Exercice 1 :

P,Q : symboles de prédicats. P est unaire et Q binaire.
a,b : constantes (de termes).

Dire si les formules sont bien formées ou non.
Proposer une correction si besoin est.

1. $\exists x.P(x) \Rightarrow P(a) \wedge P(b)$
Bien formée ? oui.

2. $\forall x.P(x) \wedge a$
"a" est un terme.
Le "$\wedge$" ne prend en arguments que des formules.
Correction proposée : $\forall x.P(x) \wedge P(a)$

3. $\forall x.P(P(x))$
P(x) = bien formé, un prédicat appliqué à un terme, c'est donc une formule.
P(P(x)) = mal formé car P attend un terme et non une formule.
Correction : $\forall x.P(x)$

4. $\forall P.\forall x.P(x) \Rightarrow \exists y.Q(x, y)$
On ne peut pas quantifier sur les symboles de prédicats et de fonctions en logique du premier ordre.
SI on peut quantifier sur les prédicats/fonctions, on a une logique d'ordre supérieure.
Correction : $\forall x.P(x) \Rightarrow \exists y.Q(x, y)$

5. $\forall x.P(x) \Rightarrow Q(x,a)$
Q(x,a) est bien formé : une formule
P(x) bien formé : une formule
$P(x) \Rightarrow Q(x,a)$: une formule
$\forall x.P(x) \Rightarrow Q(x,a)$: une formule
Bien formé.

Exercice 2 :

$A \equiv \forall x.P(x) \Rightarrow \exists y.Q(x, y)$
$B \equiv (\forall x.P(x)) \Rightarrow \exists y.Q(x, y)$
$C \equiv \forall x.\exists y.Q(x, y) \wedge \exists x.\neg Q(y, x)$

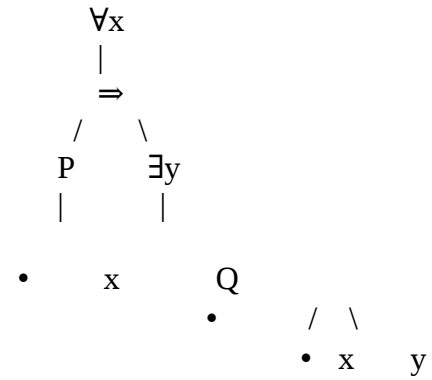
Règles (convention) :

- $\wedge$, $\vee$, et $\Leftrightarrow$ associent à gauche.
- $\Rightarrow$ associe à droite.
- $\neg > \wedge > \vee > \Rightarrow > \Leftrightarrow$.
- La portée d'un quantificateur va jusqu'à la parenthèse fermante de la formule du quantificateur.

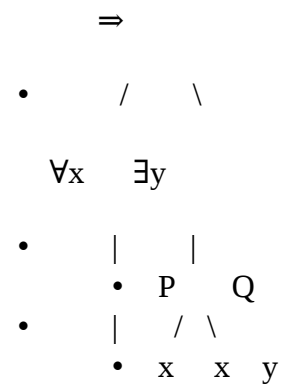
1. $A \equiv \forall x.(P(x) \Rightarrow \exists y.Q(x,y))$
$B \equiv (\forall x.P(x)) \Rightarrow \exists y.Q(x,y)$

C ≡ ∀x.(∃y.(Q(x,y) ∧ ∃x.¬Q(y,x)"zq))

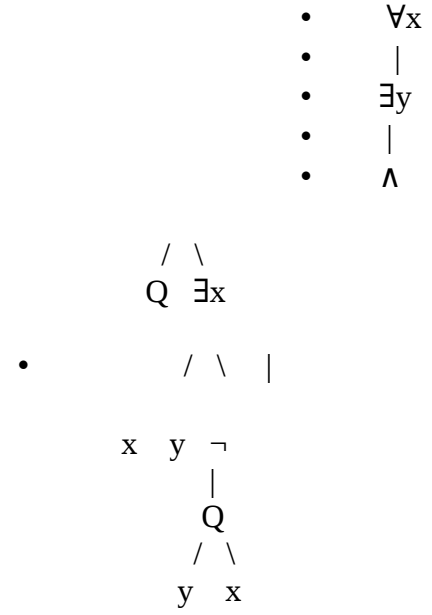
2.
A ≡ ∀x.(P(x) ⇒ ∃y.Q(x,y))



B ≡ (∀x.P(x)) ⇒ ∃y.Q(x,y)



C ≡ ∀x.(∃y.(Q(x,y) ∧ ∃x.¬Q(y,x)))



3.
Algo. :
On part d'une variable qui apparaît en feuille de l'arbre (une occurrence).
On remonte dans l'arbre jusqu'à la racine.
Si on croise un quantificateur qui porte sur le nom de la variable alors la variable est liée.
Si on arrive à la racine et qu'on n'a croisé aucun quantificateur qui porte sur le nom de la variable alors la variable libre.

