

Definitions et props

Définition 1: Commutatif les variables peuvent etre inverses

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

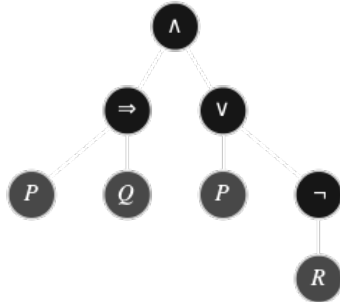


Figure 1: $(P \Rightarrow Q) \wedge (P \vee \neg R)$

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 mentionne autrement

table de \wedge : q binaire

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

table de \vee : q binaire

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

table de \oplus : q binaire

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

table de \Rightarrow : q binaire dit non commutatif

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

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

table de \Leftrightarrow : q binaire

⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥
⊥	⊥	⊥

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

$$((p \wedge q) \wedge R) \equiv ((q \wedge R) \wedge p) \quad ((p \vee q) \vee R) \equiv ((q \vee R) \vee p)$$

- idempotence de \wedge et \vee

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

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

Modus{ponens, tollens}

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

$$P, P \Rightarrow Q \vdash Q$$

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

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

$$\neg Q, P \Rightarrow Q \vdash \neg P$$

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^{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 translate_totuple(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 = translatoTuple(result)
        finalresult += result,
    return finalresult

```

Question 2.

Une formule propositionnelle FP de n variables est codée par une chaîne de caractères 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 chaîne de caractères si elle respecte la syntaxe du langage Python.

Exemple. Avec la chaîne de caractère $FP = "V[0] \text{ and } V[1]"$, 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 combinaisons, puis on laisse `exec` évaluer 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 là)

