TD4 : Logique.

Exercice 1 : DPLL

Déterminez si ces formules sont satisfiables à l'aide de l'algorithme DPLL

Q 1.1 $\varphi_1 = p \wedge q \wedge r$

Q 1.2 $\varphi_2 = (q \vee \neg p) \wedge \neg q \wedge p \wedge (r \vee \neg p)$

Q 1.3 $\varphi_3 = (q \vee p) \wedge (r \vee q) \wedge \neg r \wedge \neg q$

Q 1.4 $\varphi_4 = p \wedge q \wedge (q \vee \neg p) \wedge r$

Q 1.5 $\varphi_5 = (q \vee p) \wedge \neg q$

Q 1.6 $\varphi_6 = (q \vee p) \wedge (q \vee \neg p) \wedge (\neg r \vee \neg q) \wedge (r \vee \neg q)$

Q 1.7 $\varphi_7 = (\neg p \vee t) \wedge (p \vee \neg r) \wedge (q \vee r) \wedge (\neg q \vee s)$

Q1.1 : PU : On met p=1.
- On met q=1. SAT.
- On met R=1.

Q1.2 $\ell_2 : (q \vee \neg p) \wedge \neg q \wedge p \wedge (R \vee \neg p)$

PU : $p=1 \rightarrow q \wedge \neg q$
Simplification $\rightarrow R=1$
$\ell_2 \equiv q \wedge \neg q$

| pas SAT.

Q1.3 $(q \vee p) \wedge (R \vee q) \wedge \neg R \wedge \neg q : \varphi_3$

PU : $R=0 \rightarrow \varphi_3 \equiv (q \vee p) \wedge q \wedge \neg q$
Simplifiat° $\Rightarrow \varphi_3 \equiv q \vee p$ SAT.

Q1.4. $p \wedge q \wedge (q \vee \neg p) \wedge R$

PU : $q = 1 \Rightarrow \varphi_4 \equiv p \wedge R$ SAT.

Q1.5 $\varphi_5 : (q \vee p) \wedge \neg q$

PV : $q = 0 \Rightarrow$ SAT

Q1.6 $\varphi_6 : (q \vee p) \wedge (q \vee \neg p) \wedge (\neg R \vee \neg q) \wedge (R \vee \neg q)$ | Pas SAT.

CPC : $q = 1 \quad \varphi_6 \equiv \neg R \wedge R$

$q = 0 \quad \varphi_6 \equiv p \wedge \neg p$

. . .

Exercice 2 : Encore DPLL

Trouver si possible une valuation qui satisfasse les formules suivantes en utilisant l'algorithme DPLL. Mettez les formules sous forme normale conjonctive si nécessaire.

Q 2.1 $(a \vee b \wedge \neg c) \Leftrightarrow (a \wedge (c \vee \neg b))$ φ_1

Q 2.2 $(((b \Rightarrow \neg c) \wedge (d \Rightarrow e)) \vee a) \wedge (c \vee ((a \Rightarrow \neg f) \wedge (\neg b \Rightarrow f))) \wedge (b \vee d \vee \neg e)$

$(a \Leftrightarrow b) \equiv (a \wedge b) \vee (\neg a \wedge \neg b)$

$\equiv (a \vee \neg a) \wedge (a \vee \neg b) \wedge (b \vee \neg a) \wedge (b \vee \neg b)$

$\equiv (a \vee \neg b) \wedge (b \vee \neg a)$.

