Université Grenoble Alpes
MIAGE L3 – Programmations déclaratives

Examen Session 1
Avril 2024

Durée : 90 minutes. 1 document personnel manuscrit A4 autorisé. Sujet sur 1 recto-verso.

I. Sur la base du projet Pile ou Face menteur – en ProLog (barème indicatif : 6 points)

Rappel du projet, les stratégies du jeu Pile ou Face menteur sont des prédicats ayant le profil suivant :

```
%% nomStratégie(en entrée : 1 entier, 1 liste, 1 annonce ; en sortie : 1 réponse)
%%  rem. la liste des parties jouées est l'historique des parties [piece, annonce, réponse]
%%  ex. pour 3 parties : [[pile,pile,pile],[pile,face, pile],[face,face,pile]]
```

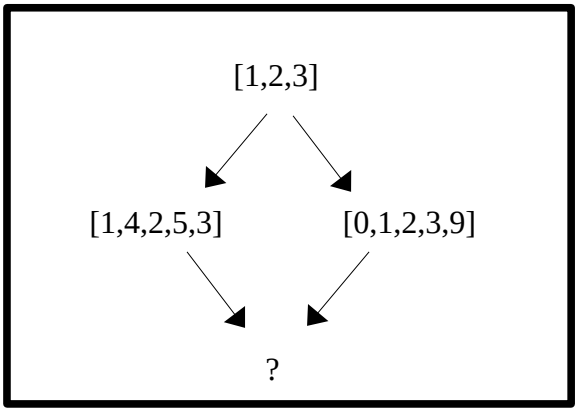
%% pour faciliter l'accès, la dernière partie est au début

Q1.1 Spécifiez et réalisez une stratégie qui alterne l'hypothèse d'un mensonge et d'une annonce vraie pour le menteur : si la dernière annonce du menteur était un mensonge, le mentaliste suppose que l'annonce courante du menteur sera vrai ; et si la dernière annonce du menteur était vraie, le mentaliste suppose que l'annonce courante sera un mensonge.

Q I.2 Spécifiez et réalisez une stratégie qui analyse l'ensemble des parties ayant la même annonce que l'annonce courante et regarde si cette annonce est plus souvent un mensonge ou une annonce vraie.

II. Fusion d'ajouts - en ProLog ou en Erlang (barème indicatif : 7 points)

Au choix, vous réaliserez cet exercice en ProLog ou en Erlang.
Avant de répondre à la première question, lire l'ensemble de l'exercice et penser à une solution.



Q II.1 Choix et argumentation : Précisez et argumentez votre choix pour ProLog ou Erlang pour résoudre le probème courant.
Quels sont les raison de votre choix pour Prolog ou Erlang, ces raisons sont-elles liées à la résolution du problème courant ou liées aux langages ?
Argumentez succinctement.

Trois listes d'entiers sont fournies,
par exemple les listes : [1, 2, 3], [1, *4*, 2, *5,* 3] et [**0**, 1, 2, 3, **9**].
La première liste est une liste initiale à partir de laquelle les 2 autres listes ont été obtenues chacune par un programmeur différent, chaque programmeur ayant ajouté des entiers dans la liste. Le premier programmeur a ajouté *4* et *5* à la liste initiale, le second programmeur a ajouté **0** et **9** à la liste initiale. Vous supposerez que les modifications apportées sont compatibles entre elles, chaque programmeur a fait des ajouts « dans son coin » : entre deux entiers de la liste initiale, il ne peut y avoir eu qu'une série d'ajouts par un programmeur ou par l'autre, mais pas par les deux en même temps. Les trois listes sont donc toutes identiques localement, s'il n'y a pas eu d'ajout, ou deux au moins sont identiques, (la liste initiale et une liste non-modifiée), ce qui permet de voir les ajouts qui ont été faits.

Q II.2 Fusion d'ajouts cohérents. Les ajouts étant supposés compatibles, spécifier et programmer une solution pour fusionner tous les ajouts et fournir la liste finale comprenant les modifications apportées par les 2 programmeurs. Dans l'exemple proposé, la liste finale serait [0, 1, 4, 2, 5, 3, 9]

III. Réduction de liste de valeurs (barème indicatif : 7 points)

Dans cet exercice, l'objectif est de réduire une liste de valeurs (par ex. des températures atmosphériques prises toutes les minutes) pour pouvoir les analyser et les afficher sans avoir trop de données à analyser et/ou afficher. L'exercice ne porte que sur l'algorithme de réduction de la liste des valeurs pas sur l'analyse ou l'affichage.

L'algorithme de réduction de la liste est le suivant :

- si l'ensemble des valeurs de la liste sont proches de la valeur moyenne de la liste (par exemple proche à ± 2 par rapport à la moyenne), la liste est réduite à une seule valeur : la valeur moyenne de la liste.
- Sinon, la liste est coupée en 2 (approximativement début/fin, selon que la liste est de longueur paire ou impaire, le découpage tombe juste ou presque juste) et l'algorithme de réduction est appliqué à chaque 1/2 liste récursivement (jusqu'à ce que les sous-listes soient réduites à 1 élément par réduction à la moyenne, ou par découpage moitié-moitié début-fin)

Q III.1 Calcul naïf séquentielle : Spécifier et programmer en Erlang une fonction réalisant ce calcul.
Q III.2 Parallélisation sur 2 cœurs : Paralléliser le calcul précédent pour l'exécuter sur une machine ayant 2 cœurs, par exemple en répartissant les calculs sur les valeurs de début sur un cœur, et sur l'autre cœur pour les valeurs de fin -s'il y a besoin de couper la liste en 2, ce qui est le cas général (pour une autre parallélisation, expliquer le principe de votre algorithme).