

**Épreuve du 20 janvier 2004, 14-17 heures.**

Ce sujet comporte 2 pages.

Tous documents autorisés.

Le barème est indicatif.

Toutes les réponses doivent être justifiées (de façon concise).

**Partie I - arbres de recherche et unification (7 pts)**

**Question 1 (3 pts)**

Soit le programme

$\text{not}(P) :- P, !, \text{fail} .$

$\text{not}(\_P) .$

Calculer les arbres de recherche des buts suivants :

$\text{not}(X = 1), X = 2$

et

$\text{not}(\text{not}(X = 1)), X = 2 .$

Indiquer les témoins des preuves si il y en a. Indiquer clairement pour chaque coupure les branches qu'elle coupe.

**Question 2 (2 pts)**

Qu'en conclure ?

Soit le prédicat  $p/2$  suivant :

$p(t([X|Y], Z, U), t(X, Y, [Z|U])) .$

et le but

$:- p(T1, t([], 1, [2, 3, 4])) ,$

$p(T2, T1) ,$

$p(T3, T2) ,$

$p(T3, T4) .$

**Question 3 (1 pt)**

Indiquer les témoins de la preuve de ce but.

**Question 4 (1 pt)**

En donner une représentation graphique.

**Partie II - Logique (6 pts)****Question 4 (3 pts)**

Le séquent  $\exists x.A(x)$  ,  $\forall x.(A(x) \Rightarrow B(x)) \vdash \exists x.B(x)$  a-t-il une preuve dans le calcul des séquents ?

**Question 5 (3 pts)**

A-t-il une preuve uniforme ?

**Partie III - Programmation logique (7 pts)**

On souhaite programmer le predicat `accepte/2` qui détermine si un automate accepte un mot. On suppose que l'automate est codé comme dans les travaux pratiques 5 et 6, et que le mot est codé par une liste d'éléments d'un vocabulaire fini donné. On suppose le vocabulaire totalement ordonné ; c-à-d. tout les éléments du vocabulaire ont un rang fixé.

**Rappel** : dans la représentation des travaux pratiques 5 et 6, un successeur inconnu signifie l'absence du successeur.

Utiliser les prédicats prédéfinis suivants :

1. `var/1` détermine si un terme est une inconnue ;
2. `n_ieme/3` détermine la valeur du  $n$ -ième élément d'une liste correspondant à l'élément de vocabulaire de rang  $n$ .

`% var( ?Terme )` réussit si et seulement si `Terme` est une inconnue

`% n_ieme( +Lettre, ?Liste, ?Element )` réussit si `Lettre` est l'élément de vocabulaire de rang  $n$  et si `Element` est le  $n$ -ième élément de `Liste`.

**Question 6 (2 pts)**

Donner les principes du codage de `accepte/2` .

**Question 7 (5 pts)**

Programmer `accepte/2` .