Les booléens

Janvier 2013

1 Contenu du document et travail

1. Connaissances à acquérir : opérations booléennes : présentation des opérations booléennes et, ou, non, ou-exclusif. Exprimer des opérations logiques par combinaison d'opérateurs de base.
2. L'ensemble des définitions et résultats de ce fichier sont à apprendre. Tous les exercices doivent être maîtrisés.
3. Quatre exercices sont à rendre (titre sur fond jaune) dans le casier numérique de vos enseignants avant le 20/01.

2 Rappel sur le type bool en python

le type bool de python

Les variables python de type booléen peuvent prendre deux valeurs : False, True.

Exemple de session ipython :

```
In [1]: a=True
In [2]: type(a)
Out[2]: bool
In [3]: b=(2==3)
In [4]: b
Out[4]: False
In [5]: type(b)
Out[5]: bool
In [6]: c=(5<=10)
In [7]: c
Out[7]: True
In [8]: type(c)
Out[8]: bool
```

3 Définition des opérations booléennes de base et, ou, non (and, or, not).

Point Histoire

George Boole (1815-1864) est un logicien, mathématicien, philosophe britannique. Il est le créateur de la logique moderne, fondée sur une structure algébrique et sémantique, que l'on appelle aujourd'hui algèbre de Boole.

Notons $\mathbb{B}$ l'ensemble $\{0;1\}$ que nous appellerons ici ensemble des booléens.
Nous allons dans ce chapitre manipuler quelques fonctions booléennes, c'est à dire des fonctions d'une ou plusieurs variables dans $\mathbb{B}$ et à valeurs dans $\mathbb{B}$.
Nous commençons par définir les fonctions de base et, ou, non. Le nom de ces fonctions vient de la convention d'associer 0 à "faux" et 1 à "vrai".

3.1 La fonction non (not)

3.1.1 Définition

Définition

La fonction non est une fonction définie sur $\mathbb{B}$ et à valeurs dans $\mathbb{B}$ par non(0)=1 et non(1)=0.

On représentera souvent la définition d'une fonction booléenne par la table de ses valeurs :

x	non(x)
0	1
1	0

La fonction non transforme donc 0 en 1 et 1 en 0 ou encore faux en vrai et vrai en faux.

3.1.2 La fonction NON en langage Python

Session ipython :

```
In [1]: for p in (True,False):
   ...:     print p, not(p)
   ...:
True False
False True
```

La seconde session ci-dessous explicite les liens sous-jacents entre 0, False et 1, True.

```
In [2]: for p in (0,1):
   ...:     print p, not(p)
   ...:
0 True
1 False
```

3.1.3 Définition arithmétique

Exercice avec corrigé 1

0 et 1 étant des entiers, définir la fonction non à l'aide des opérations usuelles sur les entiers.

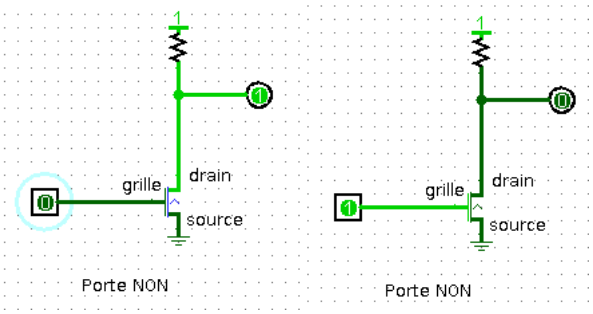
Une résolution

Copie d'une session ipython :

```
In [1]: def non(a):
   ...:     return 1-a
   ...:

In [2]: for p in (0,1):
   ...:     print p, non(p)
   ...:
0 1
1 0
```

3.1.4 Un inverseur en machine

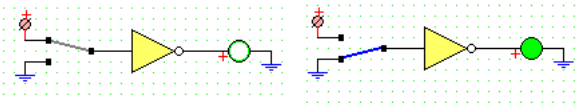
On rappelle avec le schéma ci-dessous (copie d'écran du logiciel libre logisim permettant la simulation http://ozark.hendrix.edu/~burch/logisim/, ce logiciel est déjà installé sur votre clef) comment définir un inverseur (porte NON) avec un transistor.



Un transistor est un circuit électronique à trois fils : le drain, la source et la grille.
La résistance entre le drain et la source est très petite ou très grande en fonction de la tension appliquée entre la grille et la source.
Quand cette tension est inférieure à un certain seuil, la résistance est très grande : le transistor est bloqué (le courant ne passe pas). Quand la tension est supérieure au seuil, la résistance est petite : le transistor est passant.
On peut retenir de façon simple que le comportement est celui d'un interrupteur : l'interrupteur est ouvert lorsque la tension appliquée entre la grille et la source est petite, il est fermé lorsque cette tension est grande. cf schémas ci-dessus.

3.1.5 La porte NON

La porte logique non (inverseur) : 
Fonctionnement d'une porte NON avec le logiciel digsim at http://digsim.free.fr/.



Et avec le logiciel logisim :

3.2 La fonction et (and)

3.2.1 Définition

Définition

La fonction et est une fonction définie sur $\mathbb{B}^2$ et à valeurs dans $\mathbb{B}$.
On peut résumer sa table de valeurs par l'équivalence :
(et$(x, y) = 1$) si et seulement si (x et y valent tous deux 1).

Exercice avec corrigé 2
Compléter la table de valeurs de la fonction et :

x	y	et(x,y)
0	0	
0	1	
1	0	
1	1	

remarque : on notera souvent "x et y" au lieu de "et(x,y)".

Une résolution
La table complète :

x	y	et(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

□

3.2.2 ET en langage python

Session ipython :

```
In [1]: for p in (True,False):
   ...:     for q in (True,False):
   ...:         print p, q, p and q
   ...:
True True True
True False False
False True False
False False False
```

ou encore :

```
In [2]: for p in (0,1):
   ...:     for q in (0,1):
   ...:         print p, q, p and q
   ...:
0 0 0
0 1 0
1 0 0
1 1 1
```

3.2.3 Définition arithmétique

Exercice avec corrigé 3

0 et 1 étant des entiers, définir la fonction et à l'aide des opérations usuelles sur les entiers.

Une résolution

Copie d'une session ipython :

```
In [1]: def et(x,y):
   ...:     return x*y
   ...:

In [2]: for a in (0,1):
   ...:     for b in (0,1):
   ...:         print a, b, et(a,b)
   ...:
0 0 0
0 1 0
1 0 0
1 1 1
```

Autre possibilité :

```
In [1]: def et(x,y):
   ...:     return min(x,y)
   ...:

In [2]: for a in (0,1):
```

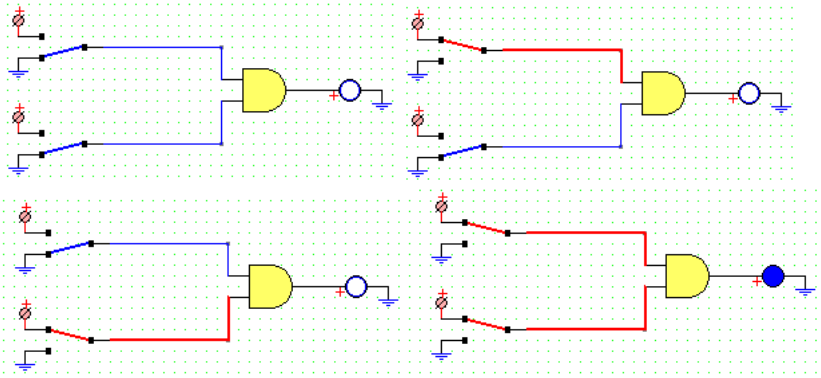
```
    ...:     for b in (0,1):
    ...:         print a,b,et(a,b)
    ...:
0 0 0
0 1 0
1 0 0
1 1 1
```

ou en explicitant un peu les affichages :

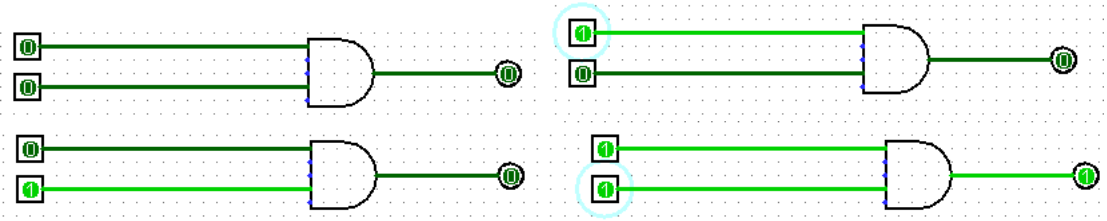
```
 In [1]: def et(x,y):
    ...:     return min(x,y)
    ...:

In [2]: for a in (0,1):
    ...:     for b in (0,1):
    ...:         print "et("+str(a)+","+str(b)+")="+str(et(a,b))
    ...:
et(0,0)=0
et(0,1)=0
et(1,0)=0
et(1,1)=1
```

3.2.4 Porte logique ET

Fonctionnement d'une porte ET avec le logiciel digsim :



Et avec le logiciel logisim :



3.3 La fonction ou (or)

3.3.1 Définition

Définition

La fonction ou est une fonction définie sur $\mathbb{B}^2$ et à valeurs dans $\mathbb{B}$.
On peut résumer sa table de valeurs par l'équivalence :
(ou$(x, y) = 1$) si et seulement si (au moins l'une des deux variables x, y est égale à 1).

