Correction du QCM THL — Théorie des Langages

EPITA – Promo 2008

Juillet 2006

Il y a toujours exactement une seule réponse valable. Lorsque plusieurs réponses sont possibles, prendre la plus restrictive.

Le langage a^n est

- $\times$ fini
- $\rightarrow$ rationnel
- $\times$ non reconnaissable par automate fini
- $\times$ vide

Le langage $a^n b^n$ pour $n < 42^{51} - 1$ est

- $\times$ infini
- $\rightarrow$ rationnel
- $\times$ non reconnaissable par automate fini
- $\times$ vide

Le langage $(ab)^n$ est

- $\times$ fini
- $\rightarrow$ rationnel
- $\times$ non reconnaissable par automate fini
- $\times$ vide

Le langage $a^n b^m$, où n, m parcourent les entiers naturel, est

- $\times$ fini
- $\rightarrow$ rationnel
- $\times$ non reconnaissable par automate fini
- $\times$ vide

L'expression rationnelle étendue $[a - zA - Z][a - zA - Z0 - 9\_]*$ n'engendre pas :

- $\rightarrow$ `__STDC__`
- $\times$ *main*
- $\times$ *eval_expr*
- $\times$ *exit*_42

Un automate fini déterministe...

- $\times$ n'est pas nondéterministe
- $\times$ n'est pas à transitions spontanées
- $\rightarrow$ n'a pas plusieurs états initiaux
- $\times$ n'a pas plusieurs états finaux

Le langage $a^n b^n$ est

- $\times$ fini
- $\times$ rationnel
- $\rightarrow$ non reconnaissable par automate fini
- $\times$ vide

Quelle est la classe de la grammaire suivante ?

$$\begin{array}{rcl} P & \rightarrow & P \ inst \ ';' \\ P & \rightarrow & inst';' \end{array}$$

- $\rightarrow$ Rationnelle (Type 3)
- $\times$ Hors contexte (Type 2)
- $\times$ Sensible au contexte (Type 1)
- $\times$ Monotone (Type 1)

Quelle est la classe de la grammaire suivante ?

$$\begin{array}{rcl} A & \rightarrow & aABC \\ A & \rightarrow & abC \\ CB & \rightarrow & BC \\ bB & \rightarrow & bb \\ bC & \rightarrow & bc \\ cC & \rightarrow & cc \end{array}$$

- $\times$ Rationnelle (Type 3)
- $\times$ Hors contexte (Type 2)
- $\times$ Sensible au contexte (Type 1)
- $\rightarrow$ Monotone (Type 1)

Quelle propriété de cette grammaire est vraie ?

$$
\begin{array}{lcl}
S & \rightarrow & aSc \\
S & \rightarrow & c
\end{array}
$$

- $\times$ Linéaire à gauche
- $\times$ Linéaire à droite
- $\rightarrow$ Hors contexte
- $\times$ Ambiguë

Quelle propriété de cette grammaire est vraie ?

$$
\begin{array}{lcl}
S & \rightarrow & SpS \\
S & \rightarrow & n
\end{array}
$$

- $\times$ Linéaire à gauche
- $\times$ Linéaire à droite
- $\times$ Rationnelle
- $\rightarrow$ Ambiguë

Un langage quelconque est

- $\rightarrow$ toujours inclus dans un langage rationnel
- $\times$ toujours inclus dans un langage hors-contexte
- $\times$ toujours inclus dans un langage sensible au contexte
- $\times$ peut ne pas être inclus dans un langage défini par une grammaire

Soit L_r est un langage rationnel. Si $L \subset L_r$, alors

- $\times$ L est rationnel
- $\times$ L est hors-contexte
- $\times$ L est sensible au contexte
- $\rightarrow$ L peut ne pas être définissable par une grammaire

LL(k) signifie

- $\times$ lecture en deux passes de gauche à droite, avec k symboles de regard avant
- $\times$ lecture en deux passes de gauche à droite, avec une pile limitée à k symboles
- $\rightarrow$ lecture en une passe de gauche à droite, avec k symboles de regard avant
- $\times$ lecture en une passe de gauche à droite, avec une pile limitée à k symboles

Si une grammaire est LL(1), alors

- × elle n'est pas rationnelle
- × elle est rationnelle
- → elle n'est pas ambiguë
- × elle est ambiguë

Si un parseur LALR(1) a des conflits, alors sa grammaire

- × est ambiguë
- × n'est pas LR(1)
- → n'est pas LR(0)
- × n'est pas déterministe

Si une grammaire hors contexte est non ambiguë

- × elle est LL(1)
- × elle est LL(k)
- → elle n'est pas nécessairement LL
- × elle produit nécessairement des conflits dans un parseur LL

Quelle forme de l'arithmétique est LL(1) ?

×

$$S \rightarrow S + S \mid S * S \mid n$$

×

$$\begin{array}{rcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & n \end{array}$$

→

$$\begin{array}{rcl} E & \rightarrow & TE' \\ E' & \rightarrow & +TE' \mid T \\ T & \rightarrow & FT' \\ T' & \rightarrow & *FT' \mid F \\ F & \rightarrow & n \end{array}$$

× LL(1) ne permet pas de traiter l'arithmétique

Lex/Flex sont des

- → générateurs de scanners
- × générateurs de parsers
- × parseurs
- × scanners

Yacc repose sur l'algorithme

- × LL(k)
- × YACC(1)
- × LR(k)
- → LALR(1)