TP Théorie des Langages et Grammaire

25/03/2020

LE FEYER Aymeric

BAUDELET Conrad

Table des matières

Explication du code

Le code est séparé en 4 classes

→ Main
Lance le programme principal avec une interface JFrame (importée depuis javax.swing)

→ Sommet
Le sommet possède 4 arguments,
name qui est un String pour son nom, utile pour l'affichage dans le graphe
x et *y* qui sont les coordonnées du sommet dans le graphe, ces valeurs seront générées aléatoirement plus tard
s qui est un Object, ce sera utile d'un point de vue code pour pouvoir stocker directement le vertex, on verra cela plus tard

Il y a également une méthode *displaySommet* qui nous servira à créer *s*.

→ Arc
L'arc posséde 3 arguments
s1 et *s2* qui sont les sommets de cet arc
cout qui est la valeur de cet arc (String)

La méthode *displayArc* permet d'afficher l'arc dans le graphe

→ JGraph
C'est ici que tout se joue

Le constructeur créé la fenêtre et importe les informations du fichier
Ensuite, il affiche le graphe

La méthode *displayGraph* parcours tous les sommets et tous les arcs pour appeler la méthode *displaySommet* et *displayArc*

La méthode *importGraph* est la plus intéressante
On créé deux Pattern, le premier pour la détection lexicale des identifiants, le second pour savoir si la valeur est un chiffre. On aurait pu faire plus simple mais j'avais envie d'utiliser un Pattern (pour le second).

```
Pattern p = Pattern.compile("^[a-z]{1,3}[0-9]+$");
```

^ : Début de ligne
[a-z] : Ensemble des lettres minuscules
{1-3} : On prends 1 à 3 éléments de l'ensemble [a-z]
[0-9] : Ensemble des chiffres
+ : On prends au moins 1 élément de l'ensemble [0-9], sans limite
$: Fin de ligne

Ligne 75 : On parcours toutes les lignes du fichier dans une boucle while

La première ligne a une exécution différente des autres :
On vérifie que le premier mot de la ligne est "sommets"
Ensuite pour chaque autre mot, s'il est en règle, on ajoute le sommet correspondant avec des valeurs *x* et *y* aléatoires

Pour les arcs :
On vérifie que la ligne contient 4 mots, le mot "arc" en première position, 2 sommets puis 1 valeur entière
On vérifie que les deux sommets existent déjà, si un des sommets n'existe pas on renvoie une erreur
Si l'arc existe déjà, on rajoute une deuxième valeur (ligne 112)

