou B).*

## ■ Connecteur propositionnel implication

Afin de représenter au mieux les raisonnements mathématiques, on a chercher à ajouter un nouveau connecteur propositionnel à la liste des connecteurs de base. Il s'agit du connecteur *Implication*

```
? Implies
```

### ■ Exercice :

Retrouvez et lisez la table de vérite de la fonction booléenne *Implies* qui la définit.
Commentez la maxime "du faux on peut déduire n'importe quoi".

### ■ Exercice :

Retrouvez et lisez la table de vérité de *formule3* ci-dessous.
En déduire la décomposition en connecteurs propositionnels élémentaires de $A \Rightarrow B$.
Cela concorde-t-il avec la dénomination de *connecteur propositionnel de base* ?

```
formule3[a_, b_] := Or[Not[a], b]
```

### ■ Exercice

Ecrivez la négation de $A \Rightarrow B$.



## ■ Trois arguments et plus

Vous trouverez ci-dessous une fonction permettant d'obtenir la table de vérité d'une formule propositionnelle comportant plus de deux variables (celle-ci n'est pas à retenir).

### ■ Formule A : A ou (B et C)

```
formuleA[{a_, b_, c_}] := {a, b, c, Or[a, And[b, c]]}
refABC = Flatten[Outer[List, ref, ref, ref], 2];
tableformuleA = Thread[formuleA[refABC]];
DisplayForm[GridBox[tableformuleA, RowLines → 1, ColumnLines → 1]]
```

### ■ Formule B : (A ou B) et C

```
formuleB[{a_, b_, c_}] := {a, b, c, And[Or[a, b], c]}
tableformuleB = Thread[formuleB[refABC]];
DisplayForm[GridBox[tableformuleB, RowLines → 1, ColumnLines → 1]]
```

**Exercice :**

A-t-on équivalence entre **A ou (B et C)** et **(A ou B) et C** ?

**Exercice :**

Quelles modifications effectuer pour passer de 3 à 4 arguments ? Pour étudier une formule comportant des variables propositionnelles différentes ?

## ■ Formules équivalentes et règles de tansformations

Les deux exercices suivants vous permettent d'utiliser la construction de formules propositionnelles équivalentes par les règles de transformations usuelles.
Vous pouvez vérifier votre raisonnement en affichant les tables de vérité dess formules obtenues.

**Exercice :**

Trouvez les formules propositionnelles équivalentes parmi les suivantes *A⇒(B⇒C), (A⇒C) et (B⇒C), (A et B)⇒C, (A ou B)⇒C, (A et B)⇒C, (A⇒C) ou (B⇒C).*

**Exercice :**

Ecrivez la négation des formules suivantes (on pourra vérifier ses résultats à l'aide des outils précèdents) :
*A⇒non B, non B⇒non A, (non A ou B)⇒C, (A et B)⇒(C et D).*

## Deux raisonnements types

### ■ Contraposition

Nous considérons ici un énoncé du type $A \Rightarrow B$ que nous cherchons à prouver. Une des possibilités consiste à prouver une forme logiquement équivalente de cet énoncé.

```
formule4[a_, b_] := Implies[Not[b], Not[a]]
```

#### ■ Exercices :

Démontrez l'équivalence des deux formules suivantes $A \Rightarrow B$ et non(*B) ⇒ non(A) :*

- en utilisant les tables de vérités
- en utilisant l'équivalence de formules

Vérifiez que la proposition obtenue est bien logiquement équivalente à l'implication de départ.
Rédigez la démonstration de la proposition suivante :
*Soit P un polynôme.*
*P'(1)=0 implique il n'y a pas (x-1)^2 en facteur dans P*

### ■ Absurde

Démontrer un énoncé *B* par l'absurde consiste à démontrer le faux par exemple *And[B,Not[B]]* à partir de la négation de *B*.

#### ■ Exercice :

Sur un exemple, reprenez la démonstration par l'absurde de "Il existe une infinité de nombres premiers". Utilisez les fonctions *Mathematica* pour vérifier chacune des étapes.
*Indication : utilisez la leçon 0.*

#### ■ Exercice :

Sur un exemple, reprenez la démonstration par l'absurde de "$\sqrt{2}$ est irrationnel". Utilisez les fonctions *Mathematica* pour vérifier chacune des étapes.
*Indication : utilisez la leçon 0.*