Booléens et portes logiques

Mardi 19 Mars

Option Informatique

Ecole Alsacienne

Plan

Correction du petit quizz

1. Introduction
2. Booléens
3. Formules booléennes classiques
4. Propriétés
5. Circuits booléens

Correction du petit quizz

Définitions

Question 1. *Qu'est-ce qu'un booléen ?*

Un booléen est un type de variable qui ne peut prendre que deux valeurs possibles : `vrai` (`true`) ou `faux` (`false`)

Question 2. *Qu'est-ce qu'une fonction récursive ?*

Une fonction récursive est une fonction qui se rappelle elle-même (on parle d'appel récursif).

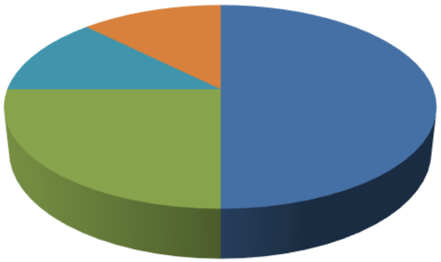
La factorielle, la suite de Fibonacci et de nombreuses fonctions sur les listes sont des fonctions récursives.

DÉFINITIONS

Question 3. *Qu'est-ce qu'une approche "Diviser pour régner" ?*

Une approche "Diviser pour régner" consiste à résoudre un problème en le divisant en plusieurs problèmes analogues mais de plus petite taille, jusqu'à arriver à un cas trivial.

La recherche dichotomique dans un vecteur trié est un exemple classique de ce principe.



DÉFINITIONS

Question 4. *Qu'est-ce qu'un graphe ?*

Un graphe, c'est :

- Un ensemble X de sommets

Ex : $X = \{ x,\ y,\ z \}$

- Un ensemble d'arêtes E constitué de paires d'éléments de X

Ex : $E = \{ (x, y),\ (x, z) \}$

Une carte routière, Internet, un réseau social, un réseau de canalisation sont autant d'exemples de graphes dans nos vies quotidiennes.

VECTEURS ET LISTES

Question 5. *Pour chacune des opérations listées ci-dessous, indiquez si cette opération peut être réalisée en temps constant sur un vecteur et sur une liste.*

	En temps constant sur	
	Un vecteur	Une liste
Ajouter un élément	☐	☑
Compter le nombre d'éléments	☑	☐
Accéder au dernier élément	☑	☐
Supprimer le premier élément	☐	☑
Accéder au premier élément	☑	☑
Trouver le plus grand élément	☐	☐

EXERCICES

Question 6. *Imaginez une fonction aire_triangle :*

- *prenant en argument deux nombres réels b et h (qu'on suppose non nuls)*
- *renvoyant un nombre réel*
- *telle que aire_triangle b h renvoie l'aire d'un triangle de base b et de hauteur h*

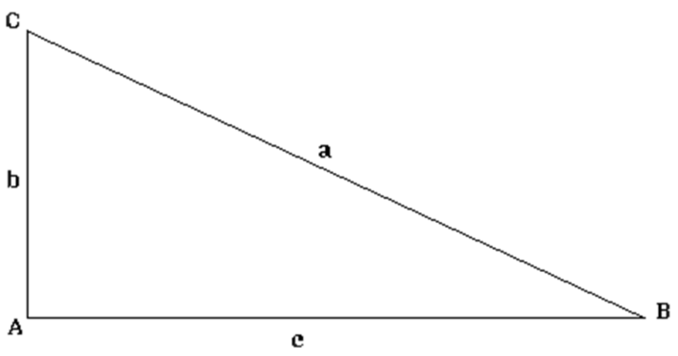
```
let aire_triangle b h =
      (b *. h) /. 2.0;;
```

EXERCICES

Question 7. *On suppose désormais qu'on n'étudie que les triangles rectangles.*

- *Imaginez une fonction aire_triangle_rectangle :*
- *prenant en argument trois nombre réels a, b et c (qu'on suppose non nuls)*
- *renvoyant un nombre réel*
- *telle que aire_triangle_rectangle a b c renvoie l'aire d'un triangle rectangle dont les côtés ont pour longueur a, b et c*

```
let aire_triangle_rectable a b c =
  if (a > b)
  then
    begin
      if (a > c)
      then aire_triangle b c
      else aire_triangle a b
    end
  else
    begin
      if (b > c)
      then aire_triangle a c
      else aire_triangle a b
    end;;
```



