Compilation
Maîtrise d'Informatique 2002-2003

Examen Partiel
5 novembre 2002

L'énoncé est composé de 2 pages. L'examen dure 3 heures. Les notes de cours et de TD manuscrites ainsi que les supports de cours distribués cette année sont les seuls documents autorisés.

1 Analyse syntaxique (11 pts)

On modélise par une grammaire G la reconnaissance des constantes entières et décimales. Les symboles terminaux de la grammaire sont : $\{\mathtt{d}, \mathtt{.}, \mathtt{-}, \sharp\}$. Le terminal `d` correspond aux chiffres de 0 à 9. Les symboles non-terminaux de la grammaire G sont : $\{S, C, E, I\}$. Les productions de la grammaire sont :

$$\boxed{\begin{array}{lll} S & := & C\sharp \\ C & := & E \\ C & := & \mathtt{-}E \\ E & := & I \\ E & := & I\,\mathtt{.}\,I \\ I & := & \mathtt{d} \\ I & := & I\ \mathtt{d} \end{array}}$$

1. Donner une dérivation gauche et une dérivation droite du mot $\mathtt{-dd.dd}\sharp$.
2. Donner les étapes de l'analyse ascendante puis de l'analyse descendante du mot $\mathtt{-dd.d}\sharp$
3. La grammaire proposée est-elle LL(1) ? justifier la réponse.
4. On ajoute le non-terminal I' et on remplace les productions :

   $$\boxed{\begin{array}{lll} I & := & I\ \mathtt{d} \\ I & := & \mathtt{d} \end{array}}$$

   par les productions :

   $$\boxed{\begin{array}{lll} I & := & \mathtt{d}\ I' \\ I' & := & \mathtt{d}\ I' \\ I' & := & \epsilon \end{array}}$$

   Soit G_1 la grammaire ainsi obtenue, la grammaire G_1 est-elle LL(1) ?
5. Modifier les productions :

   $$\boxed{\begin{array}{lll} E & := & I \\ E & := & I\,\mathtt{.}\,I \end{array}}$$

   afin de supprimer le préfixe commun dans les règles issues de E.

6. Soit G_2 la grammaire ainsi obtenue.
   (a) Calculer l'ensemble des premiers des non-terminaux S, C, E et I.
   (b) Calculer la table de transitions de la grammaire G_2 pour l'analyse LL(1).
   (c) La grammaire G_2 est-elle LL(1) ?

2 Analyse lexicale (9 pts)

Lire attentivement l'ensemble du sujet et respecter les spécifications !

On se propose de construire un outil d'analyse lexicale qui cherche à identifier les mots mal orthographiés et les adresses électroniques. Cet outil peut être utilisé dans un éditeur pour souligner les mots incorrects et insérer automatiquement des liens sur les adresses électroniques. On cherche à éviter le soulignement de trop de chaînes de caractères, aussi l'outil ignorera les chaînes comportant des symboles spéciaux.

Les mots à rechercher dans le dictionnaire sont les mots formés des caractères alphabétiques de `'a'` à `'z'` (on ne considère pas les accents), avec au plus une majuscule en tête et pouvant éventuellement être composés de deux mots simples séparés par un tiret `'-'`, le second mot commençant forcément par une minuscule.
Ces mots sont séparés par des blancs, tabulations et fin de ligne ainsi que les parenthèses ouvrantes et fermantes, les guillemets, l'apostrophe et les signes de ponctuation : point, virgule, point virgule, deux points, point d'exclamation et d'interrogation.
L'ensemble des séparateurs est :
{`' ' '\t' '\n' '(' ')' '"' '\'' '.' ',' ';' ':' '!' '?'`}

Ainsi, si on écrit : `Hello WORLD` le mot `Hello` sera cherché dans le dictionnaire mais pas le mot `WORLD` qui contient des majuscules autrement qu'en première lettre. Si l'on écrit `[Hello] world!` alors la suite de caractères `[Hello]` sera ignorée car `Hello` n'est pas entouré de séparateurs (les crochets carrés ne sont pas des séparateurs), alors que le mot `world` sera recherché (l'espace et le point d'exclamation sont des séparateurs).

1. Proposer une caractérisation lexicale des adresses électroniques valides et donner une expression régulière permettant de les reconnaître.
2. Donner les expressions régulières correspondant aux mots valides et aux suites de caractères ignorés.
3. Proposer un analyseur lexical `ocamllex` qui prend en entrée une suite de caractères et renvoie le prochain token qui peut être de la forme :
   - `Dico`(s) où s est une chaîne correspondant à un mot à chercher dans le dictionnaire;
   - `Sep`(s) où s est une chaîne correspondant à une suite de séparateurs;
   - `Ignore`(s) où s est une chaîne à ignorer qui ne contient pas de séparateurs et ne correspond pas à un mot;
   - `Email`(s) où s est une chaîne correspondant à une adresse électronique valide.
4. Donner la suite de tokens correspondant à la phrase :

Your mail concerning $subject will be read on Tuesday November 5th.
Please contact `paulin@lri.fr`.