On rappelle que le "ou" utilisé en logique est un ou inclusif.

Exercice avec corrigé 4
Compléter la table de valeurs de la fonction ou :

x	y	ou(x,y)
0	0	
0	1	
1	0	
1	1	

remarque : on notera souvent "x ou y" au lieu de "ou(x,y)".

Une résolution
La table complète :

x	y	ou(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

□

3.3.2 OU en langage python.

Session ipython :

```
In [1]: for p in (True, False):
   ...:     for q in (True, False):
   ...:         print p, q, p or q
   ...:
True True True
True False True
False True True
False False False
```

3.3.3 Définition arithmétique

Exercice avec corrigé 5
0 et 1 étant des entiers, définir la fonction ou à l'aide des opérations usuelles sur les entiers.

Une résolution
Copie d'une session ipython :

```
In [1]: def ou(x,y):
   ...:     return x+y-x*y
   ...:

```

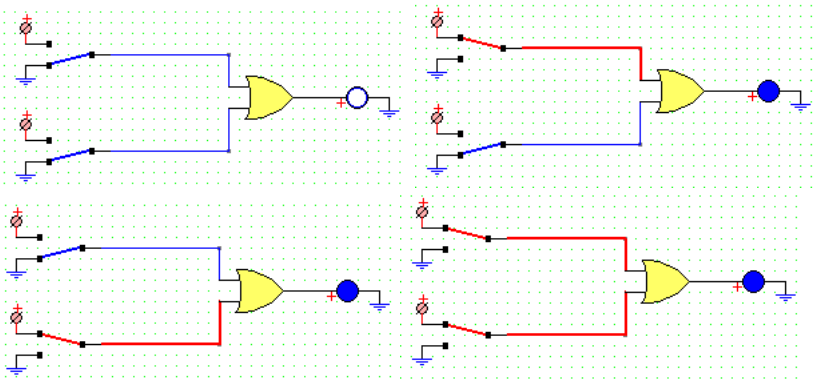
```
In [2]: for a in (0,1):
    ...:     for b in (0,1):
    ...:         print a, b, ou(a,b)
    ...:
0 0 0
0 1 1
1 0 1
1 1 1
```

Autre possibilité :

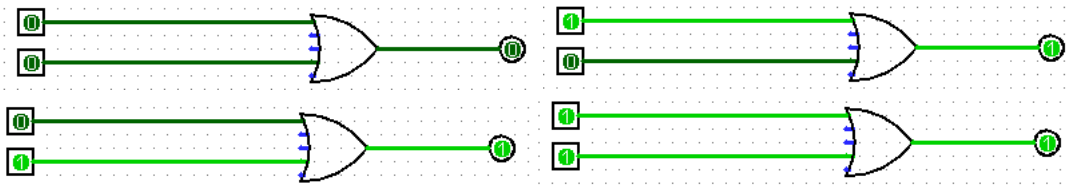
```
In [1]: def ou(x,y):
    ...:     return max(x,y)
    ...:

In [2]: for a in (0,1):
    ...:     for b in (0,1):
    ...:         print a,b, ou(a,b)
    ...:
0 0 0
0 1 1
1 0 1
1 1 1
```

3.3.4 Porte logique OU

Fonctionnement d'une porte OU avec le logiciel digsim :



Et avec le logiciel logisim :



3.4 Montage d'interrupteurs

Exercice avec corrigé 6 (interrupteurs)
On peut monter deux interrupteurs en parallèle, en série ou en va-et-vient. Faire le lien entre ces trois montages et les fonctions et, ou, non.

Une résolution
En parallèle : ou. En série : et. En va-et-vient : xor. □

4 Le ou exclusif oux (xor)

4.1 Définition

Définition

La fonction oux (le x signifie “exclusif”) est une fonction définie sur $\mathbb{B}^2$ et à valeurs dans $\mathbb{B}$.
On peut résumer sa table de valeurs par l'équivalence :
$(\text{oux}(x,y)=1)$ si et seulement si (exactement une des deux variables x,y est égale à 1).
Ou encore : $(\text{oux}(x,y)=1)$ si et seulement si $(x \neq y)$.

Exercice avec corrigé 7
Compléter la table de valeurs de la fonction oux :

x	y	oux(x,y)
0	0	
0	1	
1	0	
1	1	

remarque : on notera souvent “x oux y” au lieu de “oux(x,y)”.

Une résolution
La table complète :

x	y	oux(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

□

4.2 XOR en langage python.

Copie d'une session ipython :

```
In [1]: for p in (True, False):
   ...:     for q in (True, False):
   ...:         print p, q, p^q
   ...:
True True False
True False True
False True True
False False False
```

Ou encore :

```
In [2]: for p in (True, False):
   ...:     for q in (True, False):
   ...:         print p, q, p!=q
   ...:
True True False
True False True
False True True
False False False
```

4.3 Définition arithmétique de xor

Exercice avec corrigé 8

xor étant la fonction définie par :

x	y	xor(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

donner une expression algébrique possible de cette fonction.

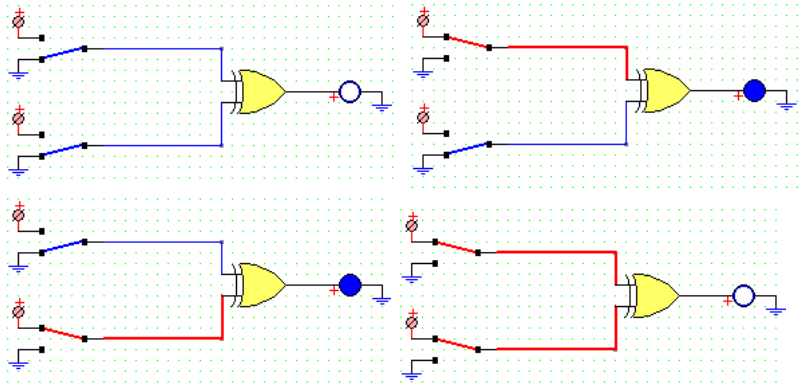
Une résolution

Copie de session ipython :

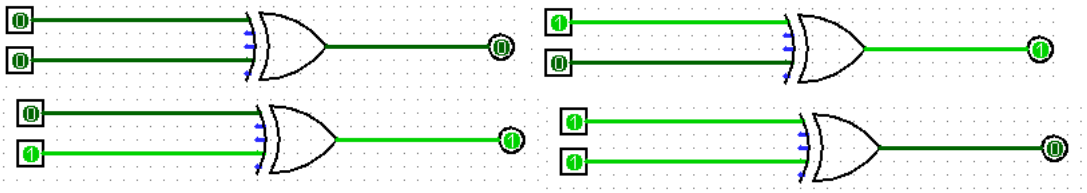
```
In [1]: def xor(a,b):
   ...:     return abs(a-b)
   ...:

In [2]: for p in (0,1):
   ...:     for q in (0,1):
   ...:         print p, q, xor(p,q)
   ...:
0 0 0
0 1 1
1 0 1
1 1 0
```

4.4 Porte XOR

Fonctionnement d'une porte XOR avec le logiciel digsim :



Et avec le logiciel logisim :



5 Les fonctions booléennes à une variable.

Les fonctions booléennes définies sur $\mathbb{B}$ et à valeurs dans $\mathbb{B}$ sont au nombre de 4.

1. La fonction NON :

x	non(x)
0	1
1	0

2. La fonction identité :

x	Id(x)
0	0
1	1

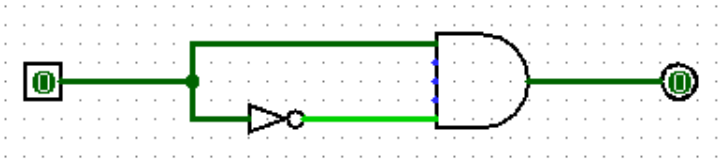
3. Les deux fonctions constantes :

x	Z(x)
0	0
1	0

x	U(x)
0	1
1	1

Exercice à rendre 1

A quelle fonction de $\mathbb{B}$ dans $\mathbb{B}$ correspond le montage ci-dessous obtenu avec logisim ?



