JFlex et Jacc : mode d'emploi

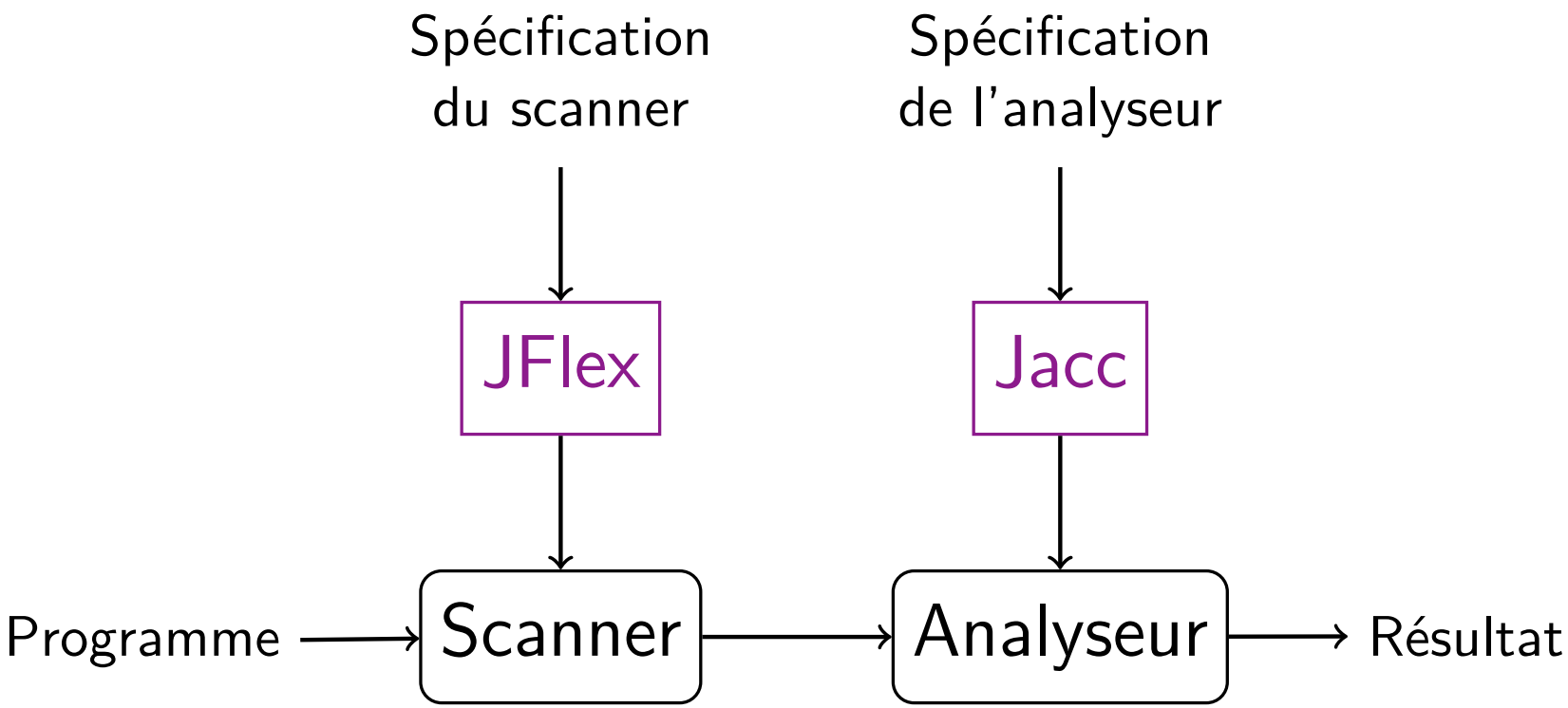
Sergiu IVANOV

sergiu.ivanov@u-pec.fr

Les diapos et le code disponibles en ligne :

http://lacl.fr/~sivanov/doku.php?id=fr:cours_de_theorie_des_langages

Image générale



Comment générer un scanner

Téléchargement et installation

1. télécharger JFlex : `http://jflex.de/release/jflex-1.6.1.zip`
2. désarchiver `jflex-1.6.1.zip`
3. copier le fichier `jflex-1.6.1/lib/jflex-1.6.1.jar` dans votre répertoire de travail

Création du scanner

1. créer la spécification du scanner en `scanner.lex`
2. lancer JFlex :

```
java -jar jflex-1.6.1.jar scanner.lex
```

3. compiler le code du scanner généré :

```
javac Yylex.java
```
(le fichier peut s'appeler différemment)

4. lancer le scanner :

```
java Yylex fichier.entree
```

Comment générer un analyseur (même truc (presque))

Téléchargement et installation

1. télécharger Jacc : `http://web.cecs.pdx.edu/~mpj/jacc/jacc.zip`
2. désarchiver `jacc.zip`
3. copier le fichier `jacc/jacc.jar` dans votre répertoire de travail

Création de l'analyseur

1. créer la spécification de l'analyseur en `parser.jacc`
2. lancer Jacc :

```
java -jar jacc.jar parser.jacc
```

3. compiler le code de l'analyseur généré :

```
javac Parser.java
```

Comment générer un analyseur (même truc (presque))

Téléchargement et installation

1. télécharger Jacc : `http://web.cecs.pdx.edu/~mpj/jacc/jacc.zip`
2. désarchiver `jacc.zip`
3. copier le fichier `jacc/jacc.jar` dans votre répertoire de travail

Création de l'analyseur

1. créer la spécification de l'analyseur en `parser.jacc`
2. lancer Jacc :

```
java -jar jacc.jar parser.jacc
```

3. compiler le code de l'analyseur généré :

```
javac Parser.java
```

Comment faire communiquer le scanner et l'analyseur ?

Le scanner ⇒ l'analyseur : les symboles

L'analyseur veut manipuler les codes numériques des symboles.

Créez une classe similaire à `ArithToken` dans votre scanner :

(l'exemple en ligne)

```
class Token {
  enum Type {
    SYMBOL1 (1), SYMBOL2 (2);
    public int intVal;
    Type(int v) { intVal = v; };
  }
  public Type type;
  ...
```

Pour une variable `t` de type `Token`, vous pourrez accéder à son code numérique ainsi : `t.type.intVal`.

Le scanner ⇒ l'analyseur : les codes numériques

Jacc génère une interface `XTokens`, qui contient les codes numériques des symboles :

```
interface XTokens {
   int SYMBOL1 = 0;
   int SYMBOL2 = 1;
   ...
}
```

Faites en sorte que les codes définis dans `Token` soient les mêmes que ceux définis dans `XTokens` *(même pour les symboles qui apparaissent dans* `XTokens` *en commentaires)*.