4.
On peut soit appliquer notre algo sur les arbres, soit appliquer les fonctions FV et BV du cours.

FV(A) = {}
BV(A) = {x,y}
Les 2 occurrences de x sont liées et l'occurrence de y est liée.
A est une formule close.

FV(B) = {x}
BV(B) = {x,y}
1 occurrence de x est liée (celle la plus à gauche) et l'autre est libre.
L'occurrence de y est liée.
Donc x est libre et liée à la fois.
B n'est pas close.

FV(C) = {}
BV(C) = {x,y}
1 occurrence de x (la plus a gauche) est liée par le ∀x.
1 occurrence de x (la plus à droite) est liée par le ∃x.
Les 2 occurences de y sont liées par le ∃y.
C est une formule close.

5. Règle du cours :

Une formule est polie ou propre si aucune variable n’est à la fois libre et liée dans cette formule, et si aucune variable liée n’est soumise à plus d’une quantification

A propre ? Oui.
B propre ? Non
Correction : $(\forall z.P(z)) \Rightarrow \exists y.Q(x, y)$
Règle : on renomme toujours les variables liées (alpha-conversion) et non les variables libres.
C propre ? non 2 quantificateurs pour x
Correction : $\forall x.\exists y.Q(x, y) \wedge \exists z.\neg Q(y, z)$ ou $\forall z.\exists y.Q(z, y) \wedge \exists x.\neg Q(y, x)$

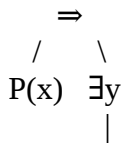
Exercice 3 :

1. Profondeur d'une formule :

<u>Version où les prédicats sont des feuilles des arbres syntaxiques</u> :

Pour A dans l'exo 2 :

∀x
|



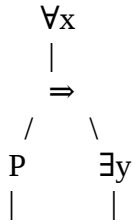
- Q(x,y)

Prof(F) : F -> N (entier)

- Si F = P(t1,...,tn) alors Prof(P(t1 , . . . , tn)) = 0.
- Si F = ⊤ ou ⊥ alors Prof(F) = 0.
- Si F =¬Φ alors Prof(F) = 1 + Prof(Φ)
- Si F = Φ ∧ Φ' ou Φ ∨ Φ' ou Φ ⇒ Φ' ou Φ ⇔ Φ' alors Prof(F) = 1 + max((Prof(Φ),Prof(Φ'))
- Si F = ∀x.Φ ou ∃x.Φ alors Prof(F) = 1 + Prof(Φ)

<u>Version où on dessine aussi les arbres syntaxiques des termes</u> :

Pour A dans l'exo 2 :



- x Q
  - / \
    - x y

Prof_terme : T -> N (entier)

- Si t = x alors Prof_terme(x) = 0
- Si t = f(t1,...,tn) alors Prof_terme(t) = 1 + max(Prof_terme(t1),...,Prof_terme(tn))

Prof(F) : F -> N (entier)

- Si F = P(t1,...,tn) alors Prof(P(t1 , . . . , tn)) = 1 + max(Prof_terme(t1),...,Prof_terme(tn))
- Si F = ⊤ ou ⊥ alors Prof(F) = 0.
- Si F =¬Φ alors Prof(F) = 1 + Prof(Φ)
- Si F = Φ ∧ Φ' ou Φ ∨ Φ' ou Φ ⇒ Φ' ou Φ ⇔ Φ' alors Prof(F) = 1 + max((Prof(Φ),Prof(Φ'))
- Si F = ∀x.Φ ou ∃x.Φ alors Prof(F) = 1 + Prof(Φ)

2. Nombre de connecteurs d'une formule :

Nbc(F) : F -> N (entier)

- Si F = P(t1,...,tn) alors Nbc(F) = 0
- Si F = ⊤ ou ⊥ alors Nbc(F) = 0
- Si F =¬Φ alors Nbc(F) = 1 + Nbc(Φ)
- Si F = Φ ∧ Φ' ou Φ ∨ Φ' ou Φ ⇒ Φ' ou Φ ⇔ Φ' alors Nbc(F) = 1 + Nbc(Φ) + Nbc(Φ')
- Si F = ∀x.Φ ou ∃x.Φ alors Nbc(F) = Nbc(Φ)

3. Nombre de quantificateurs d'une formule ;

Nquant(F) : F -> N (entier)