Exercice avec corrigé 9

1. La fonction Z constante égale à 0 est une fonction définie sur $\mathbb{B}$ et à valeurs dans $\mathbb{B}$, par $Z(0) = 0$ et $Z(1) = 0$.

x	Z(x)
0	0
1	0

Exprimer la fonction Z à l'aide des fonctions booléennes et, ou, non.

2. La fonction U constante égale à 1 est une fonction définie sur $\mathbb{B}$ et à valeurs dans $\mathbb{B}$, par $U(0) = 1$ et $U(1) = 1$.

x	U(x)
0	1
1	1

Exprimer la fonction U à l'aide des fonctions booléennes et, ou, non.

Une résolution

1. Z(x)= et(x, non(x)).

x	non(x)	x et non(x)
0	1	0
1	0	0

2. U(x)= ou(x, non(x)).

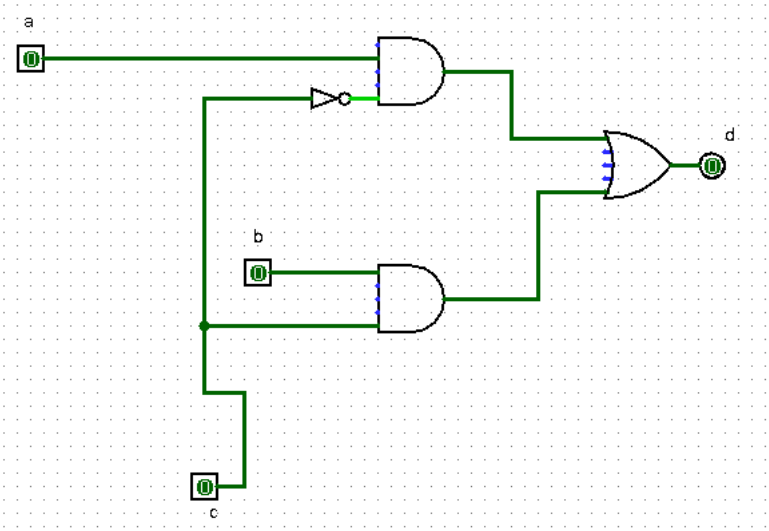
x	non(x)	x ou non(x)
0	1	1
1	0	1

□

6 La fonction multiplexeur mux

Exercice avec corrigé 10

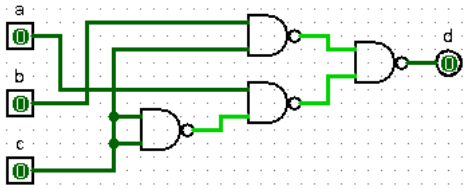
1. Avec le logiciel logisim, on a défini le circuit logique ci-dessous :



(a) Quand c est à 0, quelle est la sortie d ?

(b) Quand c est à 1, quelle est la sortie d ?

2. Même question avec le circuit ci-dessous :



Une résolution

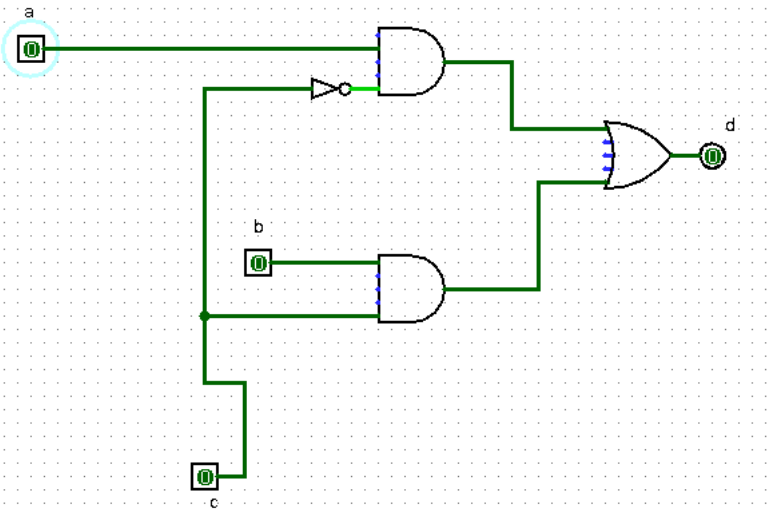
Quand c=0, d=a. Quand c=1, d=b.
On appelle mux la fonction ainsi définie :

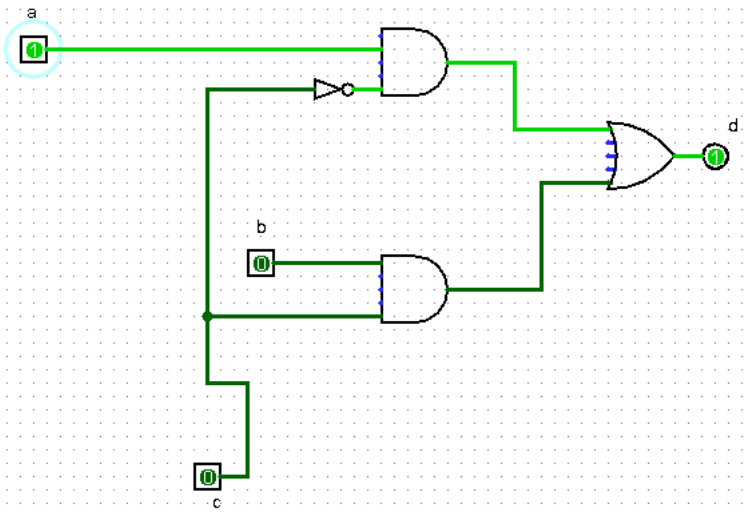
$$\text{mux}(c,a,b) = \begin{cases} a \text{ pour } c = 0 \\ b \text{ pour } c = 1 \end{cases}$$

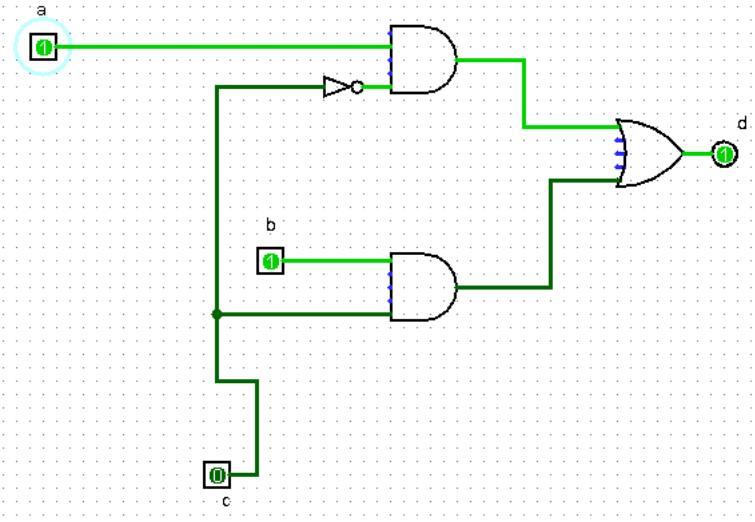
c'est à dire, pour toute valeur de a et toute valeur de b :

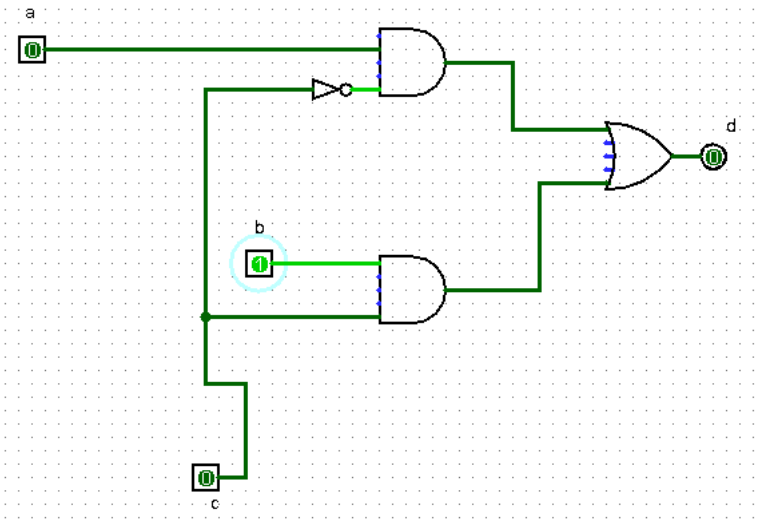
$$\text{mux}(0,a,b) = a \qquad \text{mux}(1,a,b) = b$$

Le second circuit est équivalent au premier.
Illustration pour $c = 0$:


a
b
c
d


a
b
c
d


a
b
c
d



□

fonction multiplexeur

La fonction mux est une fonction de $\mathbb{B}^3$ dans $\mathbb{B}$.
On la définit pour tout $y \in \mathbb{B}$ et tout $z \in \mathbb{B}$ par :

$$\begin{cases} \text{mux}(0, y, z) = y \\ \text{mux}(1, y, z) = z \end{cases}.$$

Exercice avec corrigé 11

Compléter la table de la fonction mux.

x	y	z	mux(x,y,z)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Une résolution

Table complète :

x	y	z	mux(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

□

Expression de mux à l'aide des fonctions de base

Pour tous booléens x, y, z : $\text{mux}(x,y,z) = (\text{non}(x) \text{ et } y) \text{ ou } (x \text{ et } z)$

Exercice avec corrigé 12

Démontrer le résultat précédent "à la main".