EXERCICES

Question 8. *Imaginez une fonction somme_vecteur :*

- *prenant en argument un vecteur d'entiers*
- *renvoyant un nombre entier*
- *telle que somme_vecteur v renvoie la somme des éléments contenus dans le vecteur v*

```
let somme_vecteur v =
  let n = Array.length v in
  let total = ref 0 in
  for i = 0 to (n-1)
  do
    total := !total + v.(i)
  done;
  !total;;
```

EXERCICES

Question 9. *Imaginez une fonction appartient_liste :*

- *prenant en argument un nombre entier x et une liste d'entiers l*
- *renvoyant un booléen*
- *telle que appartient_liste x l renvoie vrai si l'élément x apparaît dans la liste l, et false sinon.*

```
let rec appartient_liste x l =
  if (l = [])
  then false
  else
    begin
      let t = List.hd l in
      let q = List.tl l in
      if (t = x)
      then true
      else appartient_liste x q
    end;;
```

EXERCICES

Question 10. *Imaginez une fonction nombre_elements_distincts :*

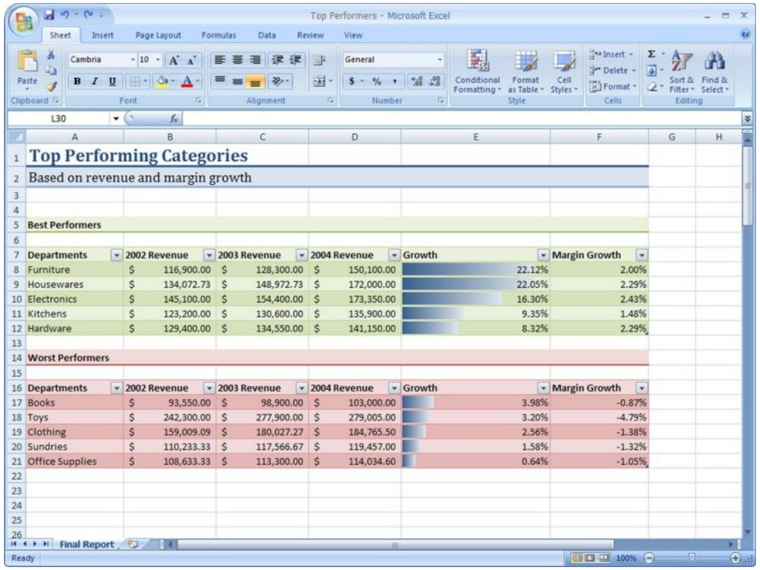
- *prenant en argument une liste d'entiers l*
- *renvoyant un nombre entier*
- *telle que nombre_elements_distincts l renvoie le nombre d'éléments distincts apparaissant dans la liste l.*

```
let rec nombre_elements_distincts l =
  if (l = [])
  then 0
  else
    begin
      let t = List.hd l in
      let q = List.tl l in
      if (appartient_liste t q)
      then nombre_elements_distincts q
      else 1 + (nombre_elements_distincts q)
    end;;
```

Introduction

Introduction

Un ordinateur, ça fait des tas de trucs…



La question : comment ?

INTRODUCTION

Plus simplement, comment un ordinateur sait-il que :

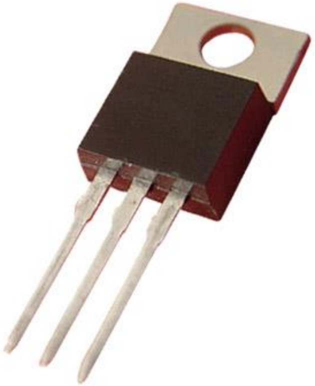
- $40 + 2 = 42$?
- $1 + 1 = 2$?
- $0 \neq 1$?

La réponse en deux mots

0 1

Quelques petites questions

- Qu'est-ce que c'est ?



Un transistor !

- Un exemple concret de porte logique

- Mais au fait, combien y a-t-il de transistors dans un ordinateur ?

