Université BORDEAUX I

Année 1994-95
Session de février

Maîtrise d'informatique

Modèles et Techniques de Programmation (J.P.Braquelaire)

Durée: 2 heures
Tous documents autorisés

Première partie – 8 points –

On considère une suite de lettres de longueur non bornée que l'on veut analyser au moyen d'un automate. Après chaque succès ou chaque échec l'automate est réinitialisé avec affichage d'un message et la lecture du mot continue à partir de l'état initial. Considérons par exemple l'automate `X` défini par la table de transition suivante :

État	a	b
1	2	échec
2	échec	3
3	succès	3

On code cet automate au moyen d'une fonction `X` écrite en Scheme. Voici un exemple d'exécution de la fonction `X`.

```
> (X)
a
a
Echec de X
a
b
b
a
Succes de X
b
Echec de X
a
...
```

On implémente l'automate en codant chaque état par une fonction. Par exemple, l'état i de l'automate `X` s'écrira :

```
(define (X-i)
  (case (next-letter)
    ⋮
    (else (echec 'X X))))
```

La fonction `next-letter` retourne la lettre suivante. La partie figurée par des pointillés code au cas par cas les transitions de l'automate.

1) – 3 points – Écrire le code de l'automate `X` de l'exemple (on ne demande pas d'écrire la fonction `next-char`).

On veut maintenant faire exécuter deux automates `X` et `Y` sur le même mot d'entrée. Celui-ci étant de longueur non bornée il n'est pas possible de lancer séquentiellement d'abord `X` puis `Y`. La solution retenue consiste à adopter une programmation par passage de continuation (CPS) afin de faire s'exécuter les deux automates en parallèle. Le premier automate est appelé avec en paramètre l'état initial du second. Par exemple, si `Y` est l'automate reconnaissant le langage `ba*ba`, voici un exemple d'exécution de X et `Y` en parallèle :

```
> (X Y)              Echec de Y           Echec de X
a                    b                    b
Echec de Y           a                    Echec de X
b                    Succes de X          a
b                    a                    Succes de Y
b                    a                    ...
```

1) – 3 points – On suppose que la fonction `next-char` est redéfinie de manière à lire deux fois chaque caractère du mot d'entrée. Écrire le code de la nouvelle version de l'automate `X`.

2) – 1 point – Définir l'automate `N` tel que l'évaluation de l'expression `(X N)` soit équivalente à celle de l'expression `(X)` pour l'automate `X` de la première question.

3) – 1 point – Expliquer pourquoi cette solution est possible en Scheme mais n'est pas raisonnable en C.

Deuxième partie – 12 points –

1) – 4 points – Écrire en C ANSI le module `memoire` composé des fonctions d'interface

```
void *memoire_allouer(size_t)
void memoire_liberer(void *)
```

réalisant respectivement l'allocation d'une zone mémoire dont la taille est passée en paramètre et sa libération. Ce module utilisera les fonctions d'allocation dynamique de la bibliothèque standard et le mécanisme d'exception vu en T.D. (instructions

`DEFINE_EXCEPTION(`$\langle name \rangle$`)`
`USE_EXCEPTION(`$\langle name \rangle$`)`
`HANDLE(`$\langle exception \rangle$`, `$\langle expr_1 \rangle$`, `$\langle expr_1 \rangle$`);`
`RAISE(`$\langle exception \rangle$`);`

définies au moyen du préprocesseur).

2) – 2 points – On souhaite étendre ce module en permettant de choisir les fonctions d'allocation mémoire utilisées. Cette extension doit préserver la compatibilité ascendante et par conséquent ne pas modifier la partie existante de l'interface. Proposer une solution.

3) – 2 points – On considère un modèle basique de programmation par objets en C dans lequelle une méthode ***m*** d'une classe ***cl*** s'écrit ***cl_m*** et possède un receveur comme premier paramètre formel. Décrire la fonction d'instanciation ***cl***`_CREER` de la classe ***cl*** (on s'intéresse ici principalement au traitement des erreurs).

4) – 2 points – Écrire une instruction C ANSI effectuant l'instanciation d'un objet ***o*** de la classe ***cl*** au moyen de la fonction précédente, et affichant lorsque se produit un problème d'allocation, un message de la forme :

$\langle fichier \rangle$`:`$\langle ligne \rangle$`:`***cl***`: erreur instanciation.`

5) – 2 points – Écrire une pseudo-fonction C ANSI de la forme

```
INSTANCIER(objet,classe)
```

qui effectue le traitement décrit dans la question précédente.