Compilation : Langages et Grammaires

Dans cette épreuve, les non terminaux sont écrits en majuscules, les terminaux en minuscules ou entre guillemets, et ε désigne le mot vide.

1 Hiérarchie de Chomsky

Pour chacune des grammaires suivantes, préciser (i) son type dans la hiérarchie de Chomsky, (ii) si elle est ambiguë, (iii) le langage qu'elle engendre, (iv) le type du langage dans la hiérarchie, (v) un automate fini qui reconnaisse le même langage.

Justifier vos réponses.

1. `P` $\rightarrow$ `P inst ';'`
   `P` $\rightarrow \varepsilon$
2. `P` $\rightarrow$ `P1`
   `P` $\rightarrow \varepsilon$
   `P1` $\rightarrow$ `P1 ';' inst`
   `P1` $\rightarrow$ `inst`
3. `P` $\rightarrow$ `P1`
   `P` $\rightarrow \varepsilon$
   `P1` $\rightarrow$ `P1 ';' P1`
   `P1` $\rightarrow$ `inst`
4. `S` $\rightarrow$ `P`
   `P` $\rightarrow$ `p P Q R`
   `P` $\rightarrow$ `p q R`
   `R Q` $\rightarrow$ `Q R`
   `q Q` $\rightarrow$ `q q`
   `q R` $\rightarrow$ `q r`
   `r R` $\rightarrow$ `r r`

2 Parsage LL(1)

Soit le langage de la logique (dite propositionnelle) composée de deux symboles `t` (vrai) et `f` (faux), de l'opération unaire $\neg$ (non), des opérations binaires $\vee$ (ou) et $\wedge$ (et), et des parenthèses. Ce langage inclut des mots tels que :

$$t \wedge t \qquad t \vee f \qquad (t \wedge t) \vee (f \wedge f)$$

Noter que l'on parle de tout ce langage, et non seulement le sous langage des formules "vraies" ; il comprend donc aussi des mots tels que :

$$f \qquad \neg t \qquad \neg t \wedge f \wedge f \wedge t$$

1. Écrire une grammaire hors contexte naïve de la logique propositionnelle. On cherchera une formulation très lisible, au prix de l'ambiguïté.
2. Désambiguïser cette grammaire en considérant les règles suivantes :

(a) $\wedge$ et $\vee$ sont associatives à gauche ;

(b) $\neg$ est prioritaire sur $\wedge$;

(c) $\wedge$ est prioritaire sur $\vee$.

c'est-à-dire que $f \vee t \vee f \wedge \neg t \wedge f$ se lit $(f \vee (t \vee ((f \wedge (\neg t)) \wedge f)))$.

3. Expliquer pourquoi cette grammaire ne peut pas être LL(1).
4. Transformer cette grammaire en une grammaire susceptible d'être LL(1).
5. Quelle critique formuler sur la grammaire obtenue ?
6. Simplifier cette grammaire en s'autorisant l'utilisation de * dans les règles. Par exemple,

   `A` $\rightarrow$ `a A` et `A` $\rightarrow$ ε équivaut à `A` $\rightarrow$ `a*`.

7. En déduire que l'on peut écrire un parseur LL(1) pour cette grammaire.