$\varphi_1 \equiv (((a \vee b) \wedge \neg c) \vee \neg \ldots)$

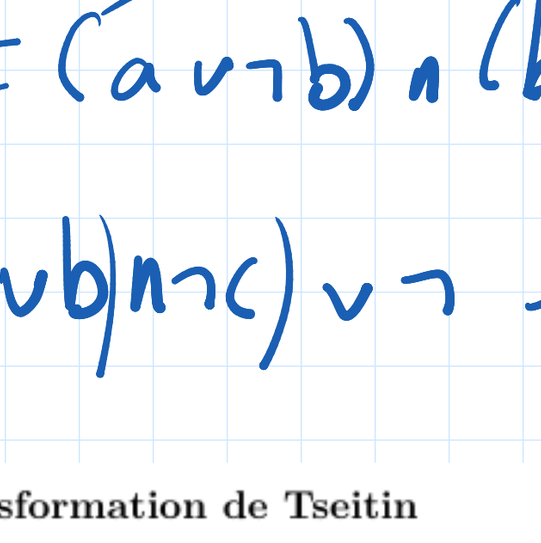
Exercice 3 : Transformation de Tseitin

Q 3.1 Convertir la formule $(p \wedge q) \Rightarrow r$ en formule CNF équisatisfaisable (en utilisant la Transformation de Tseitin).

Q 3.2 Convertir la formule $(p \vee q) \Rightarrow (p \wedge \neg r)$ en formule CNF équisatisfaisable (en utilisant la Transformation de Tseitin).

Q 3.3 Convertir la formule $\neg(p \vee (q \Rightarrow r)) \vee s$ en formule CNF équisatisfaisable (en utilisant la Transformation de Tseitin).

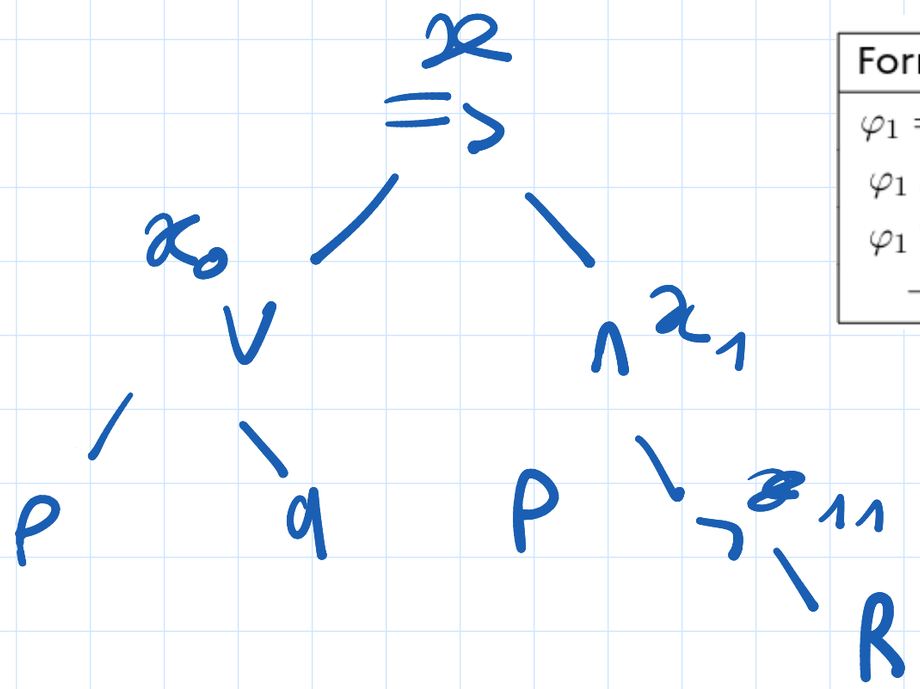
3.1



$\varphi_0 = (\neg p \vee \neg q \vee x_0) \wedge (\neg x \vee p) \wedge (\neg x \vee q)$

$\varphi = \varphi_0 \wedge (\neg x_0 \vee R \vee \neg x) \wedge (x \vee x_0) \wedge (x \vee \neg R)$

3.2.



