Université de Technologie de Compiègne

Examen : Final
NF11 - Théorie des Langages de Programmation

Semestre :	Printemps 2014	Documents :	Autorisés
Nb pages :	3	Durée :	2 heures

Dans un souci d'équité, il ne sera pas répondu à des questions pendant la durée de l'épreuve. Si l'énoncé semble comporter une imprécision, faites un choix pour la lever que vous indiquerez clairement dans votre copie.

Exercice 1 (4pt)

1. Donner un automate à pile reconnaissant le langage $L_1 = \{a^m b^n : 0 \leq n \leq m \text{ et } m - n \text{ est pair}\}$.
2. Donner un automate à pile reconnaissant le langage $L_2 = \{a^m (bc \mid de)^{m+n} a^n : m, n \geq 0\}$.

Exercice 2 (6pt)

Soit les grammaires G_1, G_2 et G_3 suivantes :

G_1 :

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow S \mid a \\ B &\rightarrow b \end{aligned}$$

G_2 :

$$\begin{aligned} S &\rightarrow A \mid BU \\ A &\rightarrow aA \mid a \\ B &\rightarrow bB \mid b \\ U &\rightarrow aUa \mid aa \end{aligned}$$

G_3 :

$$\begin{aligned} S &\rightarrow zMNz \\ M &\rightarrow aMa \mid z \\ N &\rightarrow bNb \mid z \end{aligned}$$

1. Eleminer la récursivité à gauche de la grammaire $\mathbf{G_1}$.
2. Factoriser à gauche la grammaire $\mathbf{G_2}$.
3. Construire les ensembles PREMIER et SUIVANT des symboles de $\mathbf{G_3}$.
4. $\mathbf{G_3}$ est-elle $LL(1)$. Si oui analyser la chaine $zazabzbz$.

Exercice 3 (4 pt)

CET EXERCICE EST A REDIGER SUR UNE COPIE SEPAREE DES AUTRES

Rappels :

Dans un template, la syntaxe *<value :style() ; separator = "...">* permet d'écrire chaque élément d'une liste nommée value, avec un template appelé style, séparés par la chaîne définie après le terme separator. Chaque élément de la liste value peut être une chaîne ou un objet quelconque dont on pourra utiliser les variables d'instance.

Question :

On désire générer une classe Test, telle que :

```
public class Test {
    String name;
    String givenName;
    public Test(String name, String givenName) {
      this.name = name;
      this.givenName = givenName;
    }
    public String getName {
      return name;
    }
    public void setName(String name) {
      this.name = name;
    }
    public String getGivenName {
      return givenName;
    }
    public void setGivenName(String givenName) {
      this.givenName = givenName;
    }
    public String toString() {
      return name + "-" + givenName;
    }
}
```

La seule donnée est le nom des variables d'instances dont le type String est connu (ici name et givenName, mais on pourrait avoir davantage de termes). On remarquera que les accesseurs nécessitent une chaîne commençant par une majuscule derrière le get et le set.

1. Proposer un patron de template ayant un seul attribut permettant de générer ce type de classe. Il pourra faire appel à cinq autres templates que l'on écrira.
2. La méthode public void render(String template, String attribute, Object l) permet de créer un fichier à partir du nom du template, du nom de l'attribut et de la valeur de l'attribut. Ecrire un programme simple permettant de générer la classe présentée en exemple ci-dessus.

Exercice 4 (6pt)

Soit les grammaires H_1 et H_2 suivantes :

H_1 :

1. $E \rightarrow T$
2. $E \rightarrow E + T$
3. $T \rightarrow id$
4. $T \rightarrow (E)$

H_2 :

$$S \rightarrow A \mid S ; A$$
$$A \rightarrow id = E \mid print\ E$$
$$E \rightarrow id \mid E + E \mid (E)$$

1. La grammaire $\mathbf{H_1}$ produit la table d'analyse $SLR(1)$ donnée par Tableau 1. Analyser la chaine $id + (id)$.

TABLE 1 – Table d'analyse $SLR(1)$

	Action					Successeur	
	id	+	(	)	$	E	T
0	D4		D6			1	5
1		D2			D9		
2	D4		D6				3
3		R2		R2	R2		
4		R3		R3	R3		
5		R1		R1	R1		
6	D4		D6			7	5
7		D2		D8			
8		R4		R4	R4		
9	Acceptation						

2. Donner l'ensemble SUIVANT des symboles de $\mathbf{H_2}$.
3. Calculer les états de l'automate relatif à $\mathbf{H_2}$ dans une analyse $LR(0)$.
4. Construire la table d'analyse $SLR(1)$ relative à la grammaire $\mathbf{H_2}$.
5. $\mathbf{H_2}$ est-elle $SLR(1)$? Justifier.