INFO501 : logique (et informatique)
Examen
AVEC SOLUTIONS

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/`

Durée : 1h30.
Documents et calculatrices interdits.
Un barème provisoire est donné dans la marge.
1 point négatif sera réservé à la présentation de vos réponses...

Partie 1 : calcul propositionel

(3) *Question 1.*

- Rappelez la table de vérité de l'implication "$\Rightarrow$".

A	B	$A \Rightarrow B$
0	0	?
0	1	?
1	0	?
1	1	?

- La formule $\big((A \Rightarrow B) \Rightarrow A\big) \Rightarrow A$ est elle une tautologie ?
- Cette même formule est elle démontrable avec les règles de démonstration usuelles ?

Éléments de réponse :

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

La formule $\big((A \Rightarrow B) \Rightarrow A\big) \Rightarrow A$ est une tautologie. Sa table de vérité est

A	B	$A \Rightarrow B$	$(A \Rightarrow B) \Rightarrow A$	$\big((A \Rightarrow B) \Rightarrow A\big) \Rightarrow A$
0	0	1	0	1
0	1	1	0	1
1	0	0	1	1
1	1	1	1	1

Le théorème de complétude implique que cette formule est également prouvable.

(3) *Question 2.* Faites la preuve de la formule $((A \vee B) \wedge \neg B) \Rightarrow A$ en utilisant les règles usuelles données à la fin du sujet.

Note : vous devrez utilisez la règle du "raisonnement par l'absurde".

Éléments de réponse :

$$
\dfrac{\dfrac{\dfrac{\dfrac{\text{axiome}}{A;\neg B \ \vdash\ A} \qquad \dfrac{\dfrac{\dfrac{\text{axiome}}{B;\neg B;\neg A \ \vdash\ \neg B} \quad \dfrac{\text{axiome}}{B;\neg B;\neg A \ \vdash\ B}}{B;\neg B;\neg A \ \vdash\ \bot}\,\text{(contradiction)}}{B;\neg B \ \vdash\ A}\,\text{(raisonnement par l'absurde)}}{(A\vee B);\neg B \ \vdash\ A}\,(\vee_h)\ /\ \text{(raisonnement par cas)}}{(A\vee B)\wedge\neg B \ \vdash\ A}\,(\wedge_h)}{\vdash\ ((A\vee B)\wedge\neg B)\Rightarrow A}\,(\Rightarrow_c)
$$

(3) *Question 3.* Appliquez, en détaillant ce que vous faites, l'algorithme DPLL vu en cours en utilisant uniquement la simplification des clauses unitaires sur la formule en FNC suivante.

$$(x_3 \vee x_1) \wedge (x_2 \vee x_1) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_5) \wedge (\neg x_5 \vee \neg x_4) \wedge (x_5 \vee \neg x_4)$$

Si aucune simplification n'est possible, prenez le premier littéral de la première clause, et rendez-le vrai. (Autrement dit, si le premier littéral est $\neg x_2$, commencez par utiliser $x_2 =$ `false`.)

Éléments de réponse : La formule est satisfiable, et la trace de l'algorithme est la suivante : (L'entier `3` représente le littéral x_3, et l'entier `-3` le littéral $\neg x_3$)

```
formule         : [ [3, 1], [2, 1], [-3, 4], [-2, -5], [-5, -4], [5, -4] ]
choix littéral : 3 ==> [ [2, 1], [4], [-2, -5], [-5, -4], [5, -4] ]
clause unitaire: 4 ==> [ [2, 1], [-2, -5], [-5], [5] ]
clause unitaire: -5 ==> [ [2, 1], [] ]
    !!! la formule contient la clause vide : FAUX !!!
backtrack       : -3 ==> [ [1], [2, 1], [-2, -5], [-5, -4], [5, -4] ]
clause unitaire: 1 ==> [ [-2, -5], [-5, -4], [5, -4] ]
choix littéral : -2 ==> [ [-5, -4], [5, -4] ]
choix littéral : -5 ==> [ [-4] ]
clause unitaire: -4 ==> [  ]
SATISFIABLE
-3 1 -2 -5 -4
```

(1) *Question 4.* Si $(l_1 \vee l_2 \vee l_3 \vee l_4)$ est une clause de 4 littéraux dans une formule en forme normale conjonctive, on peut la remplacer par 2 clauses de 3 littéraux en introduisant une *nouvelle* variable z :

$$(l_1 \vee l_2 \vee z) \wedge (\neg z \vee l_3 \vee l_4)$$

Expliquez pourquoi cette transformation ne modifie pas la satisfiabilité de la formule initiale.

Éléments de réponse : Si la formule initiale est satisfiable, on peut trouver des valeurs pour les variables qui la rendent vraie. En particulier, au moins un littéral parmi l_1, l_2, l_3 ou l_4 doit être rendu vrai.