Formule	Équivalence	Formule θ
$\varphi_1 \Rightarrow \varphi_2$	$x_1 \Rightarrow x_2 \Leftrightarrow x$	$(\neg x_1 \vee x_2 \vee \neg x) \wedge (x \vee x_1) \wedge (x \vee \neg x_2)$
$\varphi_1 \wedge \varphi_2$	$x_1 \wedge x_2 \Leftrightarrow x$	$(\neg x_1 \vee \neg x_2 \vee x) \wedge (\neg x \vee x_1) \wedge (\neg x \vee x_2)$
$\varphi_1 \vee \varphi_2$	$(x_1 \vee x_2) \Leftrightarrow x$	$(x1 \vee x2 \vee \neg x) \wedge (\neg x_1 \vee x) \wedge (\neg x_2 \vee x)$
$\neg\varphi$	$\neg y \Leftrightarrow x$	$(x \vee y) \wedge (\neg x \vee \neg y)$

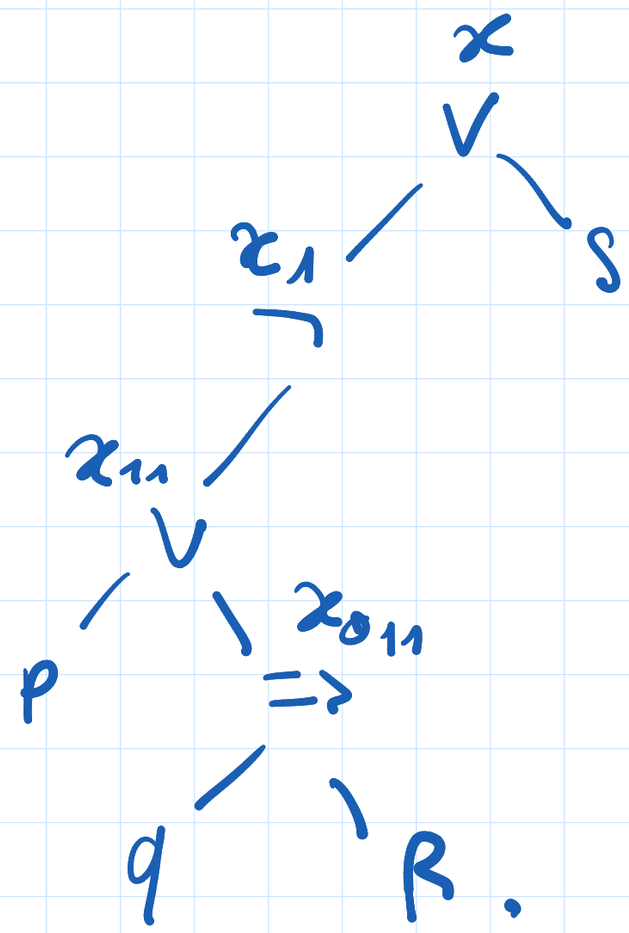
$\varphi_0 : (p \vee q \vee \neg x_0) \wedge (\neg p \vee x_0) \wedge (\neg q \vee x_0)$

$\varphi_{11} : (R \vee x_{11}) \wedge (\neg R \vee \neg x_{11})$

$\varphi_1 : (\neg p \vee \neg x_{11} \vee x_1) \wedge (\neg x_1 \vee p) \wedge (\neg x_1 \vee x_{11}) \wedge \varphi_{11}.$

$\varphi : \varphi_0 \wedge \varphi_1 \wedge (\neg x_0 \vee \neg x_1 \vee \neg x) \wedge (x \vee x_0) \wedge (x \vee \neg x_1)$

Q π.



Formule	Équivalence	Formule θ
$\varphi_1 \Rightarrow \varphi_2$	$x_1 \Rightarrow x_2 \Leftrightarrow x$	$(\neg x_1 \vee x_2 \vee \neg x) \wedge (x \vee x_1) \wedge (x \vee \neg x_2)$
$\varphi_1 \wedge \varphi_2$	$x_1 \wedge x_2 \Leftrightarrow x$	$(\neg x_1 \vee \neg x_2 \vee x) \wedge (\neg x \vee x_1) \wedge (\neg x \vee x_2)$
$\varphi_1 \vee \varphi_2$	$(x_1 \vee x_2) \Leftrightarrow x$	$(x1 \vee x2 \vee \neg x) \wedge (\neg x_1 \vee x) \wedge (\neg x_2 \vee x)$
$\neg\varphi$	$\neg y \Leftrightarrow x$	$(x \vee y) \wedge (\neg x \vee \neg y)$

