

Definitions et props

Définition 1: Commutatif les variables peuvent etre inverses

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

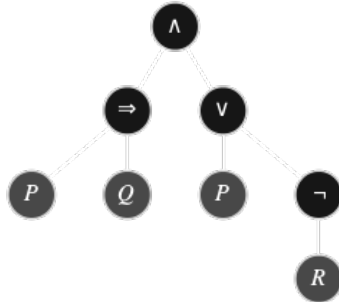


Figure 1: $(P \Rightarrow Q) \wedge (P \vee \neg R)$

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

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

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

- 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 = translate_totuple(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)),)

```

Ensembles

Définition 6: ensemble Un ensemble est une collection X d'objets *definies* 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`

Predicats

Définition 7: 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 8: 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 $P(x) : x \not\Rightarrow \neg(P(x) \wedge P(y))$.

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 sous-ensemble de Y , ce que l'on note $X \subseteq Y$ ou $Y \supseteq X$ seulement si $\forall x x \in X \Rightarrow x \in Y$

TP