INFO501 : logique (et informatique)
TD : logique du premier ordre

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 17, poste : 94 22
email : Pierre.Hyvernat@univ-smb.fr
www : http://www.lama.univ-smb.fr/~hyvernat/

Exercice 1 : formules

Question 1. Donnez une formules du premier ordre (en introduisant si besoin des fonctions ou des relations) pour les phrases suivantes.

N'oubliez pas de typer explicitement vos quantificateurs si vous estimez que c'est nécessaire.
- "Tous les oiseaux ne volent pas."
- "Tout nombre entier peut s'écrire comme la somme de 4 carrés."
- "Tous les nombres premiers sont impairs, sauf 2."
- "Toute équation linéaire à une inconnue (càd de la forme $ax + b = 0$) a une solution."

Question 2. Donnez l'arbre des formules suivantes. Identifiez les variables liées (et leurs quantificateurs) ainsi que les variables libres.

$$\forall x, \big(P(x) \Rightarrow Q(y)\big) \qquad \forall x, \big(P(x) \Rightarrow Q(x)\big)$$
$$\exists x, P(x) \Rightarrow Q(x) \qquad \big(\exists x, P(x) \wedge \forall x, Q(x) \Rightarrow \exists x, R(x)\big) \wedge S(x)$$

Renommez les variables liées pour éviter que des variables différentes portent le même nom.

Question 3. Faites les substitutions suivantes
- $f(x) = 0$, avec $x := f(y)$
- $\exists x, f(x) = y$, avec $y := f(x)$
- $t > 0 \Rightarrow \exists n, \forall s, \pi_t(s) > n$, avec $n := t + 1$
- $t > 0 \Rightarrow \forall s, \pi_t(s) > n$, avec $n := t + 1$
- $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$, avec $n := i + j$

Question 4. Donnez une formule arithmétique pour le prédicat "être impair" et une autre pour le prédicat "être premier". (Vous n'avez droit qu'aux constantes entières, aux fonctions $+$ et $\times$, et à l'égalité.)

Question 5. Donnez une formule du premier ordre pour exprimer : "`m` est l'élément minimal parmi `T[0]`, `T[1]`, ... `T[l]`",

Remarque, `m`, `l` et `T` doivent être des variables libres.

Question 6. On "restreint" parfois les quantificateurs avec un "tels que" :
- $\forall \varepsilon$ t.q. $\varepsilon > 0, F(\varepsilon)$
- $\exists p$ t.q. $p > n, F(p)$

Précisez le sens de l'abréviation "t.q." dans ces deux formules en donnant des formules correspondantes n'utilisant que les quantificateurs et connecteurs usuels.

Question 7. On utilise parfois le quantificateur $\exists! x, P(x)$ pour "il existe un *unique* x qui vérifie $P(x)$. Comment peut-on exprimer ce quantificateur en logique du premier ordre ?

Exercice 2 : adapté de l'examen 2023

On aimerait spécifier la fonction `sorted` de Python, qui prend en argument un tableau et en renvoie une version triée.

Il y a 2 types d'objets : `tableau` et `entier`. En outre, le langage contient :
- les constantes entières (`0`, `1`, `12743`, etc.),
- la fonction d'accès à un élément d'un tableau (`T[i]`), la fonction `len`, la fonction `sorted`,
- les relation "<", "≤" et l'égalité (`<`, `<=` et `==` en Python).

Donnez, dans ce langage, les formules pour :
- "Le tableau `t` est trié." (variable libre : `t`)
- "Le tableau `sorted(t)` est trié." (variable libre : `t`)
- "Le résultat de `sorted` est toujours trié." (formule close)

[3] *Question 1.* Le fait que `sorted` renvoie un tableau trié n'est pas suffisant pour spécifier `sorted` : il faut ajouter le fait que le résultat est une permutation de l'argument.
- Donnez une formule pour "le résultat de `sorted(t)` contient exactement les mêmes éléments que `t`. (variable libre : `t`)
- Expliquez pour quoi cette formule n'est pas suffisante pour exprimer que `sorted(t)` est une permutation de `t`.
- Proposez une solution en ajoutant des symboles de fonctions dans le langage, par exemple pour les méthodes usuelles sur les tableaux (`T.index`, `T.count`, etc.)

Exercice 3 : validité, modèles

Question 1. Donnez une formule non typée, sans symboles de fonction, constantes ou relation (à part "=") qui est vraie uniquement dans les modèles avec au moins 3 éléments distincts.

Déduisez en une formule qui est vraie uniquement dans les modèles avec au plus 2 éléments.

Idem pour "au plus 3", "au plus 4", etc.

Cherchez une formule du premier ordre qui est vraie uniquement dans les modèles finis.

Question 2.
- Donnez une formule non typée vraie sur les entiers naturels et fausse sur les entiers relatifs, en utilisant le langage $(0, \leq)$,
- Donnez une formule non typée vraie sur les entiers naturels et fausse sur les entiers relatifs, en utilisant le langage $(\leq)$,
- donnez une formule non typée vraie sur les entiers relatifs et fausse sur les rationnels, en utilisant le langage $(0, \leq)$,
- donnez une formule non typée vraie sur les rationnels et fausse sur les réels, en utilisant le langage $(0, 1, +, *)$,
- donnez une formule non typée vraie sur les rationnels et fausse sur les réels, en utilisant le langage $(0, \leq)$.