$\varphi_{011} : (\neg q \vee R \vee \neg x_{011}) \wedge (x_{11} \vee q) \wedge (x_{11} \vee \neg R)$

$\varphi_{11} : \varphi_{011} \wedge (\neg p \vee \neg x_{011} \vee x) \wedge (\neg x \vee p) \wedge (\neg x \vee x_{011})$

$\varphi_1 : \varphi_{11} \wedge (x_1 \vee x_{11}) \wedge (\neg x_1 \vee \neg x_{11})$

$\varphi : \varphi_1 \wedge (x_1 \vee s \vee \neg x) \wedge (x \vee x_1) \wedge (x \vee \neg s)$

Q 2.2 $\varphi_2 (((b \Rightarrow \neg c) \wedge (d \Rightarrow e)) \vee a) \wedge (c \vee ((a \Rightarrow \neg f) \wedge (\neg b \Rightarrow f))) \wedge (b \vee d \vee \neg e)$

$\varphi_2 : (((\neg b \vee \neg c) \wedge (\neg d \vee e)) \vee a) \wedge (c \vee ((a \vee \neg f) \wedge (b \vee f))) \wedge$
$(b \vee d \vee \neg e)$

$\equiv (a \vee \neg b \vee \neg c) \wedge (\neg d \vee e \vee a) \wedge (c \vee \neg a \vee \neg f) \wedge (c \vee b \vee f)$
$\wedge (b \vee d \vee \neg e)$

CPC

$\varphi_2' : a = 1. \rightarrow (c \vee \neg f) \wedge (c \vee b \vee f) \wedge (b \vee d \vee \neg e)$

CPC $c = 1$ $\varphi_2' \rightarrow b \vee d \vee \neg e$

PU : $b = 1 \rightarrow d \vee \neg e$ ... $d = 1$ $e = 0$.

$a = 1 = b = c = d = 1$, $e = 0$

Exercice 4 : Séquentialité des opérateurs booléens en programmation

Q 4.1 Propriété de l'exponentielle

Nous souhaitons définir une fonction `test_exp(x,y)` qui teste une propriété de x^y. Cette propriété est implémentée par une fonction `prop` qui prend en argument un flottant et renvoie un booléen. Avec les librairies `python` que nous utilisons et nous définissons x^y de la manière suivant les signes respectifs de x et y :

- lorsque x est strictement positif, x^y est défini par $\texttt{exp}(y * \texttt{ln}(x))$,
- lorsque x est strictement négatif et y est un entier, alors x^y est défini par : $x ** y$
- lorsque x est nul, x^y est 0, (généralement on définit 0^0 par 1, mais pour notre problème nous préférons utiliser 0 dans ce cas),
- dans les autres cas, x^y est considéré comme non défini.

Nous supposons que nous disposons d'une fonction `is_int` qui prend en argument un flottant est renvoie `True` si celui-ci correspond à un entier et `False` sinon.

Écrire la fonction `test_exp(x,y)` en faisant en sorte qu'elle renvoie `False` lorsque x^y n'est pas défini et en complétant le code suivant :


```
def test_exp(x,y):
    return ...
```

Q 4.2 Tirage équitable de Von Neumann

On suppose que nous avons accès à une fonction qui tire un booléen au hasard `rand_bool()`. Donner les conditions dans lesquelles la fonction `python` suivante ne fait pas d'appel récursif. Préciser la valeur renvoyée dans ces conditions.

```
def von_neumann():
    a = rand_bool()
    b = rand_bool()
    return a and not b or (a or not b) and von_neumann()
```

4.1 : Return is_int(x)

4.2. von_neumann ne fait pas d'appel Récursif si :

a and not b or (a or not b) est False

a -> False

a -> False , b -> Vrai :

b -> True

Voilà