Partiel Parsage

Les informaticiens s'amuseront dans le premier problème, les mathématiciens trouveront leur bonheur dans le second.

1 Opérateurs n-aires

Soit un certain langage de programmation classique, avec de l'arithmétique, des structures de contrôle, etc. On s'intéresse uniquement au fragment de sa grammaire concernant l'opérateur `?:`, en pensant à son implémentation en Yacc/Bison.

Soit la grammaire suivante, où `exp` est le seul non-terminal :

```
exp: "exp"
   | exp '?' exp ':' exp;
```

1. Montrer que cette grammaire est ambiguë en exhibant tous les arbres de dérivation engendrant le plus petit mot ambigu.
2. On résoudra l'ambiguïté de cette grammaire en ne conservant que l'une de ces possibilités. Discuter tous les choix possibles et leurs vertus pour un langage de programmation.
3. Écrire une grammaire équivalente, mais non-ambiguë, ne retenant que l'une de ces interprétations.
4. Que nécessite la mise en œuvre de *la même résolution*, mais dans le cas de l'ensemble du langage de programmation ?
5. Dessiner exactement la partie de l'automate LALR(1) conduisant à l'état (ou aux états) exhibant le conflit (ou les conflits) résultant de l'ambiguïté de la grammaire originale.
6. On rappelle qu'en Yason, une règle hérite de la priorité/associativité de son terminal le plus à droite. Quelles sont les trois possibilités les plus simples de décoration de cette grammaire pour que Yason ne voie plus de conflit ?
7. Montrez que certaines de des possibilités de la question 6 peuvent aussi s'exprimer d'autres façons par des directives d'associativité/priorité.
8. Comparer les avantages du parsage de ce langage par, d'une part une implémentation de la grammaire de la question 3 en Yason, et, d'autre part, celle exploitant les directives d'associativité/priorité de Yason comme dans la question 6.
9. L'une des possibilités de la question 6 implique que la "grammaire" ainsi définie n'est pas équivalente à la grammaire originale. Pourquoi ?
10. Discuter les vertus du choix de la question 9 dans un langage de programmation.
11. Parmi les possibilités de la question 6 conduisant à une "grammaire équivalente" en Yason, quel est le choix exerçant la pression la plus faible sur le parseur ?
12. Informellement, mais de façon convaincante, traiter le cas de l'extension suivante de cet opérateur :

```
exp: "exp"
   | exp '?' ':' exp
   | exp '?' exp ':' exp
   | exp '?' exp ',' exp ':' exp
   ;
```

(on attend en particulier une description de l'ambiguïté de cette grammaire, et un mode d'implémentation Yason).

13. Proposer une grammaire décorée de directives Yason, sans conflit, généralisant cet opérateur à 2, 3, 4, 5, 6 et 7 arguments. Expliquez en particulier vos choix de terminaux en terme de pression sur le reste du langage.
14. Critiquer l'extension suivante :

```
exp: "exp"
   | exp '?' exp ':' exp
   | exp '?' exp ':' exp ':' exp
   ;
```

2 Grammaire Hors Contexte et LaTeX

Un *symbole distingué* dans un mot est une lettre à une position donnée dans ce mot.

Lemme de Ogden. Soit $\mathbf{G} = (\mathbf{T}, \mathbf{N}, S, \mathbf{P})$ une grammaire hors-contexte. Il existe une constante $k \geq 1$ telle que pour $\omega \in \mathbf{L}(\mathbf{G})$ de longueur supérieure ou égale à k et pour tout choix de k ou plus symboles distingués dans ω, il existe un découpage de ω en $uvwxy$ tel que :

1. w contient au moins un symbole distingué.
2. La chaîne vx contient au moins un symbole distingué.
3. La chaîne vwx contient au plus k symboles distingués.
4. Un certain $A \in \mathbf{N}$ permet la dérivation suivante pour tout $i \geq 0$:

$$S \Longrightarrow uAy \Longrightarrow uvAxy \overset{*}{\Longrightarrow} uv^i wx^i y$$

1. Montrer que le *pumping lemma*, ou "lemme $uvwxy$", est une conséquence du lemme de Ogden.
2. Montrer que le langage $\mathbf{L} = \{a^n b^m a^n b^m | n, m \geq 0\}$ n'est pas hors-contexte.
3. En déduire par l'absurde que le langage $\mathbf{L}' = \{\omega\omega | \omega \in (a+b)^*\}$ n'est pas hors-contexte.
4. Les utilisateurs de LaTeX peuvent définir des *environnements*. Par exemple :

   ```
   \newenvironment{foobar}{\textbf{Foo : }}{\textbf{ : Bar}}
   ```

   dont l'utilisation :

   ```
   \begin{foobar}
     Baz
   \end{foobar}
   ```

   donne

   Foo : Baz **: Bar**

   Bien entendu les environnements doivent être équilibrés. Expliquer l'impact de cette fonctionnalité sur la grammaire de LaTeX.
5. Comment parser cette construction.
6. En s'inspirant de la démonstration du lemme $uvwxy$, démontrer le lemme de Ogden.