*fiche de i23 de Mehdi Ben Ahmed*

# Definitions et props

**Définition 1: Commutatif** les variables peuvent etre inverses

**Définition 2: L'arbre de Derivation** C'est un format de pour representer une proposition

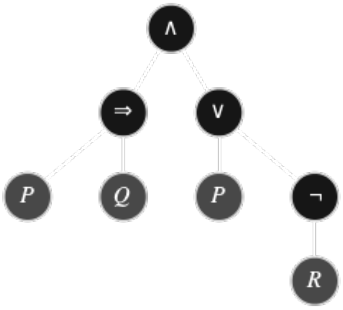


Figure 1: $(P \Rightarrow Q) \wedge (P \vee \neg R)$

**Définition 3: Loi de De Morgan** Soit $P$ et $Q$ deux assertions, alors
$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

# Tables de verite

il est assume qu'un connecteur est commutatif sauf mentione autrement

### table de $\wedge$: q binaire

| | | |
|---|---|---|
| $\bot$ | $\bot$ | $\bot$ |
| $\top$ | $\bot$ | $\bot$ |
| $\bot$ | $\top$ | $\bot$ |
| $\top$ | $\top$ | $\top$ |

### table de $\vee$: q binaire

| | | |
|---|---|---|
| $\bot$ | $\bot$ | $\bot$ |
| $\bot$ | $\top$ | $\top$ |
| $\top$ | $\bot$ | $\top$ |
| $\top$ | $\top$ | $\top$ |

### table de $\oplus$: q binaire

| | | |
|---|---|---|
| $\bot$ | $\bot$ | $\bot$ |
| $\bot$ | $\top$ | $\top$ |
| $\top$ | $\bot$ | $\top$ |
| $\top$ | $\top$ | $\bot$ |

### table de $\Rightarrow$: q binaire dit non commutatif

| | | |
|---|---|---|
| $\bot$ | $\bot$ | $\top$ |
| $\bot$ | $\top$ | $\top$ |
| $\top$ | $\bot$ | $\bot$ |
| $\top$ | $\top$ | $\top$ |

autrement dit, vrai sauf si p est vrai et q est faux

### table de $\Leftrightarrow$: q binaire

| | | |
|---|---|---|
| $\bot$ | $\bot$ | $\top$ |
| $\bot$ | $\top$ | $\bot$ |
| $\top$ | $\bot$ | $\bot$ |
| $\top$ | $\top$ | $\top$ |

vrai si les deux variables ont la meme valeur

### Proprietes

• comutativite de $\wedge$ et $\vee$

$$(p \wedge q) \equiv (q \wedge p)$$

$$(p \vee q) \equiv (q \vee p)$$

• associativite de $\wedge$ et $\vee$

$((P \wedge Q) \wedge R) \equiv ((q \wedge R) \wedge P)$
$((P \vee Q) \vee R) \equiv ((Q \vee R) \vee P)$

• idempotence de $\wedge$ et $\vee$

$(p \wedge p) \equiv p$
$(p \vee p) \equiv p$

### Modus{ponens, tollens}

**Définition 4: modus ponen** Soit $P$ et $Q$ deux propositions, si $P$ est vrai et $P \Rightarrow Q$ est vrai, alors Q est vrai.

$$P, P \Rightarrow Q \dashv Q$$

car $P \Rightarrow q \equiv \neg Q \Rightarrow \neg P$

**Définition 5: modus ponen** Soit $P$ et $Q$ deux propositions, si $\neg Q$ est vrai et $P \Rightarrow Q$ est vrai, alors la proposition $\neg P$ est vrai.

$$\neg Q, P \Rightarrow Q \dashv \neg P$$

# TPs

### Question 1: Ecrire une fonction `interpretations(nbVar)` qui renvoie le tuple constitue de toutes les interpretations possible de nbvar variables propositionnelles

la technique que j'ai opte est de calculer tous les nombre possible en binaire jusqu'a $2^{\text{nbvar}}$, puis de les retranscrire en tuple de vrai/faux. Voici le code (on assume une fonction translate to tuple defini comme le suit)

```
# Q1: ecrire une fonction Inter(nbvar) qui renvoie le tuple constitue de toutes les interpretations possible de nbvar variables propositionnelles
def translatetotuple(binary: str):
    result = []
    for i in binary:
        if i == '1':
            result.append(True)
```

```
        else:
            result.append(False)

    return tuple(result)

def inter(nbvar):
    finalresult = ()
    for i in range(nbvar**2):
        result = bin(i)
        result = result[2:]
        while len(result)<nbvar:
            result = '0'+result
        result = translatetotuple(result)
        finalresult += result,
    return finalresult
```