- Si F = P(t1,...,tn) alors Nquant(F) = 0
- Si F = ⊤ ou ⊥ alors Nquant(F) = 0
- Si F =¬Φ alors Nquant(F) = Nquant (Φ)
- Si F = Φ ∧ Φ' ou Φ ∨ Φ' ou Φ ⇒ Φ' ou Φ ⇔ Φ' alors Nquant (F) = Nquant(Φ)+Nquant(Φ')
- Si F = ∀x.Φ ou ∃x.Φ alors Nquant(F) = 1 + Nquant(Φ)

4. Nombre de sous-formules d'une formule :

NbSF(F) : F -> N

- Si F = P(t1,...,tn) alors NbSF(F) = 1
- Si F = ⊤ ou ⊥ alors NbSF(F) = 1
- Si F =¬Φ alors NbSF(F) = 1 + NbSF(Φ)
- Si F = Φ ∧ Φ' ou Φ ∨ Φ' ou Φ ⇒ Φ' ou Φ ⇔ Φ' alors NbSF (F) = 1 + NbSF(Φ) + NbSF(Φ')
- Si F = ∀x.Φ ou ∃x.Φ alors NbSF(F) = 1 + NbSF(Φ)

Exercice 4 :

1. Les chiens et les oiseaux sont des animaux domestiques.

C(x) = x est un chien
O(x) = x est un oiseau
D(x) = x est un animal domestique

∀x.(C(x) ∨ O(x) ⇒ D(x))

2. Tobby est un chien qui aime les enfants.

C(x) = x est un chien
t = Tobby
E(x) = x est un enfant
A(x,y) = x aime y (prédicat binaire : on appelle ça une relation)

C(t) ∧ (∀x.E(x) ⇒ A(t,x))

3. Les oiseaux n'aiment pas les chiens.

O(x) = x est un oiseau

C(x) = x est un chien
A(x,y) = x aime y

$\forall x,y.(O(x) \wedge C(y) \Rightarrow \neg A(x,y))$
$\forall x.(O(x) \wedge \forall y.(C(y) \Rightarrow \neg A(x,y)))$
$\forall x,y.(O(x) \Rightarrow (C(y) \Rightarrow \neg A(x, y)))$
$\forall x.(O(x) \Rightarrow \forall y.(C(y) \Rightarrow \neg A(x, y)))$

4. Serge aime tous les animaux domestiques sauf les chiens.

s = Serge
D(x) = x un animal domestique
C(x) x est un chien
A(x,y) = x aime y

$\forall x.((D(x) \wedge \neg C(x)) \Rightarrow A(s,x))$
$\forall x.(D(x) \Rightarrow (\neg C(x) \Rightarrow A(s,x)))$

5. Tous les enfants n'ont pas peur des chiens

E(x) = x est un enfant
C(x) = x est un chien
P(x,y) = x a peur de y

$\neg(\forall x,y.E(x) \wedge C(y) \Rightarrow P(x,y))$
$\neg(\forall x,y.(E(x) \Rightarrow (C(y) \Rightarrow P(x, y))))$

$\neg(\forall x,y.E(x) \wedge C(y) \Rightarrow P(x,y))$
$= \exists x.\neg(\forall y.E(x) \wedge C(y) \Rightarrow P(x,y))$
$= \exists x,y.\neg(E(x) \wedge C(y) \Rightarrow P(x,y))$
$= \exists x,y.\neg(\neg(E(x) \wedge C(y)) \vee P(x,y))$
$= \exists x,y..\neg\neg(E(x) \wedge C(y)) \wedge \neg P(x,y)$
$= \exists x,y.E(x) \wedge C(y) \wedge \neg P(x,y)$

Règles de manipulation syntaxique :
$\neg\forall x.P(x) = \exists x.\neg P(x)$

6. Certains chiens aiment les enfants.

E(x) = x est un enfant
C(x) = x est un chien
A(x,y) = x aime y

$\exists x.(C(x) \wedge (\forall y.E(y) \Rightarrow A(x, y)))$

7. Certains chiens aiment les enfants et réciproquement.

E(x) = x est un enfant
C(x) = x est un chien
A(x,y) = x aime y

$(\exists x.(C(x) \wedge (\forall y.E(y) \Rightarrow A(x, y)))) \wedge (\exists x.(E(x) \wedge (\forall y.C(y) \Rightarrow A(x, y))))$

8. Les enfants aiment certains chiens.

E(x) = x est un enfant
C(x) = x est un chien
A(x,y) = x aime y

Les enfants aiment un chien mais pas forcément le même :

$\forall x.E(x) \Rightarrow \exists y.C(y) \wedge A(x,y)$

Les enfants aiment un chien, le même pour tous les enfants.

$\exists y.C(y) \wedge \forall x.E(x) \Rightarrow A(x, y)$