```

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

```

axiomes et predicats

Ensembles

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

exemple:

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

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

Définition 7: Difference Soit X et Y deux ensembles. la différence entre les ensembles X et Y est l'ensemble $\{x \in X \mid x \notin Y\}$, qui est l'ensemble qui contient les éléments de X mais pas les éléments de Y . on note aussi XY l'ensemble qui contient seulement les différences d'un ensemble $X \cap Y$ est $X \Delta Y$

Définition 8: Cardinal On appelle le cardinal d'un ensemble sa taille. Lorsqu'un ensemble est fini, le cardinal est la longueur de cet ensemble

Predicats

Définition 9: Predicat énonce contenant des variables tel qu'en substituant chaque variable par une valeur choisie, on obtient une proposition

exemple: $x|P(x)$ (se lit x tel que $P(x)$) est un prédicat dans lesquelles la proposition $P(x)$ est vraie pour x la théorie de ZF distingue deux types de prédicats:

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

considérant le prédicat $P(x, y)$ défini sur deux variables réelles x et y suivant:

$$x^2 - y = 1$$

on peut définir le prédicat $Q(x)$ de la variable suivante:

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

Quantificateurs

Définition 10: quantificateur Il existe 3 quantificateurs:

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

le quantificateur $\exists!$ est lui-même 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 code l'existence et le terme droit l'unicité en exprimant sous forme contraposée que deux éléments distincts x et y de l'ensemble X ne peuvent simultanément satisfaire le prédicat $P(x) : x \neq y \Rightarrow \neg(P(x) \wedge P(y))$.

Axiomes

Définition 11: 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

logique de boole

Algre de boole

soit \mathbb{B} un ensemble munit d'une structure algebrique, on l'appelle algre de boole.

Définition 12: on appelle boolean toute variable defini sur un ensemble a deux elements

Pour simplifier l'écriture des expressions logique, l'operande \neg peut etre ecrit de cette facon: \bar{x} . et on a

x	0	1
\bar{x}	1	0

dans le cadre de l'algre de Boole, un litterale designe la aussi une variable x (litteral positif) ou sa negation \bar{x} (litteral negatif)

Proprietes de calcul

on dispose des nombreuses proprietess suivantes heritees du calcul propositionnel:

1. associativite: $(a + b) + c = a + (b + c) = a + b + c$
2. commutativite $a + b = b + a$
3. distributivite $a(b + c) = ab + (ac)$
4. idempotence: $a + a + a + a \dots = a$ et $aaa \dots = a$
5. element neutre: $a + 0 = 0 + a = a$ et $a1 = 1a = a$
6. absorption $0a = a$ et $1 + a = 1$
7. simplification: $a + \bar{a}b = a + b$ et $a(\bar{a} + b) = ab$
8. redondance: $ab + \bar{a}c = ab + \bar{a}c + bc$ et $(a + b)(\bar{a} + c) = (a + b)(\bar{a} + c)(b + c)$
9. DeMorgan: $\overline{ab} = \bar{a} + \bar{b}$
10. Involution: $\bar{\bar{a}} = a$
11. tiers exclu: $\bar{a} + a = 1$
12. non contradiction: $a\bar{a} = 0$

on retrouve les cinq autres operateur binaire, implication, equifvalence, disjonction exclusive, non conjonction et non disjonction:

$$\begin{aligned} a \Rightarrow b &= \bar{a} + b, \\ a \Leftrightarrow b &= (\bar{a} + b)(a + \bar{b}) \\ a \oplus b &= (a + b)(\bar{a} + \bar{b}) \\ a \uparrow b &= \bar{a}\bar{b} \\ a \downarrow b &= \widetilde{a + b} \end{aligned}$$

qui ont les tables de verite:

\Rightarrow	0	1
---------------	---	---

0	1	1
1	0	1

\Leftrightarrow	0	1
0	1	0
1	0	1

\oplus	0	1
0	0	1
1	1	0

\uparrow	0	1
0	1	1
1	1	0

\downarrow	0	1
0	1	0
1	0	0

Definitions:

Définition 13: antilogie L'antilogie est le cas ou une formule repond toujours faux, a l'inverse de la tautologie qui repond toujours vrai

Code Gray

Définition 14: Code Gray

Minterme, maxterme

Définition 15: Minterme, Maxterme on appelle Minterme toute fonction d'ordre n , prenant une seule fois la valeur 1

Relations et applications

analyse combinatoire

Ensembles naturel

Définition 16: Ensemble Naturel On appelle ensemble naturel (N, \preceq) tout ensemble ordonne qui satisfait les trois proprietes suivantes:

- Toute partie non vide admet un plus petit element
- Toute partie non vide et majoree admet un plus grand element
- L'ensemble n'admet pas de plus grand element

l'existence d'un ensemble naturel est acquise grace a l'axiome de l'infini (consulter wikipedia) Pour demontrer qu'un ensemble naturel est ordonne, on peut emettre la proposition suivante:

$$\exists m \in \{a, b\} (m \preceq a) \wedge (m \preceq b)$$

deux element a, b dans l'ensemble N . D'apres l'axiome de la paire, l'ensemble $\{a, b\}$ existe, n'est pas vide et admet donc un plus petit element (un ensemble naturel est toujours minore mais jamais majore) Comme $m \in N$ on a $(m = a) \wedge (m = b)$ et on deduit que $(a \preceq b) \vee (b \preceq a)$

Soit $n \in \mathbb{N}$ la demi droite $]n, \rightarrow[$ n'est pas vide, sinon n serait le plus grand element, ce qui va a l'encontre de la 3eme propriete. $]n, \rightarrow[$ admet un plus petit element appele $\text{succ}(n)$, le successeur de n

recurrence

Définition 17: Theoreme principe de recurrence Toute partie de \mathbb{N} qui contient 0 et stable pour l'application successeur est egale a \mathbb{N}

Définition 18: theoreme recurrence simple

Définition 19: theoreme recurrence forte

Définition 20: theoreme recurrence multiple

Définition 21: theoreme recurrence finie

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 22: tableau de Karnaugh il sert a représenter l'ensemble des arguments d'une fonction booléenne, a la meme façon qu'un tableau de valeur. cette forme est efficace pour trouver:

- la FND d'une fonction
- trouver la fonction booléenne ayant le moins de variable et d'opérateurs possible: simplification des fonctions booléennes

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

exemple pour un tableau $n=3$:

xy	00	01	11	10
z				
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 23: forme nominal disjonctive (FND) let n variables, $x_1...x_n$, on appelle monome d'ordre n le produit $y_1, y_2...y_n$ avec $y_i = x_i$ ou $y_i = \tilde{x}_i$ 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'écrire de façon unique sous forme nominale disjonctive.

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

xy	00	01	11	10
z				
0	1	0	1	1
1	0	1	0	0

La FND de f est $f(x, y, z) = \tilde{x}\tilde{y}\tilde{z} + xy\tilde{z} + x\tilde{y}z + \tilde{x} + yz$
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

Conception d'algorithme

Définition 24: Invalider un algo Pour montrer qu'un algo n'est pas valide, il suffit de montrer un contre exemple, soit un cas où l'algorithme ne marcherait pas

Analyse asymptotique

Définition 25: Analyser un algorithme c'est analyser les couts par rapport au temps d'exécution, l'espace mémoire, et la consommation électrique

Définition 26: le modele random access machine machine hypothétique où:

- 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
- ramassage 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 27: $f(n) = O(g(n))$ il existe une constance c et un entier n_0 tels que $\forall n \geq n_0, f(n) \leq cg(n)$

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

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 29: $f(n) = \Theta(g(n))$ il existe une constance c et un entier n_0 tels que $\forall n \geq n_0, c_1g(n) \leq f(n) \leq c_2g(n)$
($f(n) = O(g(n))$ 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

parcours de tableau

Algos de recherche

Algos de tri

3 types d'algos de tri:

- tri par selection
- tri par propagation
- tri par insertion

tout les algorithmes reposent sur une méthode d'échange d'items basé sur leurs indice:

```
def swap(T, i, j):
    a = T[i]
    T[i] = T[j]
    T[j] = a
```

tri par selection

bien que simple, l'algorithme est considere comme inefficace a cause de son temps d'exécution quadratique

$$\Omega(n) = n^2$$

$$O(n) = n^2$$

$$\Theta(n) = n^2$$

il consiste a parcourir une liste, et echanger le plus petit element par le premier, puis d'avancer l'indice du premier jusqu'a finir de parcourir la liste

```
procédure tri_selection(tableau t)
    n ← longueur(t)
    pour i de 0 à n - 2
        min ← i
        pour j de i + 1 à n - 1
            si t[j] < t[min], alors min ← j
        fin pour
        si min ≠ i, alors échanger t[i] et t[min]
    fin pour
fin procédure
```

tri par propagation (Bubble sort)

il a une complexite de n^2 , sauf pour le meilleur cas, où $\Omega(n) = n$ il consiste a echanger les elements qui sont dans le désordre ($n+1 < n$), a la fin de chaque iteration, le dernier element est le plus grand, donc l'indice est soustre a chaque fois

```
tri_à_bulles(Tableau T)
    pour i allant de (taille de T)-1 à 1
        pour j allant de 0 à i-1
            si T[j+1] < T[j]
                (T[j+1], T[j]) ← (T[j], T[j+1])
```

tri par insertion

il parcourt la liste, et a chaque fois que l'algo trouve un nombre inferieur au precedent, il revient en arriere pour le placer correctement

```
procédure tri_insertion(tableau T)

    pour i de 1 à taille(T) - 1

        # mémoriser T[i] dans x
        x ← T[i]

        # décaler les éléments T[0]..T[i-1] qui sont
plus grands que x, en partant de T[i-1]
        j ← i
        tant que j > 0 et T[j - 1] > x
            T[j] ← T[j - 1]
            j ← j - 1

        # placer x dans le "trou" laissé par le
décalage
        T[j] ← x
```

Algos de recherche

Recheche Dichotomique