Lois de Moore

- **Gordon Earle Moore**, un des trois fondateurs d'Intel

- Première loi de Moore (1965)

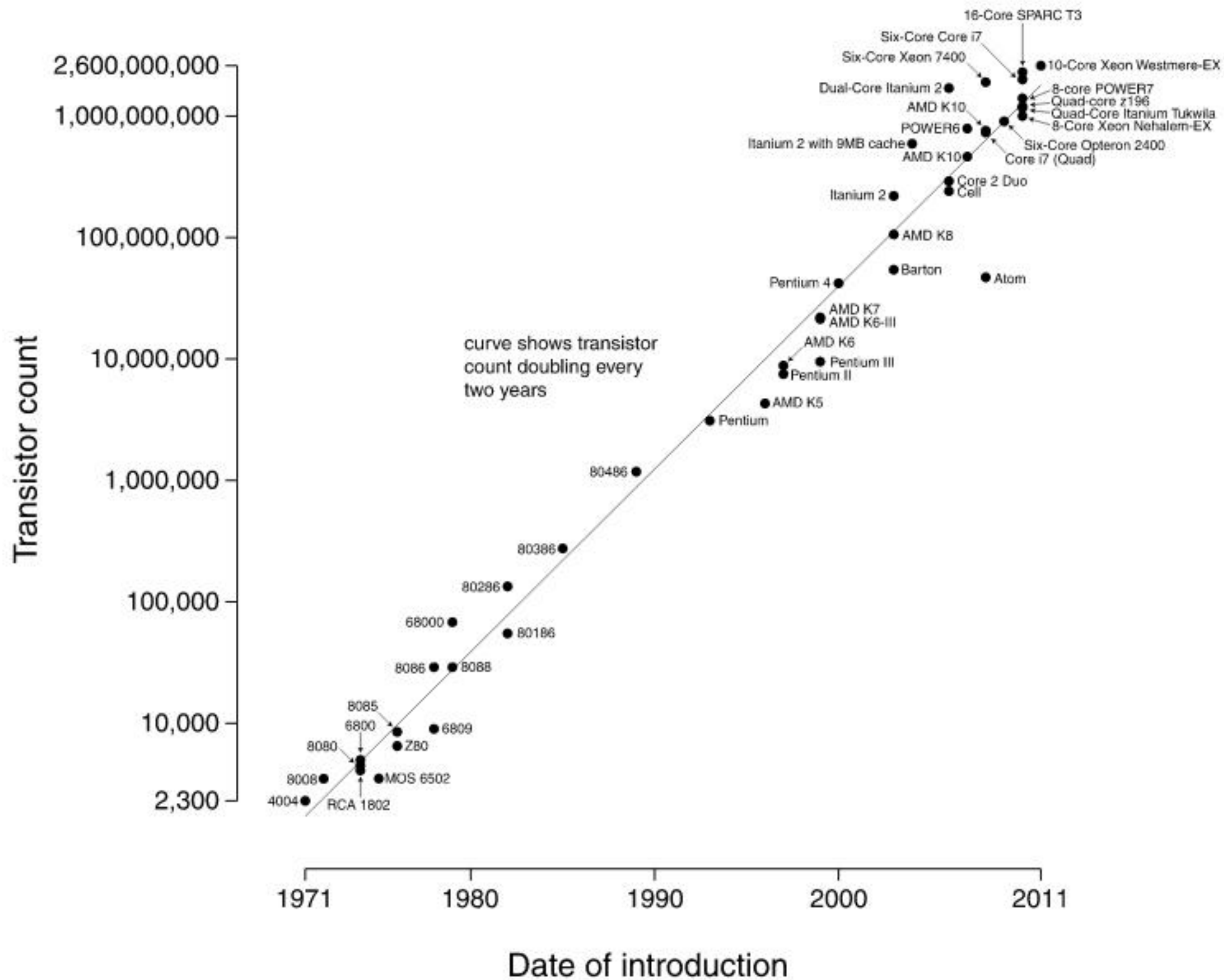
  La complexité des semiconducteurs proposés en entrée de gamme double tous les ans à cout constant

- Seconde loi de Moore (1975)

  Le nombre de transistors des microprocesseurs double tous les deux ans

Lois de Moore

Microprocessor Transistor Counts 1971-2011 & Moore's Law



BOOLÉENS

0 ou 1

0	1
Faux	Vrai
Non	Oui
Ouvert	Fermé
Arrêt	Marche
Nul	Positif
Tension 1	Tension 2

DÉFINITION

- Un **booléen** est une donnée qui ne peut avoir que deux états possibles.