Une résolution

Il suffit de vérifier que les tables de valeurs sont identiques.

x	y	z	non(x) et y	x et z	(non(x) et y) ou (x et z)	mux(x,y,z)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	1	1

□

Exercice avec corrigé 13

Démontrer le résultat précédent à l'aide d'un programme python.

Une résolution

Programme python `mux/mux.py` vérifiant l'égalité des deux tables de valeurs :

Python

```python
# -*- coding: utf-8 -*-

def m(x,y,z):
    """ m est une fonction de B^3 dans B"""
    return (not(x) and y)or(x and z)

def mux(x,y,z):
    """ fonction mux """
    if x==False: return y
    if x==True :return z

def testEgalite(f,g):
    """teste l'égalité de deux fonctions de B^3 dans B """
    for x in (True,False):
        for y in (True,False):
            for z in (True, False):
                if f(x,y,z)!=g(x,y,z):
                    return "fonctions distinctes"
    return "fonctions égales"

print testEgalite(mux,m)
```

□

Exercice avec corrigé 14 (expression de la fonction oux à l'aide des fonctions de base)

1. Vérifier que pour tout $x \in \mathbb{B}$ et tout $y \in \mathbb{B}$, on a :

$$\text{oux}(x,y) = \text{mux}(y, \text{oux}(x,0), \text{oux}(x,1))$$

2. Exprimer la fonction définie sur $\mathbb{B}$ par $x \mapsto \text{oux}(x,0)$ à l'aide des fonctions booléennes de base.
3. Exprimer la fonction définie sur $\mathbb{B}$ par $x \mapsto \text{oux}(x,1)$ à l'aide des fonctions booléennes de base.
4. En déduire une expression de la fonction oux à l'aide des fonctions booléennes de base.

Une résolution

1. Pour tout x :
   - si $y = 0$: $\text{mux}(y, \text{oux}(x,0), \text{oux}(x,1)) = \text{oux}(x,0) = \text{oux}(x,y)$
   - si $y = 1$: $\text{mux}(y, \text{oux}(x,0), \text{oux}(x,1)) = \text{oux}(x,1) = \text{oux}(x,y)$.

   Les deux fonctions sont donc bien égales.
2. Table des valeurs :

x	oux(x,0)
0	0
1	1

   Il s'agit de la fonction identité Id.
3. Table des valeurs :

x	oux(x,1)
0	1
1	0

   Il s'agit de la fonction not.
4. On a : $\text{oux}(x,y) = \text{mux}(y, \text{oux}(x,0), \text{oux}(x,1)) = \text{mux}(y, x, \text{non}(x))$ On utilise ensuite l'expression de la fonction mux : $\text{oux}(x,y) = (\text{non}(y) \text{ et } x) \text{ ou } (y \text{ et non}(x))$. □

A savoir

Pour tout $x \in \mathbb{B}$ et tout $y \in \mathbb{B}$: x xor $y = (x \text{ et non}(y)) \text{ ou } (\text{non}(x) \text{ et } y)$.

7 La fonction "si et seulement si " (que l'on notera ↔).

La fonction ↔ est une fonction définie sur $\mathbb{B}^2$ et à valeurs dans $\mathbb{B}$ par :

$$\forall x \in \mathbb{B}, \forall y \in \mathbb{B} : \qquad \leftrightarrow (x,y) = 1 \text{ si et seulement si } (x = y)$$

Exercice à rendre 2 (expression de ↔ par les fonctions de base)

1. Compléter la table des valeurs de la fonction ↔.

x	y	↔(x,y)
0	0	
0	1	
1	0	
1	1	

2. Vérifier que pour tout $x \in \mathbb{B}$ et tout $y \in \mathbb{B}$, on a :

$$\leftrightarrow (x,y) = \text{mux}(y, \leftrightarrow (x,0), \leftrightarrow (x,1))$$

3. Exprimer la fonction définie sur $\mathbb{B}$ par $x \mapsto \leftrightarrow (x,0)$ à l'aide des fonctions booléennes de base.
4. Exprimer la fonction définie sur $\mathbb{B}$ par $x \mapsto \leftrightarrow (x,1)$ à l'aide des fonctions booléennes de base.
5. En déduire une expression de la fonction ↔ à l'aide des fonctions booléennes de base.

8 La fonction →

On définit la fonction → sur $\mathbb{B}^2$ par la table suivante :

x	y	→(x,y)
0	0	1
0	1	1
1	0	0
1	1	1

Exercice avec corrigé 15

1. En suivant le modèle donné pour la fonction oux et la fonction ↔, exprimer la fonction → à l'aide des fonctions de base.
2. Vérifier que pour tous x et y de $\mathbb{B}$, on a : $\rightarrow(x,y) = \text{non}(x)$ ou y.

Une résolution

1. Pour tout $x \in \mathbb{B}$ et tout $y \in \mathbb{B}$, on a : $\rightarrow(x,y) = \text{mux}(y, \rightarrow(x,0), \rightarrow(x,1))$

x	→(x,0)
0	1
1	0

   Il s'agit de la fonction not.

x	→(x,1)
0	1
1	1

   Il s'agit de la fonction constante égale à 1.

   On a : $\rightarrow(x,y) = \text{mux}(y, \rightarrow(x,0), \rightarrow(x,1)) = \text{mux}(y, \text{non}(x), \mathbf{1})$ On utilise ensuite l'expression de la fonction mux : $\rightarrow(x,y) = (\text{non}(y) \text{ et } \text{non}(x)) \text{ ou } (y \text{ et } \mathbf{1})$.
2. On vérifie que les tables sont les mêmes. □

Savoir Faire

Exprimer une fonction booléenne de deux variables à l'aide des fonctions booléennes de base.

Exercice avec corrigé 16

Vérifier que pour tous x et y dans $\mathbb{B}$, on a : $\leftrightarrow(x,y) = \rightarrow(x,y)$ et $\rightarrow(y,x)$.

Une résolution

Il suffit de vérifier que les tables de valeurs sont les mêmes.

x	y	x→y	y→x	(x→y) et (y→x)
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

□

9 Une fonction pour les exprimer toutes.

Exercice avec corrigé 17

1. Combien de fonctions booléennes différentes à une variable peut-on définir ?

2. Combien de fonctions booléennes différentes à deux variables peut-on définir ? (on en a déjà rencontré 5 : et, ou, oux, $\leftrightarrow$, $\rightarrow$).
3. Combien de fonctions booléennes différentes à trois variables peut-on définir ?
4. Combien de fonctions booléennes différentes à n variables peut-on définir (n entier naturel non nul) ?

Une résolution

1. La fonction constante 0, la fonction constante 1, la fonction identité, la fonction non : soit quatre fonctions de $\mathbb{B}$ dans $\mathbb{B}$.
   Une fonction de $\mathbb{B}$ dans $\mathbb{B}$ peut se représenter ainsi : (image de 0 ; image de 1).
   Comme il y a deux choix pour chaque image, il y a $2^2 = 4$ fonctions de $\mathbb{B}$ dans $\mathbb{B}$.
2. Une fonction de $\mathbb{B}^2$ dans $\mathbb{B}$ peut se représenter ainsi :
   (image de (0,0) ; image de (0,1) ; image de (1,0) ; image de (1,1)).
   Comme chaque image a deux valeurs possibles, il y a $2^4 = 16$ fonctions de $\mathbb{B}^2$ dans $\mathbb{B}$.
3. Pour une fonction de $\mathbb{B}^3$ dans $\mathbb{B}$, il y a $2^3 = 8$ variables.
   Comme chaque image a deux valeurs possibles, le nombre de fonctions de $\mathbb{B}^3$ dans $\mathbb{B}$ est 2^8.
4. Pour chaque image, il y a deux valeurs possibles. Le nombre de fonctions de $\mathbb{B}^n$ dans $\mathbb{B}$ est $2^{(2^n)}$. □

Exercice avec corrigé 18

L'exercice a pour objectif de vous faire manipuler la décomposition des fonctions booléennes définies sur $\mathbb{B}^2$ à l'aide de la fonction mux par le biais de l'étude d'un programme : `generation/genere.py`.