- Si l_1 ou l_2 est rendu vrai, on choisi la valeur $z =$ `false`. Ceci permet de rendre vraie la nouvelle clause "$(\neg z \vee l_3 \vee l_4)$". (La clause "$(l_1 \vee l_2 \vee z)$" reste vraie car un des littéraux l_1 ou l_2 est vrai par hypothèse.)
- Si l_3 ou l_4 est rendu vrai, on choisi la valeur $z =$ `true`. Ceci permet de rendre vraie la nouvelle clause "$(l_1 \vee l_2 \vee z)$". (La clause "$(\neg z \vee l_3 \vee l_4)$" reste vraie car un des littéraux l_3 ou l_4 est vrai par hypothèse.)

Réciproquement, si la nouvelle formule est rendue vraie, au moins un des littéraux de "$(l_1 \vee l_2 \vee z)$" et un des littéraux de "$(\neg z \vee l_3 \vee l_4)$" sont rendus vrais. Si l_1, l_2, l_3 et l_4 étaient tous faux, il faudrait que z et $\neg z$ soient tous les 2 vrais ! C'est impossible ; il y a donc forcément au moins un des littéraux l_1, l_2, l_3 ou l_4 qui est vrai. Les valeurs choisies pour rendre la nouvelle formule vraie rendent également la formule initiale vraie ! (Sans utiliser la valeur de la nouvelle variable z.)

Partie 2 : logique du premier ordre

(2) *Question 1.* Un tableau est dit “unimodal” s’il commence par des éléments triés dans l’ordre croissant, suivis d’éléments triés dans l’ordre décroissant.

Par exemple, `[1,2,3,3,4,5,5,3,2,0]` est unimodal, mais `[1,2,1,2]` ne l’est pas. Chacune des parties (croissante ou décroissante) peut être vide. Ainsi, le tableau vide ou les tableaux triés sont unimodaux.

Donnez une formule $F($`T`$)$ qui sera vraie exactement lorsque T est un tableau unimodal.

Notes :

- `T` est une variable libre
- vous pouvez utiliser l’accès aux éléments d’un tableau “`T[j]`”, la fonction “`len`”, les relations “`==`”, “`<`”, “`>`”, “`<=`” et “`>=`” et les opérations arithmétiques usuelles.

Éléments de réponse : On peut par exemple prendre

$$F(\texttt{T}) \quad := \quad (\texttt{len(T)==0}) \ \lor \ \Big(\exists i, \big(\forall k, k\texttt{<}i \Rightarrow \texttt{T[}k\texttt{]<=T[}k\texttt{+1]}\big) \land \big(\forall k, i\texttt{<}k \Rightarrow \texttt{T[}k\texttt{-1]>=T[}k\texttt{]}\big)\Big)$$

Pour être précis, il faudrait également ajouter des bornes sur les quantifications :

- “$\underline{\texttt{0<=}i \land i\texttt{<len(T)}} \land \ldots$” après la quantification $\exists i$,
- “$\underline{\texttt{0<=}k} \land k\texttt{<}i \Rightarrow \ldots$” après la première quantification $\forall k$,
- “$\texttt{i<}k \land \underline{k\texttt{<len(T)}} \Rightarrow \ldots$” après la seconde quantification $\forall k$,

(2) *Question 2.* La tactique “`induction n`” de Coq permet de faire une preuve par induction sur la variable `n`. Pour les entiers :

- on doit prouver le cas de base ($\texttt{n} = 0$)
- on doit prouver l’étape d’induction : si le but est vrai pour `n`, alors il est vrai pour `n+1`.

Donnez une règle que l’on pourrait ajouter à celles données en fin de sujet pour cette méthode de démonstration :

$$\frac{??? \quad ??? \quad ???}{\Gamma \ \vdash \ F(n)} \text{(induction)}$$

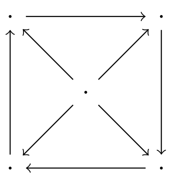
Éléments de réponse : On peut prendre

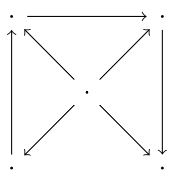
$$\frac{\Gamma \ \vdash \ F(0) \qquad \Gamma; F(n) \ \vdash \ F(n+1)}{\Gamma \ \vdash \ F(n)} \text{(induction)}$$

(2) *Question 3.* On s’intéresse au langage contenant une unique relation binaire R que l’on interprète par “$R(x, y)$ signifie qu’il y a une flèche de x vers y”.

Donnez une formule qui sera vraie dans le premier graphe, et fausse dans le second.

N’oubliez pas de justifier !





Éléments de réponse : On peut par exemple prendre

$$\forall x, \exists y, R(x, y)$$

pour dire qu’il y a une flèche sortante de tous les sommets. C’est vrai pour le premier graphe, mais pas pour le second.