- Ces deux états, aussi appelés **valeurs de vérité**, sont généralement noté `Vrai` et `Faux`, ou encore 1 et 0

- Un booléen peut être
  - Une **constante booléenne** (dont la valeur de vérité est toujours la même)
  - Une **variable booléenne** (dont la valeur de vérité peut changer)

EXEMPLES DE BOOLÉENS

Constantes

- Vrai
- False
- 4 est plus grand que 6

Variables

- Le nombre P est plus grand que le nombre Q
- Il pleut aujourd'hui

FONCTION LOGIQUE

- On appelle **fonction logique** une fonction prenant en argument un (ou plusieurs) booléen(s) et renvoyant un (ou plusieurs) booléen(s).

$$f : \mathbb{B}^n \to \mathbb{B}^m$$

- On parle aussi de **porte logique**, mais plutôt pour désigner la représentation graphique associée ou le composant électronique correspondant.

Table de vérité

- Une **table de vérité** est une liste exhaustive des valeurs d'une fonction logique pour toutes les valeurs possibles de ses arguments

- Une telle table est souvent représentée de la façon suivante :

x	y	$f(x,y)$
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$

EXEMPLE : LA PORTE ET

- La fonction ET réalise la conjonction entre deux booléens :

$$f : \mathbb{B}^2 \to \mathbb{B}$$

$$f(x, y) = \texttt{Vrai} \text{ si et seulement si } \begin{cases} x = \texttt{Vrai} \\ y = \texttt{Vrai} \end{cases}$$

x	y	$f(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1

Fonctions booléennes classiques

Constante `Vrai`

- **Description :** Constante toujours vraie
- **Notation** : `Vrai` ou ⊤ ("*top*")
- **Valeur de vérité** : 1
- **Représentation graphique** :

1

- **Nom anglais** : `True`

CONSTANTE **FAUX**

- **Description :** Constante toujours fausse
- **Notation :** `Faux` ou `False` ou ⊥ ("*bottom*")
- **Valeur de vérité :** 0
- **Représentation graphique :**

0

- **Nom anglais :** `False`

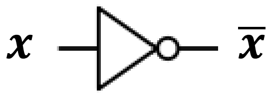
PORTE **NON**

- **Description :** Contraire d'un booléen x

- Notation : $\neg x$ ou $\bar{x}$ ("x barre")

- Table de vérité :

x	$\bar{x}$
0	1
1	0

- Représentation graphique : 

- Nom anglais : `NOT`

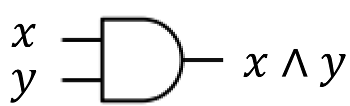
PORTE **ET**

- **Description :** Conjonction de deux booléens x et y

- **Notation :** $x \wedge y$

- **Table de vérité :**

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

- **Représentation graphique :** 

- **Nom anglais :** `AND`

Porte OU

- **Description :** Disjonction de deux booléens x et y

- **Notation :** $x \vee y$

- **Table de vérité :**

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

- **Représentation graphique :**

x
y

$x \vee y$

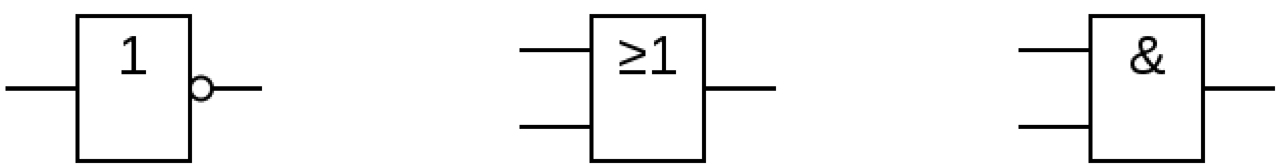
- **Nom anglais :** `OR`

Remarque géographique

- Les symboles présentés dans les slides précédents sont appelés **symboles américains**

- C'est une norme de représentation (nommée ANSI/IEEE 91-1984) adaptée aux schémas simples et aux tracés à la main.

- Il existe d'autres normes, notamment des **symboles** dits **européens** (norme CEI 60617-12), moins utilisés, mais permettant de représenter davantage de circuits :



Propriétés

COMMUTATIVITÉ

- Les fonctions `ET` et `OU` sont commutatives :

$$x \vee y = y \vee x$$

$$x \wedge y = y \wedge x$$

DISTRIBUTIVITÉ

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

x	y	z	$(y \wedge z)$	$x \vee (y \wedge z)$	$(x \vee y)$	$(x \vee \boldsymbol{z})$	$(x \vee y) \wedge (x \vee z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Avec les constantes

- Vrai ∧ Vrai = Vrai
- Vrai ∧ Faux = Faux
- Faux ∧ Vrai = Faux
- Faux ∧ Faux = Faux