1. On considère la fonction python suivante :

   Python

   ```python
   def bi(i):
       """ i entier entre 0 et 15"""
       b=[0,0,0,0]
       b[3]=i%2
       i=i//2
       b[2]=i%2
       i=i//2
       b[1]=i%2
       i=i//2
       b[0]=i%2
       return b
   ```

   Pour i entier entre 0 et 15, que retourne bi(i) ?
2. A l'aide de la fonction bi précédente, on définit la fonction suivante :

   Python

   ```python
   def f(i):
       def g(x,y):
           if (x,y)==(0,0):return bi(i)[0]
           if (x,y)==(0,1):return bi(i)[1]
           if (x,y)==(1,0):return bi(i)[2]
           if (x,y)==(1,1):return bi(i)[3]
       return g
   ```

   Vérifier que f(7) est la fonction ou.
3. On définit le tableau ci-dessous :

Python

```
T=[]
for i in range(0,16):
    T.append(f(i))
```

Que peut-on dire des éléments de T ?

4. Dans les paragraphes qui précèdent, nous avons vérifié pour quelques fonctions f de $\mathbb{B}^2$ dans $\mathbb{B}$ l'égalité :

$$\forall x \in \mathbb{B},\ \forall y \in \mathbb{B},\quad f(x,y) = \text{mux}\big(y, f(x,0), f(x,1)\big)$$

c'est à dire

$$\forall x \in \mathbb{B},\ \forall y \in \mathbb{B},\quad f(x,y) = (\text{non}(y) \text{ et } f(x,0)) \text{ ou } (y \text{ et } f(x,1))$$

Il est clair que le raisonnement fait est générique et s'applique à toute fonction f de $\mathbb{B}^2$ dans $\mathbb{B}$.
On demande ici d'utiliser les fonctions python précédentes pour vérifier à l'aide d'un programme python que cette égalité est vraie pour toute fonction f de $\mathbb{B}^2$ dans $\mathbb{B}$.

Une résolution

1. Écriture binaire de l'entier, chiffres dans une liste.
2. On peut le vérifier en dressant la table :

Python

```
for x in (0,1):
    for y in (0,1):
        print x,y,f(7)(x,y)
```

3. Le tableau contient les 16 fonctions de $\mathbb{B}^2$ dans $\mathbb{B}$.
4. On définit la fonction mux :

Python

```
def mux(x,y,z):
    """ fonction mux """
    if x==0: return y
    if x==1 :return z
```

et la fonction $(x,y) \mapsto \text{mux}\big(y, f(x,0), f(x,1)\big)$ associée à f :

Python

```
def muf(f):
    def g(x,y):
        return mux(y,f(x,0),f(x,1))
    return g
```

puis une fonction testant l'égalité de deux fonctions de $\mathbb{B}^2$ dans $\mathbb{B}$:

Python

```
def testEgalite(f,g):
    """teste l'égalité de deux fonctions de B^2 dans B"""
    for x in (0,1):
        for y in (0,1):
            if f(x,y)!=g(x,y):return False
    return True
```

Il reste à tester :

Python

```
for i in range(0,16):
    print testEgalite(T[i],muf(T[i]))
```

cf fichier `generation/genere.py`. □

Exercice avec corrigé 19

Nous avons vu comment exprimer une fonction booléenne définie sur $\mathbb{B}^2$ à l'aide des fonctions booléennes de base et, ou, non.
Peut-on faire de même avec des fonctions booléennes de 3, 4, 5 ... variables ? La réponse est oui.
Ce résultat se prouve par récurrence : il suffit de réitérer le procédé utilisé pour deux variables.
Écrire cette preuve.

Une résolution

1. Les fonctions booléennes d'une variable ont déjà toutes été rencontrées. On peut les considérer comme étant des fonctions de base (elles peuvent être exprimées à l'aide de et, ou, non).
2. Nous avons exprimé quelques fonctions booléennes de deux variables à l'aide des fonctions booléennes de base et le procédé utilisé est générique, il peut être appliqué à toute fonction booléenne de deux variables. Étudier notamment l'exercice précédent.
3. Soit $n \geqslant 2$ un entier. Supposons (hypothèse de récurrence) que nous pouvons exprimer toute fonction booléenne de n variables à l'aide des seules fonctions de base.
   Considérons maintenant une fonction booléenne f de $n+1$ variables (c'est à dire une fonction définie sur $\mathbb{B}^{n+1}$ et à valeurs dans $\mathbb{B}$).
   On remarque, comme déjà traité en exercice, que pour tout $(x_1, x_2, \ldots, x_n, x_{n+1}) \in \mathbb{B}^{n+1}$, on a :
   $f(x_1, x_2, \ldots, x_n, x_{n+1}) = \text{mux}\big(x_{n+1}, f(x_1, x_2, \ldots, x_n, 0), f(x_1, x_2, \ldots, x_n, 1)\big)$, c'est à dire :
   $f(x_1, x_2, \ldots, x_n, x_{n+1}) = (\text{non}(x_{n+1}) \text{ et } f(x_1, x_2, \ldots, x_n, 0)) \text{ ou } (x_{n+1} \text{ et } f(x_1, x_2, \ldots, x_n, 1))$ $\quad (*)$.
   Les fonctions $(x_1, x_2, \ldots, x_n) \mapsto f(x_1, x_2, \ldots, x_n, 0)$ et $(x_1, x_2, \ldots, x_n) \mapsto f(x_1, x_2, \ldots, x_n, 1)$ sont des fonctions booléennes de n variables. On sait donc, par hypothèse de récurrence, les exprimer à l'aide des fonctions de base : on sait donc finalement exprimer f à l'aide des fonctions booléennes de base en remplaçant dans $(*)$. □

A savoir

Toute fonction de $\mathbb{B}^n$ dans $\mathbb{B}$ peut s'exprimer en utilisant uniquement les fonctions et, ou, non.

Point Histoire

Auguste De Morgan (1806 - 1871) est un mathématicien et logicien britannique. Il est le fondateur, avec G. Boole, de la logique moderne.

Exercice avec corrigé 20 (lois de De morgan)

Démontrer les lois de De Morgan :

1. Pour tout $(x, y) \in \mathbb{B}^2$, on a : non(x) et non(y) = non(x ou y).
2. Pour tout $(x, y) \in \mathbb{B}^2$, on a : non(x) ou non(y) = non(x et y).

Une résolution

1. On compare les tables.

x	y	not(x)	not(y)	et(not(x),not(y))
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

x	y	ou(x,y)	not(ou(x,y))
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

2. Analogue. □

Exercice avec corrigé 21 (réduire la base)

1. Démontrer que toute fonction booléenne peut encore s'exprimer à l'aide des fonctions "ou" et "non".
2. Démontrer que toute fonction booléenne peut encore s'exprimer à l'aide des fonctions "et" et "non".

Une résolution

1. Pour tout $(x,y) \in \mathbb{B}^2$, on a : non(x) ou non(y) = non$(x$ et $y)$. On en déduit que pour tout $(x,y) \in \mathbb{B}^2$, on a : non$\big($non(x) ou non$(y)\big)$ = x et y. La fonction et s'exprime donc à l'aide des autres fonctions de base et on peut donc s'en passer.
2. Pour tout $(x,y) \in \mathbb{B}^2$, on a : non(x) et non(y) = non$(x$ ou $y)$. On en déduit que pour tout $(x,y) \in \mathbb{B}^2$, on a : non$\big($non(x) et non$(y)\big)$ = x ou y. La fonction ou s'exprime donc à l'aide des autres fonctions de base et on peut donc s'en passer. □

A savoir

Toute fonction booléenne peut s'exprimer à partir des seules fonctions ET, NON.
Toute fonction booléenne peut s'exprimer à partir des seules fonctions OU, NON.

Point Histoire

Henry Maurice Sheffer (1882-1964) est un logicien américain. Sheffer a prouvé en 1913 que l'algèbre booléenne peut être définie à l'aide d'un seul opérateur binaire, NAND.

Exercice à rendre 3 (base de Sheffer)

On définit la fonction de Sheffer sur $\mathbb{B}^2$ par S(x,y) = non$(x$ et $y)$ pour tout $(x,y) \in \mathbb{B}^2$. Cette fonction est souvent nommée NAND.

1. Dresser la table de cette fonction.
2. Exprimer la fonction non à l'aide de la fonction S.
3. Exprimer la fonction ou à l'aide de la fonction S.
4. En déduire l'affirmation du point-histoire ci-dessus, c'est à dire que toute fonction booléenne de $\mathbb{B}^n$ dans $\mathbb{B}$ peut s'exprimer en utilisant la seule fonction nand.

A savoir

Toute fonction booléenne peut être exprimée à l'aide de la seule fonction NAND (non-et). En particulier, tout circuit logique pourrait être constitué de seules portes NAND.

