Département Informatique

Examen de Compilation - 4ème Année

Mercredi 13 janvier 2010 - 13h30-16h30 - Amphi A

Durée de l'épreuve : 3h.
Responsable : M. Ducassé (82 52)
Aucun document autorisé sauf une page A4 manuscrite (recto-verso).

L'énoncé comporte **??** pages, dont les pages **??** à **??** sont **à détacher, remplir et joindre à la copie.** Des définitions, possiblement utiles à cet examen, se trouvent en annexe, pages 6 à **??**. Le barème et les temps sont donnés à titre indicatif.

1 Questions de cours (*1/2 heure, 4,5 points*)

Question 1.1 *Que signifie le "1" dans "grammaire LL(1)" ?*

Question 1.2 *Supposons qu'un langage de programmation autorise les commentaires imbriqués, comme par exemple* `/* Ceci est /* un commentaire */ imbriqué */`.
A quelle(s) étape(s) de compilation seront-ils traités ? On justifiera en quelques phrases.

Question 1.3 *Donner le nom de la règle suivante, dire à quoi elle correspond. La paraphraser* **exhaustivement** *en français.*

$$\frac{TS \vdash E1 : T \quad TS \vdash E2 : T \quad ... \quad TS \vdash En : T}{TS \vdash [E1; E2; ...; En] : liste(T)}$$

1	*prog*	$\longrightarrow$	*"programme" bloc "fin"*
2	*bloc*	$\longrightarrow$	*"var" sdec corps*
3	*sdec*	$\longrightarrow$	*dec "," sdec*
4	*sdec*	$\longrightarrow$	*dec*
5	*dec*	$\longrightarrow$	*ident*
6	*corps*	$\longrightarrow$	*"debut" sinst "fin"*
7	*sinst*	$\longrightarrow$	*inst ";" sinst*
8	*sinst*	$\longrightarrow$	*inst*
9	*inst*	$\longrightarrow$	*bloc*
10	*inst*	$\longrightarrow$	*ident "←" expr*
11	*expr*	$\longrightarrow$	*exp "@" exp*
12	*expr*	$\longrightarrow$	*ident*
13	*ident*	$\longrightarrow$	*<lettre> sident*
14	*sident*	$\longrightarrow$	*<lettre> sident*
15	*sident*	$\longrightarrow$	*<chiffre> sident*
16	*sident*	$\longrightarrow$	ε

FIG. 1 – G : Grammaire d'un langage impératif simplifié

2 Problème

Le problème est divisé en deux parties *indépendantes* qui utilisent la grammaire non-contextuelle G de la figure 1 où l'opérateur "@" est associatif à droite.

2.1 Analyses lexicale et syntaxique (*1 heure, 6 points*)

Question 2.1 *Dans la grammaire G, on peut remarquer l'utilisation de différentes notations pour distinguer plusieurs types de mots. À quoi correspondent ces types de mots ?*

Question 2.2 *Lors de la compilation, est-ce que les mots entre guillemets (double quotes) vont être représentés par des chaînes de caractères ? Justifiez votre réponse.*

Question 2.3 *Quelles règles correspondent à l'analyse lexicale ? À l'analyse syntaxique ?*

Question 2.4 *Donner les ensembles V_N et V_T de la grammaire G.*

Question 2.5 *La grammaire présentée est-elle simplement LL(1) ? Justifiez votre réponse en une phrase.*

On se propose de fabriquer les tables SLR correspondant au problème en utilisant les algorithmes vus en cours et rappelés en annexe 3.2.

Attention, *les questions 2.6 à 2.9 doivent impérativement être rédigées directement sur les pages* **??** *à* **??** *de cet énoncé (que vous rendrez avec votre copie).*

Question 2.6 *Calculer les ensembles d'items SLR demandés page* **??** *pour la grammaire G, et seulement ceux là.*

Question 2.7 *Donner les ensembles "suivant" pour les non-terminaux de la grammaire, comme demandé page* **??***. On ne demande pas de faire un calcul formel.*

Question 2.8 *Remplir les parties des tables d'analyse SLR, comme demandé page* **??***, avec ce qui peut être construit à l'aide des ensembles précédemment calculés. On indiquera toutes les possibilités données par le calcul d'items dans les cases correspondantes.*

