

Ce sujet comporte 2 pages. Tous les documents de cours, TD et TP sont autorisés. Le barème est indicatif.

Justifier toutes les réponses de façon concise mais précise. Utiliser de façon cohérente les notations du cours. Ne pas répondre au hasard.

Partie I – La notion de réduction (5 points)

Question 1

Résumer en moins d'une page le principe du raisonnement par réduction et l'usage qui en est fait dans le cours.

Partie II – Sémantique du langage WHILE (5 points)

Contexte : on considère le langage UNTIL qui diffère du langage WHILE uniquement parce que la boucle WHILE est remplacée par la boucle UNTIL, `do I until C`, dont la sémantique intuitive est que la commande I est exécutée tant que la condition C est fausse.

Question 2

Donner la sémantique formelle de la boucle UNTIL dans le style de la sémantique formelle de la boucle WHILE donnée en cours. On utilisera la même convention qu'en cours concernant le codage de la valeur de vérité de C.

Question 3

Le langage UNTIL est-il plus expressif que le langage WHILE ? L'est-il autant ? L'est-il moins ?

Partie III – L'interpréteur du langage WHILE en Scheme (5 points)

Contexte : on souhaite modifier l'interpréteur programmé en travaux pratiques de telle sorte qu'il retourne le coût de calcul du programme exécuté, en plus du résultat de l'exécution évidemment.

Question 4

On adopte d'abord comme mesure de coût le nombre de fois où une opération (c-à-d. `cons`, `nil`, `hd`, `tl` ou `=?`) est évaluée. Indiquer les grandes lignes des modifications à apporter à l'interpréteur réalisé en travaux pratiques, et détailler celles qui concernent la fonction `EVAL`.

Question 5

On reprend l'interpréteur de la question 4, et on souhaite qu'il attribue des coûts différents à chaque opération. De quel sujet de travaux pratiques s'inspirer pour réaliser cette fonctionnalité ? Quelle modification apporter à l'interpréteur ? Ne pas la détailler.

Partie IV – Complexité (5 points)

Question 6

Supposons un programme `p` et une donnée `v`, pour lesquels l'interpréteur construit à la question 4 mesure un coût `c`. Appliquons à ce programme la transformation qui consiste à éliminer les expressions complexes ; appelons `p'` le programme résultant. Que penser du coût `c'` de l'exécution du programme `p'` pour la donnée `v` ?

Question 7

On adopte maintenant comme mesure de coût le nombre de fois où une affectation ($:=$) est exécutée. On appelle f_p la fonction qui associe à chaque donnée le coût de l'exécution de p pour cette donnée. On appelle $f_{p'}$ la fonction qui associe à chaque donnée le coût de l'exécution de p' (programme résultant de l'élimination des expressions complexes du programme p) pour cette donnée. Est-ce que $f_p = O(f_{p'})$? Est-ce que $f_{p'} = O(f_p)$?