10 Quelques énigmes de logique

Exercice avec corrigé 22

Un roi cruel avait pour habitude de proposer le marché suivant à ses condamnés :

Tout prisonnier doit choisir entre deux cellules. Chaque cellule peut contenir une corde ou une clef (un seul objet par cellule). Si le prisonnier choisit une cellule avec une corde, il sera pendu. S'il choisit une cellule avec clef, il sera libéré. Pour rendre plus cruelle encore la situation, les deux cellules contiennent parfois toutes deux une corde, mais aussi parfois toutes deux une clef et parfois l'une contient une clef et l'autre une corde.
Le roi donne par ailleurs des indications. Voici les indications données à l'un des prisonniers :
– sur la porte de la cellule 1 est inscrit : « il y a une clef dans cette cellule et une corde dans l'autre cellule. »
– sur la porte de la cellule 2 est inscrit : « il y a une clef dans l'une des cellules et une corde dans l'autre cellule. »
Le roi annonce par ailleurs au prisonnier qu'une seule des deux inscriptions est correcte.

1. On note L1 :« la cellule contient une clef », L2 : « la cellule 2 contient une clef ». Traduire les affirmations A1 et A2 inscrites sur les portes à l'aide des propositions L1, L2 et des connecteurs logiques (et, ou, non).
2. Écrire un programme en langage python qui affiche la table des valeurs de vérité de A1 et A2 suivant les valeurs de vérité de L1 et L2.
3. Conclure sur le choix que doit faire le prisonnier.
4. Avec le logiciel digsim ou avec le logiciel logisim, simuler un circuit résolvant le problème.

Une résolution

1. A1 = L1 et non(L2) ; A2= (L1 et non(L2)) ou (non(L1) et L2) ou encore A2= (L1 xor L2).
2. Un programme python possible :

Python

```
for L1 in (True,False):
    for L2 in (True,False):
        A1=(L1 and not(L2))
        A2=(L1 and not(L2)) or (not(L1) and L2)
        print L1,L2,A1,A2
```

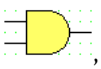
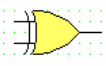
3. On obtient avec le programme précédent :

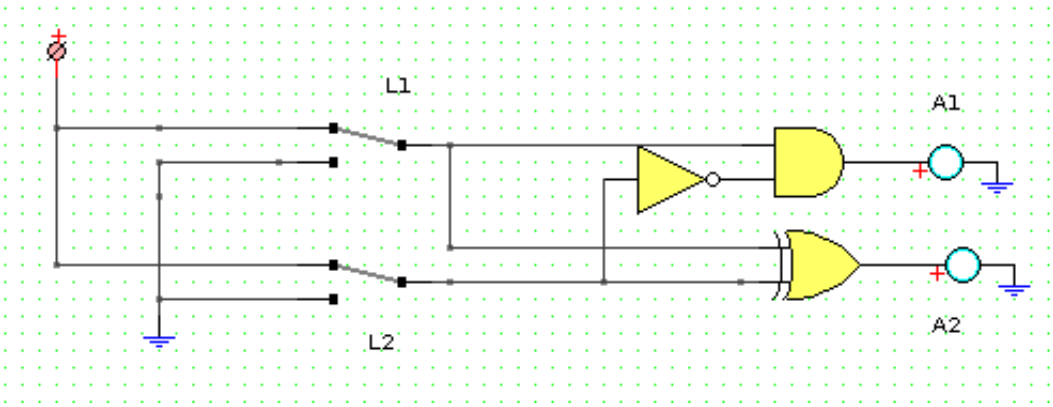
   True True False False
   True False True True
   False True False True
   False False False False
   Seule la ligne 3 correspond au cas d'une seule affirmation juste. La liberté est donc donnée par la porte 2.
   On peut intégrer l'affirmation finale du roi ("une seule des deux inscriptions est vraie") en n'affichant le couple (L_1, L_2) que lorsque la phrase "une seule des deux inscriptions est vraie" est vraie, c'est à dire lorsque l'on a : (A1 and not(A2)) or (not(A1) and A2)= vrai, c'est à dire (A1 xor A2)==True.

Python

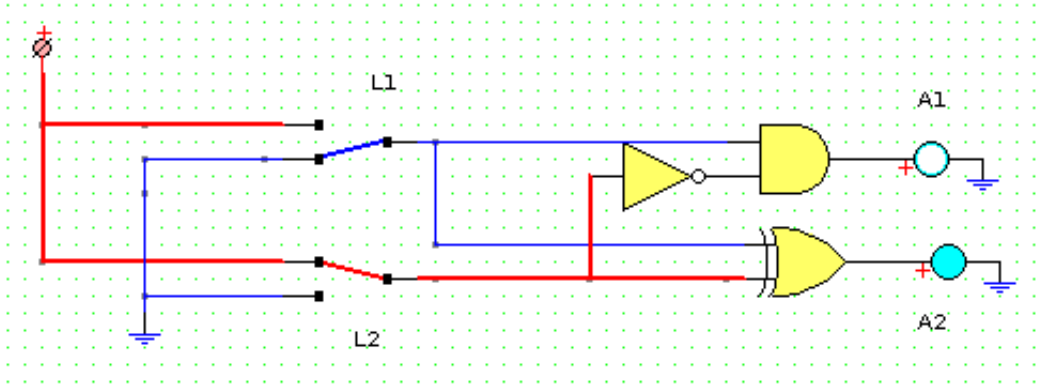
```
for L1 in (True,False):
    for L2 in (True,False):
        A1=(L1 and not(L2))
        A2=(L1 and not(L2))or(not(L1) and L2)
        if (A1 and not(A2)) or (not(A1) and A2):
            print L1,L2
```

   On obtient dans ce cas le seul affichage False, True : porte 1 marquée d'un 0 (en d'autres termes : L_1 est fausse, la cellule 1 ne contient donc pas de clef) et porte 2 marquée d'un 1 (en d'autres termes : L_2 est vraie, la cellule 2 contient une clef).
4. Avec le logiciel digsim (fichier prison/fichiers_digsim/prisonnier).

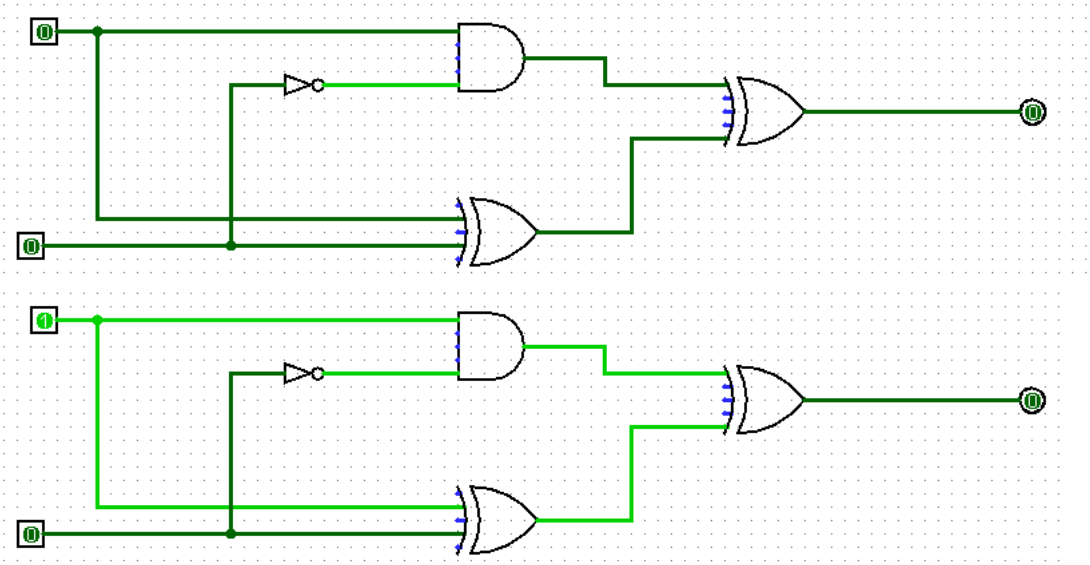
   On utilise une porte non  , une porte et  , et une porte oux  .
   Schéma (simulation non activée) :