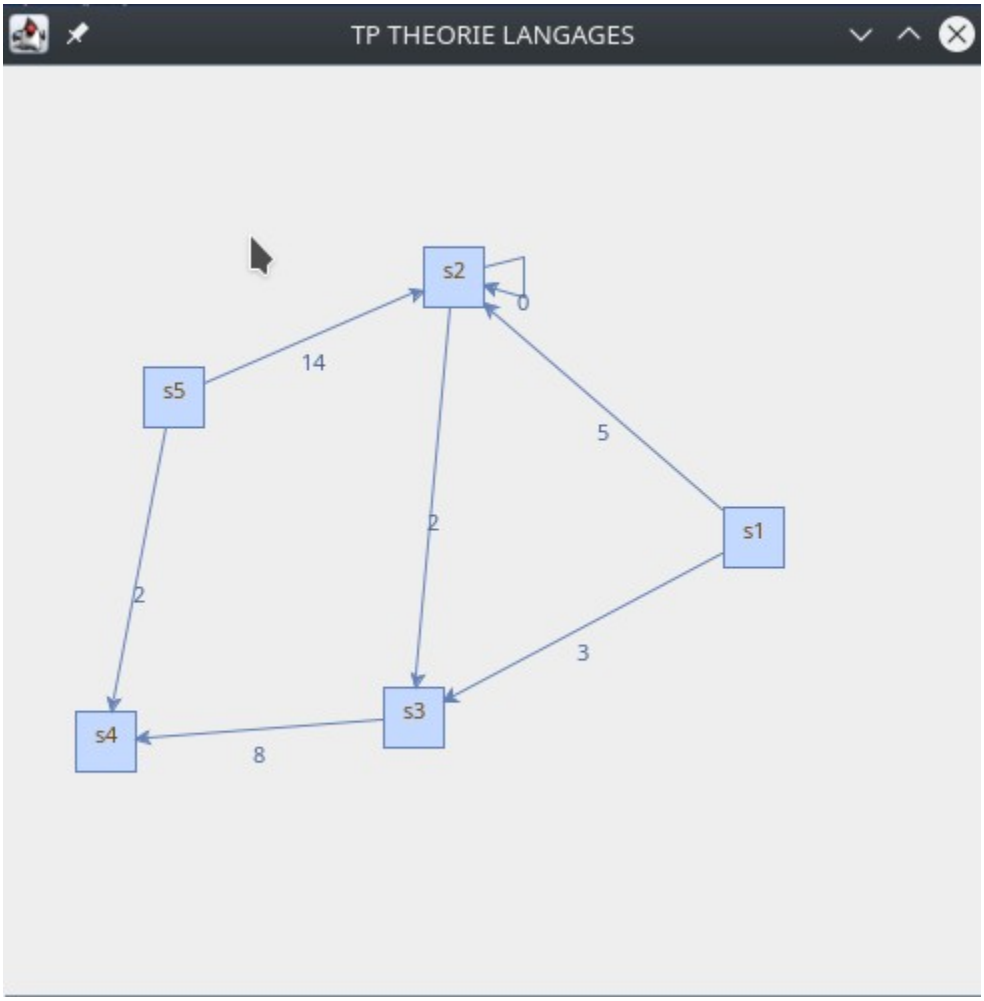
Exemple d'exécution

Avec le fichier input.txt suivant

```
sommets s1 s2 s3 s4 s5
arc s1 s2 5
arc s1 s3 3
```

```
arc s2 s2 0
arc s2 s3 2
arc s3 s4 8
arc s5 s4 2
arc s5 s2 14
```

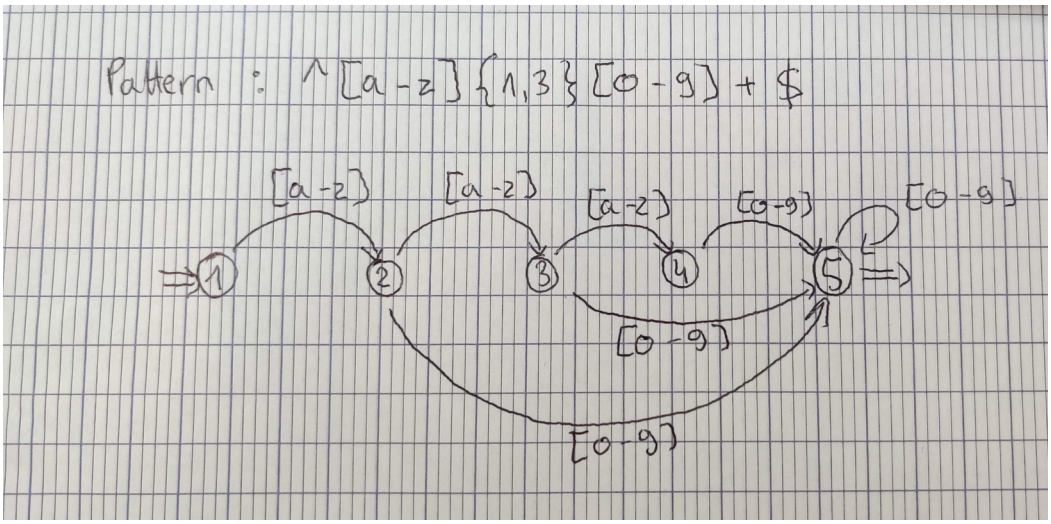
On obtiens le graphe ci-dessous



Analyse lexicale

Formation des identifiants

On utilise le pattern suivant pour la création des identifiants
`^[a-z]{1,3}[0-9]+$`
Ce qui reviens a l'automate suivant



Mots du lexique

Les seuls mots acceptes par le lexique sont “arc” et “sommets”
- sommets : attends un ou plusieurs identifiants
- arc : attends 2 identifiants et 1 valeur numérique

Analyse syntaxique

Agencement des mots sur les lignes

Toutes les lignes commencent par “arc”, sauf la première qui commence par “sommets”
Ensuite chaque identifiant est séparé par un espace

Analyse sémantique

Détection des erreurs de signification

A chaque création d’un arc, on parcours la liste des sommets pour s’assurer que le sommet en question existe.
Ensuite on parcours la liste de sommets pour vérifier si cet arc existe déjà, s’il existe, on rajoute la valeur après une “,” sur le graphe afin de ne pas superposer les arcs, s’il existe dans l’autre sens, on ajoute

la valeur en la décalant un peu pour ne pas superposer avec la valeur de l'autre arc
Ensuite on affiche cet arc

Erreurs possibles

Si la première ligne ne commence pas par "sommets"

→ `throw new Exception("Pas de sommet"); {line 93}`

Si un identifiant ne respecte pas l'automate (result[i] est le sommet posant problème)

→ `throw new Exception("Probleme de format sur le sommet " + result[i]); {line 89}`

Si on veux ajouter à un arc une valeur non entière positive (result[3] la valeur)

→ `throw new Exception("La valeur finale n'est pas un entier ! : " + result[3]); {line 100}`

Un sommet n'existe pas lors de la création d'un arc (result[1] et result[2] sont les deux sommets)

→ `throw new Exception("Au moins un des sommets n'existe pas " + result[1] + ", " + result[2]); {line 125}`

Le premier mot n'est pas "arc" (à partir de la seconde ligne) (result[0] est le premier mot)

→ `throw new Exception("Probleme sur le premier mot " + result[0]); {line 127}`

La syntaxe de création d'un arc n'est pas respectée

→ `throw new Exception("Probleme de parametre sur l'arc de la ligne : " + lineNumber); {line 128}`