Question 2.9 *Sur la page* **??***, Essayer de résoudre les conflits détectés. Il est interdit de changer la grammaire. On peut toutefois s'appuyer sur des propriétés bien connues, ou sur des propriétés sans conséquence pour le programmeur. Dans ce cas, il faudra bien expliciter ces propriétés. Pour chacun des conflits, bien dire si on peut ou non le résoudre et si oui comment. Justifiez vos réponses.*

2.2 Analyse sémantique(*1 1/2 heure, 9,5 points*)

Les questions 2.12 et 2.13 doivent impérativement être rédigées directement sur les pages **??** *à* **??** *de cet énoncé (que vous rendrez avec votre copie). Les autres questions doivent être rédigées sur des intercalaires qui ne traitent que d'analyse sémantique.*

On se propose, maintenant, d'écrire une grammaire attribuée qui prend en entrée un programme correct et rend un programme transformé qui est le programme initial dans lequel les variables sont remplacées par un triplet (identificateur, numéro du bloc de déclaration, rang dans ce bloc). La portée des variables est celle utilisée habituellement, une variable est visible dans tout un bloc sauf si une variable de même nom a été déclarée dans un des sous-blocs.

Par exemple, le programme P ci-dessous

```
programme
   var a, b
   debut
     var b, c
     debut
        c <- a @ b ;
        var c, d
        debut
          c <- c @ b
        fin
     fin ;
     b <- a @ b ;
     var d
     debut
      d <- b @ a
     fin
   fin
fin
```

devient le programme P' ci-dessous.

```
programme
   var (a,1,1), (b, 1, 2)
   debut
     var (b,2,1), (c,2,2)
     debut
        (c,2,2) <- (a,1,1) @ (b,2,1) ;
        var (c,3,1), (d,3,2)
        debut
          (c,3,1) <- (c,3,1) @ (b,2,1)
        fin
     fin ;
     (b,1,2) <- (a,1,1) @ (b,1,2) ;
     var (d,4,1)
     debut
      (d,4,1) <- (b,1,2) @ (a,1,1)
     fin
   fin
fin
```

Question 2.10 *Sur un intercalaire dédié, donner l'arbre syntaxique correspond au programme P. Cet arbre utilisera la grammaire G, en omettant les terminaux. Par exemple, la règle*

prog $\longrightarrow$ *"programme" bloc "fin"*

ne donnera qu'une branche partant de prog et arrivant sur bloc, et la règle

inst $\longrightarrow$ *ident "←" expr*

ne donnera qu'une branche partant de inst et arrivant sur expr. Dans ce dernier cas, on mettra toute l'instruction sous expr (par exemple `b <- a @ b`*). On mettra également les identificateurs sous dec.*

Question 2.11 *Dans un premier temps on veut simplement fabriquer une table des symboles qui rassemble tous les triplets possibles. De manière* **exceptionnelle** *il est demandé que les compteurs de blocs et de rang dans les blocs soient gérés par des* **variables globales**. *Quelle est la conséquence sur la stratégie de parcours de l'arbre ?*

Question 2.12 *Construire une grammaire attribuée GA*1 *attachée à la grammaire non contextuelle G qui permet de construire cette table des symboles.*

Les deux compteurs mentionnés ci-dessus sont les seules variables globales autorisées.

La mise à jour des variables globales sera faite à l'aide d'instructions impératives, mises entre accolades. On considèrera que cette mise à jour se fait **avant** tout calcul d'attributs. Par exemple, dans les calculs suivants, `X.a` prend la valeur de `ma_var` **après** son incrémentation :

```
{ ma_var := ma_var + 1}
X.a = ma_var
```

On introduira les attributs explicitement, en indiquant leur objet, leur type et s'ils sont synthétisés ou hérités. On définira avec soin les structures de données utilisées. Les fonctions sémantiques devront être définies avec précision. Il n'est absolument pas demandé de code Yacc.

Question 2.13 *Spécifier maintenant une grammaire attribuée GA*2*, basée sur GA*1*, qui fabrique le programme transformé.* **Attention**, *on n'autorise pas d'autres variables globales que les compteurs de blocs et de rang dans les blocs.*

Question 2.14 *Si on n'avait pas autorisé les variables globales pour les compteurs, quelles auraient été les conséquences ?*
A contrario, si on avait utilisé des variables globales à la question 2.13 quelles auraient été les conséquences ?
*Qu'en déduisez-vous quant au choix du langage de programmation des grammaires à attributs GA*1 *et GA*2 *?*

Question 2.15 *On souhaiterait détecter le variables utilisées non déclarées. Dites en une ou deux phrases ce qu'il faudrait changer dans GA2.*

Fin des questions