Examen Final
Mercredi 13 Janvier 2010, de 8h à 10h

Coefficient : 40%.
Aucun document autorisé.

Remarques et conseils :
- Lisez **attentivement** chaque question avant d'y répondre.
- Indiquer clairement sur votre copie le numéro de l'exercice avant d'y répondre.
- **Les exercices 1 à 3 seront traités sur une copie et l'exercice 4 sur une copie séparée.**
- Lorsque vous définissez un prédicat, son profil et sa définition formelle doivent être au moins indiqués. Un jeu d'essais est facultatif.
- Expliquez autant que possible vos choix lors de la définition d'un prédicat ou d'une fonction.
- Le barème défini ci-après est susceptible d'être modifié.

Exercice 1. (2 points) **Unification**

Unifier les termes suivants, en sachant que les variables sont en minuscules et les constantes en majuscules :

p(f(z), f(x), f(f(x))) et p(f(f(y)), y, f(y))

h(g(x), f (A, y), z) et h(y, z, f (u, x))

Exercice 2. (4 points) **Système formel**

Soit le système formel : <V, L, A, R>

Vocabulaire : V={a, b, c,d}

Langage L ensemble des mots constructibles à partir de V

Axiome A = { ├ B}

Ensemble R des règles d'inférence :

- B → a C b
- C → c | d | e | B

1) Donnez 3 mots significatifs de longueurs différentes du langage L.

2) Quelle est la forme générale des mots de L?

3) Écrire une solution Prolog, basée sur les règles d'inférence, qui reconnaît syntaxiquement les mots du langage L.

Exercice 3. (5 points) **Ne ratez pas les parties en Lisp**.

Écrire en Lisp une fonction permettant de donner la liste des parties d'un ensemble.

exemple : (***parties*** '(1 2)) → ((1 2) (1) (2) ())

1) Donnez le résultat de (***parties*** '(1 2 3)) et de (***parties*** ())

2) Écrivez la fonction ***augmente*** , de profil : Exp × List-list → List-list: ***(augmente e L)*** rend la liste issue de L où chaque sous-liste de L a été augmentée avec l'objet e

exemple : (***augmente*** 'a '((b) (c))) → ((a b) (a c))

3) Écrivez la fonction ***concat*** de profil : List × List → List qui concatène 2 listes (cette fonction s'appelle ***append*** en Lisp).

exemple : (***concat*** '(a b) '(c d e)) → (a b c d e)

4) En déduire la fonction ***parties*** de profil : List → List-list qui génère la liste des parties de l'ensemble passé en paramètre.

<u>*Notes*</u>: seules les fonctions prédéfinies cons, car, cdr, if et null seront utilisées.

Les listes (1 2) et (2 1) sont une même représentation de l'ensemble {1, 2}

Exercice 4. (9 points) **Tant va la cruche à l'eau ...**

Disposant de deux cruches de contenances respectives 3 litres et 7 litres, initialement vides, comment obtenir 8 litres d'eau, sachant que l'on peut remplir ou vider chacune des cruches, ou transvaser le contenu (ou une partie du contenu) de l'une dans l'autre pour vider l'une (respectivement remplir l'autre).

Remarque :
Les différentes parties sont indépendantes, mais la réflexion pour chacune d'elles peut aider à résoudre les autres.

1) Résolution graphique

On créera un graphe dont les sommets sont toutes les situations possibles, c'est-à-dire les couples (x,y) des contenances respectives des deux cruches, et les arcs les mouvements permettant de passer d'une situation à une autre.
En utilisant la représentation suivante : la situation *(x,y)* est figurée dans le plan par le point de coordonnées *x* et *y,* donnez le graphe complet en décrivant géométriquement les mouvements possibles à partir d'une situation quelconque *(x,y)*.
Enfin tracer le chemin de la solution optimale.

2) Parcours en largeur

Donnez le chemin obtenu en appliquant une stratégie de recherche basée sur un parcours en largeur.
Vous indiquerez toutes les branches explorées, qu'elles fassent partie de la solution ou non.

3) Parcours en profondeur
Même question que 2) avec un parcours en profondeur d'abord.

4) Algorithme A*
En choisissant comme heuristique la fonction h suivante :

soit Tot = Contenu(1)+Contenu(2) :

si Tot = 8 h = 0

sinon h = min (distance (5, Tot), distance (1, Tot)),

appliquez l'algorithme A*. A chaque étape vous donnerez les états des listes Open et Closed triées selon f.