Question 3. On considère le langage avec une seule relation binaire R. La formule atomique $R(x, y)$ est interprétée comme "les points x et y sont reliés par une arête".

On s'intéresse aux 2 modèles suivants :

- premier modèle :
- second modèle : 

Donnez une formule qui est vraie dans un des deux modèles suivants, et fausse dans l'autre.

Est-ce que la formule $\forall x, \forall y, \forall z, R(x, y) \wedge R(y, z) \Rightarrow R(x, z)$ conviendrait ?

Question 4. Que pensez vous de la modélisation suivante pour la phrase "Richard a deux frères : John et Geoffrey" :

$$\texttt{frere}(\texttt{Richard}, \texttt{John}) \wedge \texttt{frere}(\texttt{Richard}, \texttt{Geoffrey})$$

où :
- `frere` est une relation binaire (qui satisfait des axiomes non spécifiés),
- `Richard`, `John` et `Geoffrey` sont des constantes introduite pour l'occasion.

[!] *Question 5.* Modélisez la phrase suivante "Dans la promo de L3, il y a un étudiant qui, s'il réussit le cours de logique, tous les étudiants réussissent."

Est-ce que la formule correspondante est valide ?

[!] *Question 6.* Que pensez vous de la formule $\big(\forall x, F(x)\big) \Rightarrow \big(\exists x, F(x)\big)$?

Exercice 4 : déduction naturelle

Rappels : voici les règles de la déduction naturelle pour la logique du premier ordre.

$$\frac{\Gamma \ \vdash \ F(x)}{\Gamma \ \vdash \ \forall x, F(x)} \ (\forall_c)^{(i)} \qquad \frac{\Gamma; \forall x, F(x); F(u) \ \vdash \ C}{\Gamma; \forall x, F(x) \ \vdash \ C} \ (\forall_h) \ / \ \text{(spécialisation)}^{(ii)}$$

$$\frac{\Gamma \ \vdash \ F(u)}{\Gamma \ \vdash \ \exists x, F(x)} \ (\exists_c\dagger) \qquad \frac{\Gamma; F(x_0) \ \vdash \ C}{\Gamma; \exists x, F(x) \ \vdash \ C} \ (\exists_h) \ / \ \text{(utilisation du } \exists)^{(i)}$$

avec
- dans la règle $(\forall_c)$, x est une variable *qui n'est pas libre dans le séquent*,
- dans les règle (spécialisation) et $(\exists_c)$, u est un terme arbitraire,
- dans la règle (utilisation du $\exists$), x_0 est une variable *qui n'est pas libre dans le séquent.*

Question 1. Faites une preuve de la formule

$$\Big(\forall x, \big(P(x) \wedge Q(x)\big)\Big) \quad \Rightarrow \quad \Big(\big(\forall x, P(x)\big) \wedge \big(\forall x, Q(x)\big)\Big)$$

puis de l'implication inverse.

Question 2. Plus difficile, faites une preuve de la formule

$$\Big(\big(\exists x, P(x)\big) \vee \big(\exists x, Q(x)\big)\Big) \quad \Rightarrow \quad \Big(\exists x, \big(P(x) \vee Q(x)\big)\Big)$$

puis de l'implication inverse.

Question 3. Démontrez l'implication suivante : $\big(\exists x, \neg\varphi(x)\big) \Rightarrow \neg\big(\forall x, \varphi(x)\big)$

Question 4. Nettement plus difficile, montrez l'implication inverse : $\neg\big(\forall x, \varphi(x)\big) \Rightarrow \big(\exists x, \neg\varphi(x)\big)$

Exercice 5 : unification, programmation logique

Question 1. Appliquez l'algorithme vu en cours pour unifier les expressions suivantes.
- $f(g(x), y)$ et $f(z, y)$
- $g(x, f(y))$ et $g(f(y), g(z))$
- $f(g(x, y))$ et $f(g(a, f(x)))$
- $f(g(x, y))$ et $f(y)$
- $g(x, f(y))$ et $g(f(z), h(z))$
- $f(g(x, f(x)), f(z))$ et $f(z, f(y))$

Question 2. Appliquez l'algorithme de résolution de Prolog (SLD) sur les exemples suivants

```
%%% exemple 1
aime(juliette, pizza).                          %(1)
aime(romeo, Truc) :- aime(juliette, Truc).      %(2)
-----------------------------------------
?- aime(romeo, pizza).

%%% exemple 2
aime(romeo, anchois).                                          %(1)
aime(juliette, pizza(anchois)).                                %(2)
aime(romeo, Truc) :- nourriture(Truc), aime(juliette, Truc).   %(3)
nourriture(pizza(X)) :- nourriture(X).                         %(4)
nourriture(X) :- aime(romeo, X).                               %(5)
-----------------------------------------
?- aime(romeo, pizza(anchois)).

%%% exemple 3
% mêmes règles que l'exemple 2
-----------------------------------------
 ?- aime(romeo, pizza(crevettes)).
```

Question 3. Que pensez vous de la modélisation suivante pour la phrase "Richard a deux frères : John et Geoffrey" en Prolog

```
frere(richard, john).
frere(richard, geoffrey).
```

où :

- `frere` est une relation binaire (qui satisfait des axiomes non spécifiés),
- `richard`, `john` et `geoffrey` sont des constantes introduite pour l'occasion.