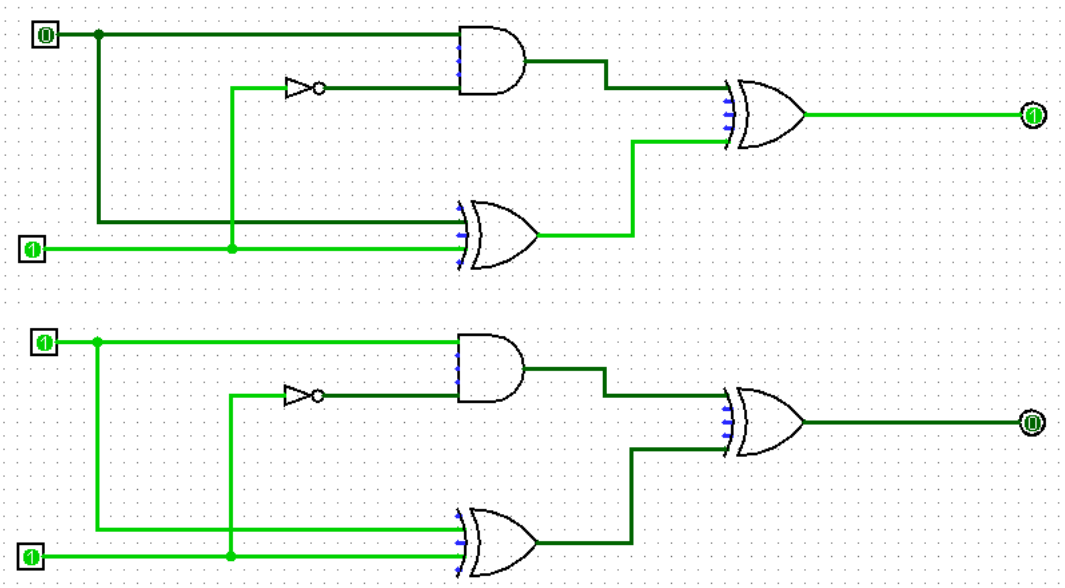


Seule la position False– True des deux “contacts” donne une seule lampe allumée :



Avec logisim, en intégrant une porte xor pour tenir compte de l’affirmation finale du roi :





Le seul montage donnant 1 en sortie (ce 1 correspond à la vérification que l'affirmation finale du roi est satisfaite) correspond à porte 1 marquée d'un 0 (en d'autres termes : L_1 est fausse, la cellule 1 ne contient donc pas de clef) et porte 2 marquée d'un 1 (en d'autres termes : L_2 est vraie, la cellule 2 contient une clef). □

Exercice avec corrigé 23

Même exercice que le précédent avec les indications suivantes :
- Sur la porte de la cellule 1, on trouve l'affirmation A1 : « Au moins l'une des deux cellules contient une clef ».
- Sur la porte de la cellule 2, on trouve l'affirmation A2 : « L'autre cellule contient une corde ».
- Le roi précise que les affirmations sont vraies toutes deux ou fausses toutes deux.

Une résolution

Programme python :

Python

```
for L1 in (True,False):
    for L2 in (True,False):
        A1=(L1 or L2)
        A2=not(L1)
        print L1,L2,A1,A2
```

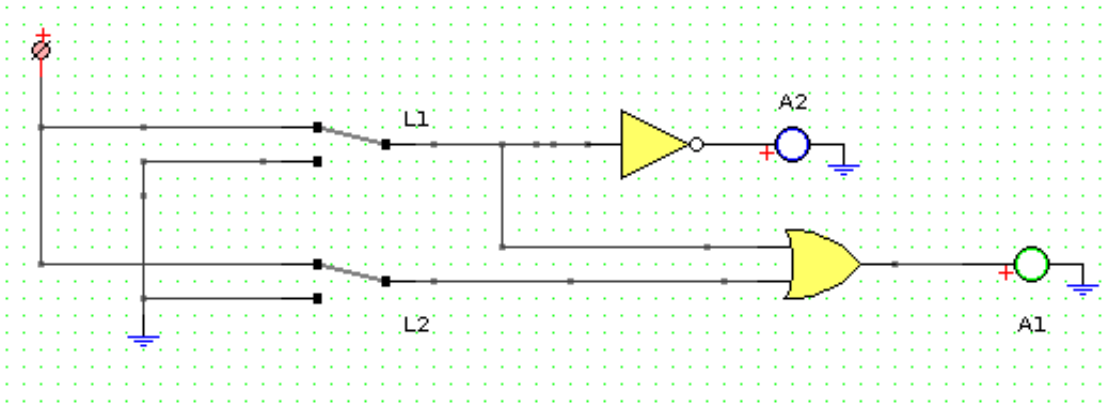
On obtient :
True True True False
True False True False
False True True True
False False False True

Le seul cas pour lequel les deux affirmations ont même valeur de vérité est le cas 3. Il faut choisir la cellule 2.
On peut aussi intégrer l'affirmation du roi dans le programme :

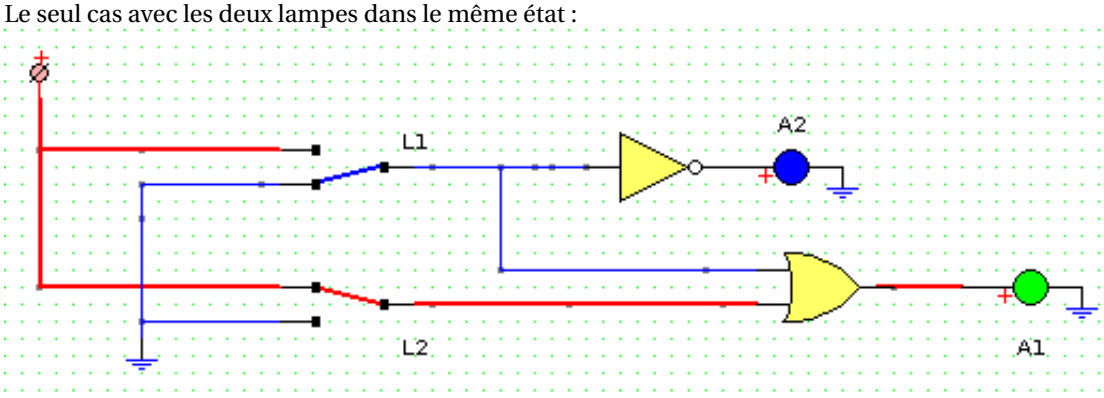
Python

```python
for L1 in (True,False):
    for L2 in (True,False):
        A1=(L1 or L2)
        A2=not(L1)
        if (A1 and A2) or (not(A1) and not(A2)):
            print L1,L2
```

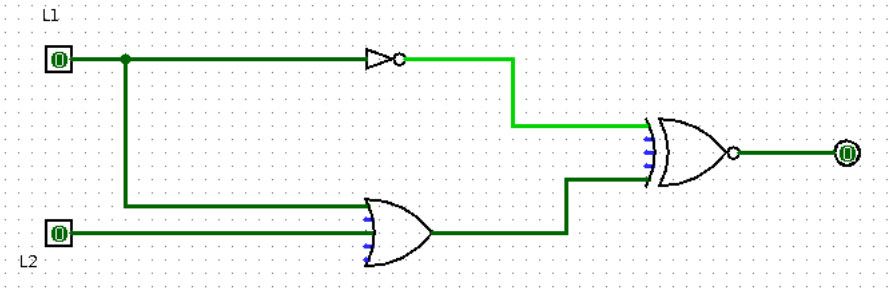
Avec digsim (fichier prison/fichiers_digsim/prisonnier2).
Schéma (simulation non activée) :

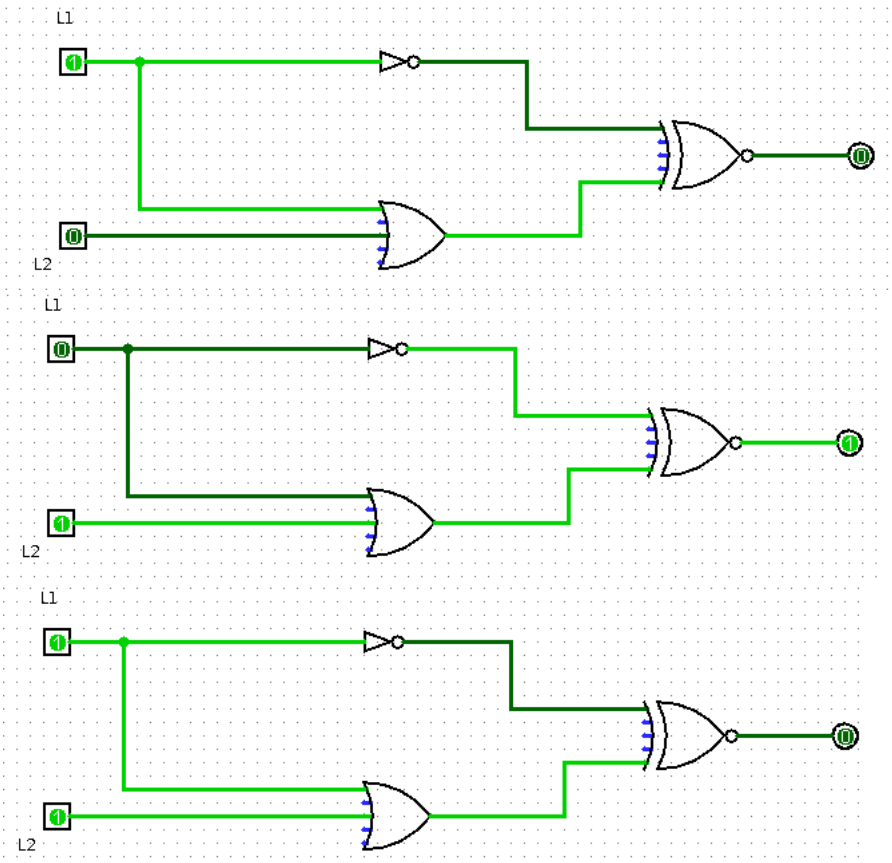


Le seul cas avec les deux lampes dans le même état :



