CMP1 – Construction des compilateurs

EPITA – Promo 2012
Tous documents (notes de cours, polycopiés, livres) autorisés
Calculatrices et ordinateurs interdits.

Janvier 2010 (1h30)

Une copie synthétique, bien orthographiée, avec un affichage clair des résultats, sera toujours mieux notée qu'une autre demandant une quelconque forme d'effort de la part du correcteur. Une argumentation informelle mais convaincante, sera souvent suffisante. Une lecture préalable du sujet est recommandée.

1 Warm Up

1. Comment faire comprendre à votre scanner Flex que l'entrée Tiger `'for'` doit produire un token FOR et non un token ID (correspondant au symbole `'for'`) ?
2. Pourquoi est-ce que votre scanner Flex génère à partir de l'entrée Tiger '<=' un seul token LE (*lower or equal*) et non un token LT (*lower than*) suivi d'un token EQ (*equal*) ?
3. Qu'est-ce que le sucre syntaxique ?
4. Pourquoi les références du C++ peuvent-elles être considérées comme du sucre syntaxique ?

2 Dessine-moi un programme

Considérez la grammaire suivante, qui s'inspire d'un sous-ensemble du langage Logo.

```
⟨instrs⟩ ::= ⟨instr⟩
           | ⟨instrs⟩ ⟨instr⟩

⟨instr⟩  ::= forward num
           | left num
           | right num

           # Loop.
           | repeat num [ ⟨instrs⟩ ]

           # Subprogram definition.
           | to id ⟨instrs⟩ end

           # Subprogram call.
           | id
```

Dans cette grammaire, le terminal num désigne un entier littéral et le terminal id un identifiant (qui suit les mêmes règles lexicales que les identifiants du langage Tiger).

Le langage Logo permet (entre autres) de dessiner sur un écran à l'aide d'une « tortue » que l'on manipule avec des instructions comme `forward 10` (avance de 10 unité en traçant un trait) ou `right 30` (tourne à droite de 30 degrés).

1. Est-ce que cette grammaire est LL-k et si oui, pour quel k ? Pourquoi ?
2. Est-elle LR-k et si oui, pour quel k ? Justifiez votre réponse.
3. On décide d'implémenter un parser Bison pour ce langage. Quel(s) changement(s) éventuel(s) pensez-vous apporter à cette grammaire lors de cette opération ?
4. Les arbres de syntaxe abstraite (ASTs) produits par Bison seront implémentés sous forme d'objets C++, à l'instar de l'implémentation des ASTs de `tc`. Donnez une description synthétique des classes représentant les différents nœuds de la syntaxe abstraite du langage étudié dans cet exercice, soit sous forme de diagramme de classes UML, soit sous forme de code C++.
5. Voici un programme Logo qui trace un carré de 100 unités de côté :

   ```
   repeat 4 [ forward 100 right 90 ]
   ```

   Donnez une représentation arborescente (graphique ou textuelle) de la syntaxe abstraite de ce programme.
6. Même exercice avec le programme suivant :

   ```
   to square
   repeat 4 [ forward 100 right 90 ]
   end
   to triangle
   repeat 3 [ forward 50 right 120 ]
   end
   repeat 12 [ triangle repeat 3 [ square right 10 ] ]
   ```
7. Intéressons-nous à l'analyse des noms du programme de la question précédente. Sur le schéma de votre réponse à la question 6, indiquez par des flèches (de préférence en utilisant une couleur différente) les liens entre les sites d'utilisations de noms de sous-programmes et les sites de définitions correspondant.
8. La sémantique de ce langage n'a pas été spécifiée très formellement, et rien n'a été dit quant à la portée des identifiants de sous-programmes. Donnez un exemple d'entrée qui pourrait être interprétée comme une ambiguïté sémantique.
9. Essayons d'améliorer la spécification sémantique de notre langage pour pallier les problèmes évoqués à la question précédente. Quelle(s) règle(s) relative(s) aux noms de sous-programmes allez-vous ajouter pour lever les ambiguïtés éventuelles évoquées ci-avant ?
10. À votre avis, est-il nécessaire d'effectuer une analyse des types après la passe d'analyse des noms ? Justifiez votre réponse.
11. Si l'on souhaite écrire un interprète pour ce langage (travaillant directement à partir des ASTs), quel(s) outil(s) (au sens large) sera (seront) nécessaire(s) à son implémentation ?
12. **Bonus.** Écrivez un programme Logo qui dessine une spirale carrée.

3 Un peu de C++ pour la route

1. Pourquoi une méthode de classe (fonction membre `static` en C++) ne peut pas être abstraite (`virtual`) ?
2. À votre avis, pourquoi une méthode C++ ne peut pas être à la fois `virtual` et `template` ?
3. **Bonus.** Écrivez un bout de code C++ qui calcule statiquement (à la compilation, donc) la valeur 10! (bien entendu, l'affichage du résultat sera dynamique, puisqu'il faudra quand même exécuter le programme pour ce faire).