- Vrai ∨ Vrai = Vrai
- Vrai ∨ Faux = Vrai
- Faux ∨ Vrai = Vrai
- Faux ∨ Faux = Faux

Avec les constantes

Pour toute variable booléenne x,

- $\mathtt{Vrai} \land x = x$
- $\mathtt{Faux} \land x = \mathtt{Faux}$

- $\mathtt{Vrai} \lor x = \mathtt{Vrai}$
- $\mathtt{Faux} \lor x = x$

Règles de priorité

- **Question** : Que vaut cette formule logique ?

$$\texttt{Vrai} \lor \texttt{Faux} \lor \texttt{Vrai} \land \texttt{Faux}$$

- **Réponse** : `Vrai`

- **Explication** : Le `ET` est prioritaire sur le `OU`.

C'est comme si on avait les parenthèses suivantes :

$$\texttt{Vrai} \lor \texttt{Faux} \lor (\texttt{Vrai} \land \texttt{Faux})$$

- Pour modifier ces priorités, on utilise des parenthèses :

$$(\texttt{Vrai} \lor \texttt{Faux} \lor \texttt{Vrai}) \land \texttt{Faux}$$

Théorème de De Morgan

- Théorème de De Morgan :

$$\overline{x \vee y} = \bar{x} \wedge \bar{y}$$

$$\overline{x \wedge y} = \bar{x} \vee \bar{y}$$

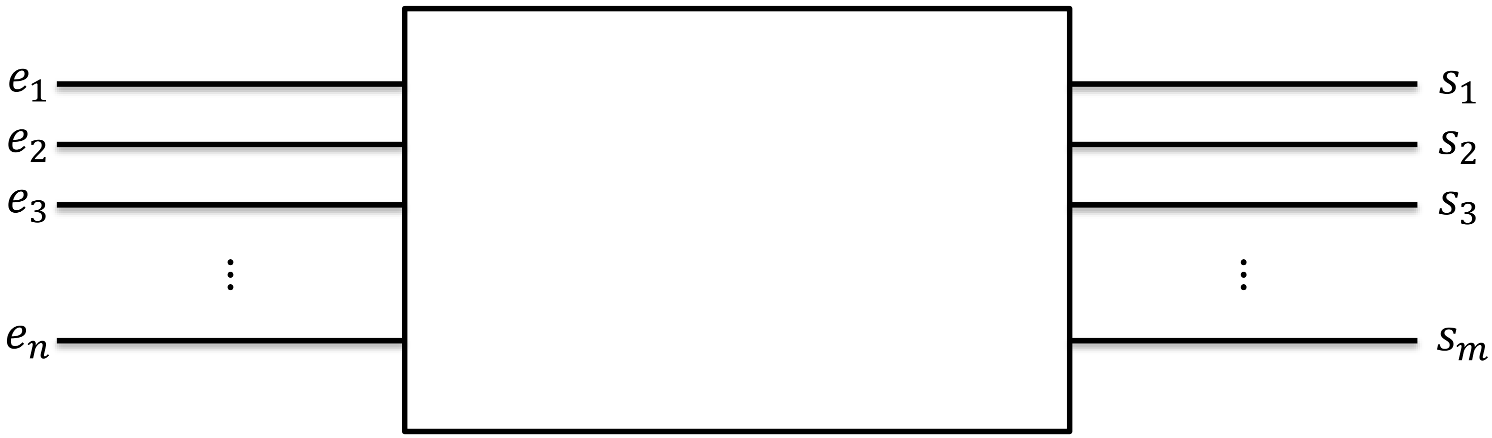
x	y	$x \vee y$	$\overline{x \vee y}$	$\bar{x}$	$\bar{y}$	$\bar{x} \wedge \bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Circuits booléens

DÉFINITIONS

- Un **circuit booléen** est une combinaison de plusieurs portes logiques connectées entre elles.

- Ces circuits reçoivent en entrée un certain nombre de booléens, appelés **variables d'entrée**, et renvoient un autre jeu de booléens appelés **variables de sorties**.



Logique combinatoire

- On travaille dans le cadre de la **logique combinatoire** : les valeurs de vérité des variables de sortie dépendent uniquement des valeurs de vérité des variables d'entrée.