Avec logisim, en intégrant une porte xnor (non-oux) correspondant à la déclaration du roi :





□

Exercice avec corrigé 24

Sur une planète vivent les Purs (qui disent toujours la vérité) et les Pires (qui mentent toujours). Vous croisez deux personnes sur cette planète A et B. A affirme : « Au moins l'un de nous deux est un pire ».
Notons a :« A est pur », b : « B est pur ».
A l'aide d'une table de vérité, dire ce que sont A et B.

a	b	...
true	true	
true	false	
false	true	
false	false	

Une résolution

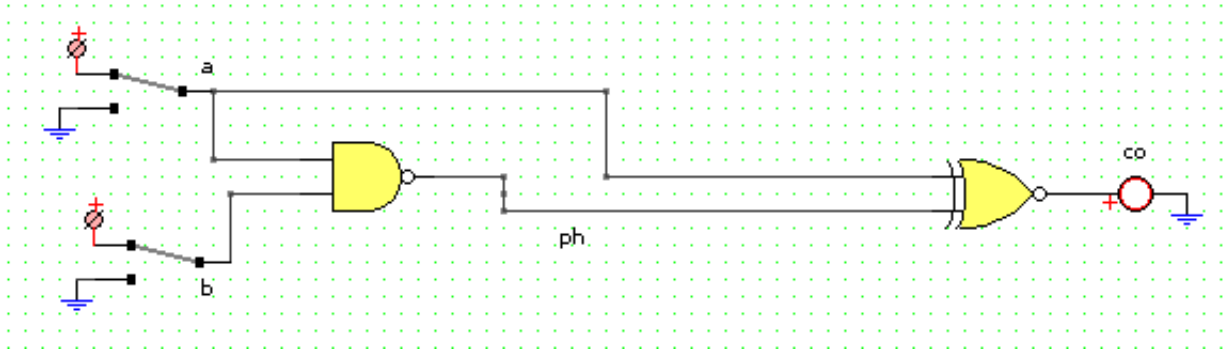
On tient compte de la phrase de A : ph= not(a) ou not(b). Et de la cohérence entre la valeur de vérité de cette phrase et la "race" de A : (a et ph) ou (not(a) et not(ph))=1.
Programme python :

Python

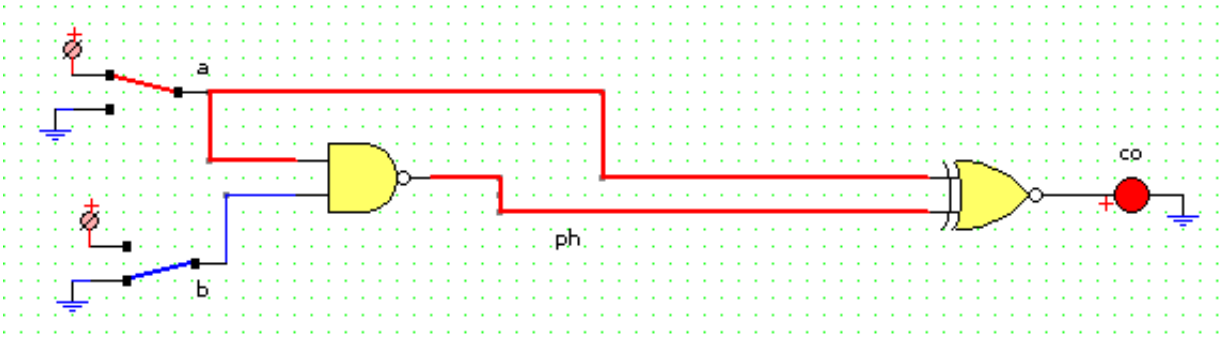
```
for a in (True,False):
    for b in (True,False):
        ph=(not(a) or not(b))
        co=(a and ph)or(not(a)and not(ph))
        if co:
            print a,b
```

On obtient un seul affichage : True, False.

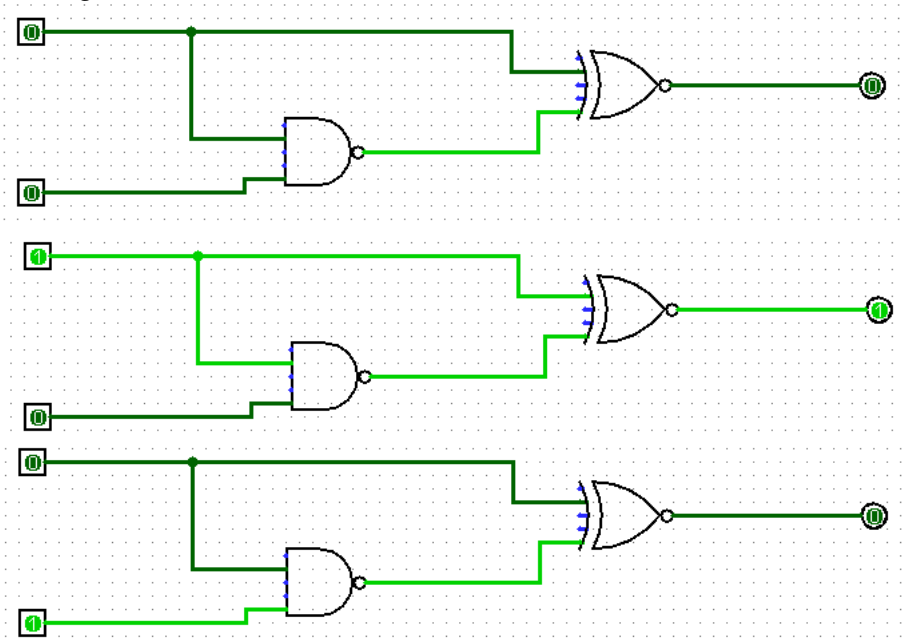
A est pur, B est pire.
Raisonnement direct : Si A était pire, il ne dirait pas la phrase annoncée (puisqu'elle constituerait une vérité). Donc il est pur…
Avec digsim (fichier prison/fichiers_digsim/purpire).
Schéma (non activé) :

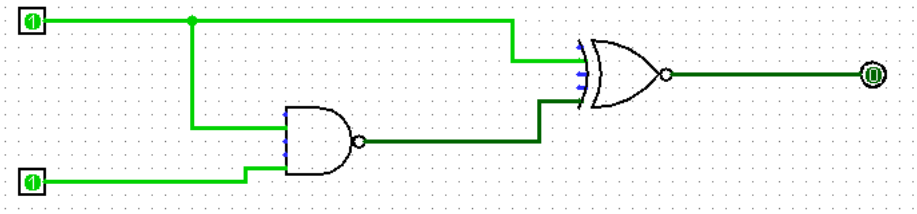


Le seul cas allumant la lampe :



Avec logisim :





Exercice à rendre 4

Sur la planète NP vivent les Naïfs (qui disent toujours la vérité) et les Politiciens (qui mentent toujours). Vous croisez deux personnes sur cette planète A et B. A affirme : « Je suis un Politicien ou B est un Naïf ». Que sont A et B ?

1. Répondre par un raisonnement.
2. Répondre à l'aide d'un programme python.
3. Répondre à l'aide d'un circuit logisim.

Exercice avec corrigé 25

1. Sur la planète NP, un habitant déclare : " Je suis un Politicien ou 2+2=4". Quelle est la nature de A ?
2. Sur la planète NP, un habitant déclare : " Je suis un Naïf ou 2+2=4". Quelle est la nature de A ?

Une résolution

2+2=4 étant vraie, la phrase "je suis un Politicien ou 2+2=4" est vraie. A est Naïf. Idem pour le second cas. □

Exercice avec corrigé 26

J'ai quitté la planète NP il y a déjà fort longtemps. Je n'arrive plus à me souvenir si la scène suivante a effectivement eu lieu ou si ma mémoire me joue des tours : « je croise un habitant A de la planète et il m'annonce : " Je suis un Politicien ou 2+2=5". ». Qu'en pensez-vous ?

Une résolution

On note p="A est un Naïf", q="2+2=5". ph="non(p) ou q".
Cohérence entre la nature de A et la valeur de vérité de son affirmation : (p et ph) ou (non(p) et non(ph))=1. La table :

p	q	ph	(p et ph) ou (non(p) et non(ph))
0	0	1	0
1	0	0	0

Quelle que soit la nature de A, le test de cohérence est à faux : il est impossible qu'une personne de cette planète ait prononcée cette phrase (ou alors elle avait des lacunes en logique !) □