Une autre différence est que le premier graphe a un cycle de 4 flèches consécutives, mais pas le second. La formule est plus complexe :

$$\exists x, \exists y, \exists z, \exists t, R(x, y) \land R(y, z) \land R(z, t) \land R(t, x)$$

Partie 3 : Prolog et unification

(2) *Question 1.* Détaillez l'algorithme d'unification vu en cours sur les termes suivants :
- `f(Y, g(Z))` et `f(g(X), Y)` ;
- `f(X, f(X,Y,Z), g(Y))` et `f(g(Z), f(X, Z, Z), X)`.

Éléments de réponse : Les 2 premiers termes sont unifiables :

```
    f(Y, g(Z)) = f(g(X), Y)
--> Y = g(X) ; g(Z) = Y
--> g(Z) = G(X)                          [ Y:=g(X) ]
--> Z = X                                [ Y:=g(X) ]
--> ''                                   [ Y:=g(X) ; Z:=X ]
```

Note : dans le passage de la deuxième à la troisième ligne, il ne faut pas oublier d'appliquer la substitution "`y :=g(X)`" aux équations restantes. La substitution finale est donc "`[ Y := g(X) ; Z := X ]`".

Pour les 2 termes suivants, on obtient :

```
    f(X, f(X,Y,Z), g(Y)) = f(g(Z), f(X, Z, Z), X)
--> X = g(Z) ; f(X,Y,Z) = f(X,Z,Z) ; g(Y) = X
--> f(g(Z),Y,Z) = f(g(Z),Z,Z) ; g(Y) = g(Z)          [ X:=g(Z) ]
--> g(Z) = g(Z) ; Y = Z ; Z = Z ; g(Y) = g(Z)        [ X:=g(Z) ]
--> Y = Z ; Z = Z ; g(Y) = g(Z)                      [ X:=g(Z) ]
--> Z = Z ; g(Z) = g(Z)                              [ X:=g(Z) ; Y:=Z ]
--> g(Z) = g(Z)                                      [ X:=g(Z) ; Y:=Z ]
--> Z = Z                                            [ X:=g(Z) ; Y:=Z ]
--> ''                                               [ X:=g(Z) ; Y:=Z ]
```

Les termes sont donc unifiables, et la substitution finale est "`[ X := g(Z) ; Y := Z ]`".

(2) *Question 2.* Écrivez en Prolog, un prédicat ternaire `supprime/3`.

On veut que `supprime(X, L1, L2)` soit vrai lorsque la liste `L2` est égale à la liste `L1` dans laquelle on a supprimé toutes les occurences de `X`.

On aura par exemple `supprime(3, [1,3,2,4,3], [1,2,4])`.

Remarques :
- le prédicat `supprime/3` prédicat est récursif,
- la syntaxe `[E | L]` de Prolog désigne une liste qui commence par l'élément `E`, et où `L` contient tous les éléments suivants (éventuellement aucun).

Éléments de réponse :

```
supprime(X, [], []) :- supprime(X, L1, L2).
supprime(X, [X | L1], L2) :- supprime(X, L1, L2).
supprime(X, [Y | L1], [Y | L2]) :- X \= Y, supprime(X, L1, L2).
```

Rappels : voici les règles de la déduction naturelle pour la logique propositionnelle.

$$\frac{\Gamma; F \ \vdash \ G}{\Gamma \ \vdash \ F \Rightarrow G} \ (\Rightarrow_c) \qquad \frac{\Gamma; F \Rightarrow G \ \vdash \ F}{\Gamma; F \Rightarrow G \ \vdash \ G} \ (\Rightarrow_h) \ / \ \text{(modus ponens)}$$

$$\frac{\Gamma \ \vdash \ F_1 \qquad \Gamma \ \vdash \ F_2}{\Gamma \ \vdash \ F_1 \wedge F_2} \ (\wedge_c) \qquad \frac{\Gamma; F_1; F_2 \ \vdash \ G}{\Gamma; F_1 \wedge F_2 \ \vdash \ G} \ (\wedge_h)$$

$$\frac{\Gamma \ \vdash \ F_i}{\Gamma \ \vdash \ F_1 \vee F_2} \ (\vee_{c,i}) \qquad \frac{\Gamma; F_1 \ \vdash \ G \qquad \Gamma; F_2 \ \vdash \ G}{\Gamma; F_1 \vee F_2 \ \vdash \ G} \ (\vee_h) \ / \ \text{(raisonnement par cas)}$$

$$\frac{\Gamma; F \ \vdash \ \bot}{\Gamma \ \vdash \ \neg F} \ (\neg_c) \qquad \frac{\Gamma \ \vdash \ \neg F \qquad \Gamma \ \vdash \ F}{\Gamma \ \vdash \ \bot} \ \text{(contradiction)}$$

$$\frac{}{\Gamma; F \ \vdash \ F} \ \text{(axiome)} \qquad \frac{\Gamma; \neg F \ \vdash \ \bot}{\Gamma \ \vdash \ F} \ \text{(raisonnement par l'absurde)}$$

$$\frac{\Gamma \ \vdash \ G \qquad \Gamma; G \ \vdash \ F}{\Gamma \ \vdash \ F} \ \text{(résultat intermédiaire)}$$

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

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

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

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.*