- Il existe d'autres modèles, dont notamment la logique séquentielle, où la sortie dépend aussi des états précédents des entrées

Porte NON-ET

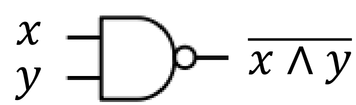
- **Description** : Contraire d'une porte ET

- Notation : $\overline{x \wedge y}$

- Table de vérité :

x	y	$\overline{x \wedge y}$
0	0	1
0	1	1
1	0	1
1	1	0

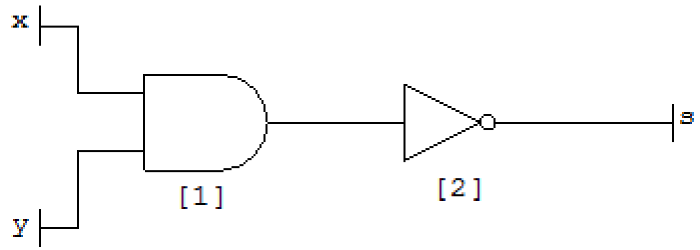
- Représentation graphique :



- Nom anglais : NAND

PORTE NON-ET

Circuit booléen correspondant :



x	y	$\overline{x \wedge y}$
0	0	1
0	1	1
1	0	1
1	1	0

PORTE **NON–OU**

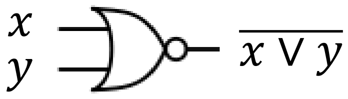
- **Description :** Contraire d'une porte OU

- **Notation :** $\overline{x \vee y}$

- **Table de vérité :**

x	y	$\overline{x \vee y}$
0	0	1
0	1	0
1	0	0
1	1	0

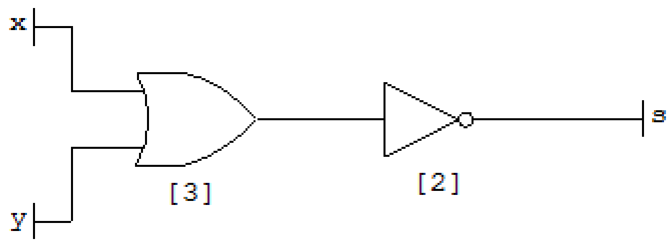
- **Représentation graphique :**



- **Nom anglais :** NOR

PORTE **NON-OU**

Circuit booléen correspondant :



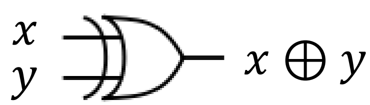
x	y	$\overline{x \vee y}$
0	0	1
0	1	0
1	0	0
1	1	0

PORTE **OU-EXCLUSIF**

- **Description :** "Soit l'un, soit l'autre, mais pas les deux"
- Notation : $x \oplus y$
- Table de vérité :

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

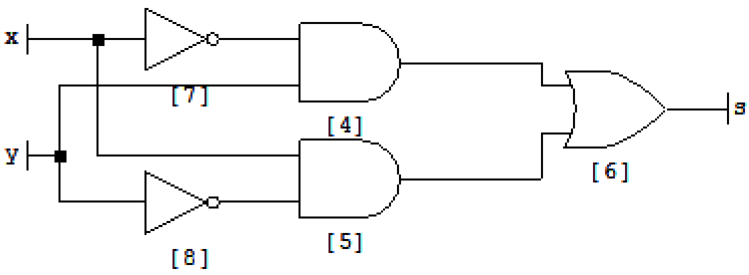
- Représentation graphique :



- Nom anglais : `XOR`

PORTE OU-EXCLUSIF

Circuit booléen correspondant :



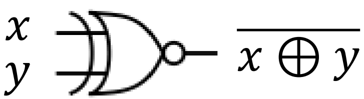
x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Porte NON-OU-EXCLUSIF

- Description : "Soit aucun, soit les deux"
- Notation : $\overline{x \oplus y}$
- Table de vérité :

x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1

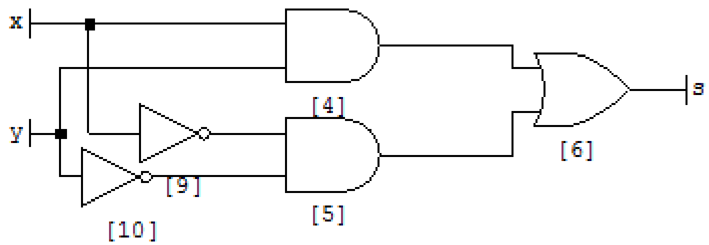
- Représentation graphique :



- Nom anglais : `XNOR`

PORTE NON-OU-EXCLUSIF

Circuit booléen correspondant :



x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1

Prochaine séance

Vendredi 16 mars 2011

[TD] Logic Friday