### Question 2.

Une formule propositionelle FP de n variables est codee par une chiande de caracteres respectant la syntaxe python. les variables étant toujours codées V[0], V[1],… ,V[n-1]. Écrivez une fonction TV(FP,n) qui renvoie la table de vérité de la formule FP sous forme de tuple de tuples à l'aide de la fonction Inter et la fonction d'évaluation eval(chaine) du Python qui évalue une chaine de caractères si elle respecte la syntaxe du langage Python.

Exemple. Avec la chaîne de caractère FP = "V[0] and V[1]", l'appel de la fonction TV(FP,2) doit renvoyer le tuple ((False,False,False),(False,True,False), (True,False,False),(True,True,True))

le code est incomplet, mais le concept est juste. on genere les combinaison, puis on laisse exec evaluer l'expression, enfin on append dans la variable resultat final et on renvoie (le code n'est pas au point mais le concept est celui la)

```
def TV(fp, nbvar):
   variables = inter(nbvar)
   endresult = ()
   for V in variables:
       endresult += (i, (exec(fp)),)
```

## Ensembles

**Définition 6: ensemble** Un ensemble est une collection X d'objets *definis* et *unique*. un objet appartenant a l'ensemble est dit membre de *X* et on dit que l'objet et membre. un membre est unique dans un ensemble, il ne peut pas y avoir deux fois le meme element

exemple:

$$\{a, b, c, a\} = \{a, b, c\}$$

sur python, un type ensemble existe qui est appele `set`

## Predicats

**Définition 7: Predicat** enonce contenant des variables tel qu'en substituant chaque variables par une valeure choisi, on obtient une proposition

exemple: $x|P(x)$ (se lit x tel que P(x)) est un predicat dans lesquelles la proposition P(x) est vraie pour x la theorie de ZF distingue deux tyupes de predicats:

- 1. predicat collectivisant: un predicat $P(X)$ tel que les valeurs de x pour lesquelles la proposition P(x) est vrai constituent un enssemble note $(x|P(x))$
- 2. predicat non collectivisant: un predicat P(x) tel que les valeurss x pour lesquelles la prop P(X) est vraie ne constituent pas un ensemble

considerant le predicat $P(x, y)$ defini sur deux variables reelles x et y suivant:

$$x^2 - y = 1$$

on peut definir le predicat $Q(x)$ de la variable suivante:

$$\exists y \in \mathbb{R} x^2 - y = 1$$

## Quantificateurs

**Définition 8: quantificateur** Il existe 3 quantificateurs:

- $\forall$ qui se lit "pour tout" (appele forall en latex et typst
- $\exists$ qui se lit "il existe"
- $\exists!$ qui est un "il existe" unique

le quantificateur $\exists!$ est lui meme une proposition qui est:
$(\exists x \in X P(X)) \wedge (\forall x \in X \forall y \in X P(x) \wedge P(y) \Rightarrow x = y$ le terme de gauche codel'existence et le terme droit l'unicite en exprimant sous forme contraposee que deux elements distincts $x$ et $y$ de l'ensemble $X$ ne peuvent simultanement satisfaire le predicat $P(x) : x \neq\Rightarrow \neg(P(x) \wedge P(y))$.

## Axiomes

**Définition 9: axiome** Soit X et Y deux ensembles. on dit que X est inclus dans Y ou que X est une partie de Y ou encore que X est un sousensemble de Y, ce que l'ont note $X \subseteq Y$ ou $Y \supseteq X$ seulement si $\forall x x \in X \Rightarrow x \in Y$

## TP

Fiches de revisions 2nd semestre

*fiche de i22 de Mehdi Ben Ahmed*

# bases et codage

### rappels

| decimale | binaire | octal | hexa |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |

### rappels logique

| $x$ | $y$ | $x+y$ | $xy$ | $\tilde{x}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

### Arithmetique tronque a gauche:

# logique combinatoire

**Définition 10: tableau de Karnaugh** il sert a representer l'ensemble des arguments d'une fonction booleenne, a la meme facon qu'un tableau de valeur. cette forme est efficace pour trouver:
- la FND d'une fonction
- trouver l fonction booleenne ayant le moins de variable et d'operateurs possible: simplification des fonctions booleennes

un tableau de karnaugh a pour argument n, qui signifie n nombre d'arguments d'une fonction booleenne

exemple pour un tableau $n=3$:

| xy<br>z | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $f(0,0,0)$ | $f(0,1,0)$ | $f(1,1,0)$ | $f(1,0,0)$ |
| 1 | $f(0,0,1)$ | $f(0,1,1)$ | $f(1,1,1)$ | $f(1,0,1)$ |

**Définition 11: forme nominal disjonctive (FND)** let $n$ variabless, $x_1 ... x_n$, on appelle monome d'ordre n le produit $y_1, y_2 ... y_n$ avec $y_i = x_i$ ou $y_i = \widetilde{x_i}$ pour chaque $i \in \{1, ..., n\}$. une fonction est dite sous forme nominal disjonctive si la fonction est une somme de monomes d'ordre n. toute fonctions non nulle de $n$ variables peut s'ecrire de facon unique sous forme nominale disjonctive.

exemple: soit une fonction $f$ boolenne de 2 arguments dont son tableau de karnaugh est

| $xy$<br>$z$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |

La FND de $f$ est $f(x,y,z) = \tilde{x}\tilde{y}\tilde{z} + xy\tilde{z} + x\tilde{y}\tilde{z} + \tilde{x} + yz$ on peut donc simplifier la fonction a

$$xy + x\tilde{y} = x$$

en effet $xy + x\tilde{y} = x(y + \tilde{y}) = x1 = x$

# logique sequentielle

*fiche de i21 de Mehdi Ben Ahmed*

## Conception d'algorithme

**Définition 12: Invalider un algo** Pour montrer qu'un algo n'est pas valide, il suffit de montrer un contre exemple, soit un cas ou l'algorithme ne marcherait pas

## Analyse asymptotique

**Définition 13: Analyser un algorithme** c'est analyser les couts par rapport au temps d'execution, l'espace memoire, et la consommation electrique

**Définition 14: le modele random access machine** machine hypothetique ou:
- les operands consomment une unite de temps
- les boucles depend du nombre d'iterations et des operation inside
- un read consomme une unite de temps
- la memoire est illimite

l'efficacite d'un algo est defini par une fonction notee C(n) ou T(n), meme si dans un cas reel ca serait plutot note O(n)

exemple:
- recherche d'un element:
  - n cases a tester
  - 5 cases: > 5 tests
  - 10 cases: > 10 tests
- ramasssage de plots:
  - n! chemins a tester
  - 5 plots: 120 chemins possible

la notation est qui suit:
- $\Omega(n)$: meilleur cas
- $O(n)$: pire cas
- $\Theta(n)$: cas moyen

**Définition 15:** $f(n) = O(g(n))$ il existe une constance $c$ et un entier $n_0$ tels que $\forall n \geq n_0, f(n) =< cg(n)$

exemples:
- $3n^2 - n + 6 = O(n^2)$ en prenant $c = 3$ et $n_0 = 6$
- $3n^2 - n + 6 = O(n^3)$ en prenant $c = 1$ et $n_0 = 4$
- $3n^2 - n + 6 \neq O(n)$ car $\forall c, cn < 3n^2 - n + 6$ quand $n > c + 1$

**Définition 16:** $f(n) = \Omega(g(n))$ il existe une constance $c$ et un entier $n_0$ tels que $\forall n \geq n_0, f(n) \Rightarrow cg(n)$

exemples:
- $3n^2 - n + 6 = \Omega(n^2)$ en prenant $c = 2$ et $n_0 = 2$
- $3n^2 - n + 6 \neq \Omega(n^3) \forall c, 3n^2 - n + 6 < cn^3$ quand $cn > 3$ et $n > 6$
- $3n^2 - n + 6 = \Omega(n)$ en prenant $c = 1$ et $n_0 = 1$

**Définition 17:** $f(n) = \Theta(g(n))$ il existe une constance $c$ et un entier $n_0$ tels que $\forall n \geq n_0 c_1 g(n) \leq f(n) \leq c_2 g(n)$ ( $f(n) - O((g(n)$ et $f(n) = \Omega(g((n)))$ )

- $3n^2 - n + 6 = \Theta(n^2)$
- $3n^2 - n + 6 = \Theta(n^3)$
- $3n^2 - n + 6 = \Theta(n)$

## Bases d'algo

## Algos de tri

## Algos de recherche

## piles et files