German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail

CSEN1002 Compilers Lab, Spring Term 2022
Task 5: CFG Left-Recursion Elimination

Due: Week starting 02.04.2022

1 Objective

For this task you will implement the context-free grammar (CFG) left-recursion elimination algorithm introduced in Lecture 3 of CSEN1003. Recall that a CFG is a quadruple (V, Σ, R, S) where V and Σ are disjoint alphabets (respectively, containing *variables* and *terminals*), $R \subseteq V \times (V \cup \Sigma)^*$ is a set of *rules*, and $S \in V$ is the *start variable.*

2 Requirements

- We make the following assumptions about input CFGs for simplicity.
  a) The set V of variables consists of upper-case English symbols.
  b) The start variable is the symbol S.
  c) The set Σ of terminals consists of lower-case English symbols (except the letter `e`).
  d) We only consider CFGs with no cycles and no ε-rules.
- You should implement a class constructor `CFG` and a method `lre` which takes an input string encoding a CFG and returns a string encoding an equivalent CFG which is not left-recursive.
- A string encoding a CFG is a semi-colon separated sequence of items. Each item represents a largest set of rules with the same left-hand side and is a comma-separated sequence of strings. The first string of each item is a member of V, representing the common left-hand side. The first string of the first item is S.
- For example, consider the CFG $(\{S, T, L\}, \{\mathtt{i}, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, R, S)$, where R is given by the following productions.

$$\begin{array}{lcl} S & \longrightarrow & S\,\mathtt{c}\,T \mid S\,\mathtt{a} \mid T \mid \mathtt{b} \\ T & \longrightarrow & \mathtt{a}\,S\,\mathtt{b} \mid \mathtt{i}\,\mathtt{a}\,L\,\mathtt{b} \mid \mathtt{i} \\ L & \longrightarrow & S\,\mathtt{d}\,L \mid S \end{array}$$

  This CFG will have the following string encoding.

  `S, ScT, Sa, T, b; T, aSb, iaLb, i; L, SdL, S`

- The function `LRE` will assume the ordering of variables as they appear in the string encoding of the CFG. Thus, in the above example, the variables are ordered thus: S, T, L.

- `lre` returns a string encoding the resulting CFG where a newly-introduced variable, for the elimination of immediate left-recursion for variable A, is the string A'. The letter `e` denotes the empty string. Newly added rules appear in the order indicated in Slides 33 and 34 of Lecture 3.

- Thus, for the above example, the output should be as follows (split here onto two lines for clarity).

```
S, TS′, bS′; S′, cTS′, aS′, e; T, aSb, iaLb, i; L, aSbS′dL, iaLbS′dL, iS′dL, bS′dL, aSbS′,
iaLbS′, iS′, bS′
```

- Important Details:
  - Your implementation should be done within the template file "`CFG.java`" (uploaded to the CMS).
  - You are not allowed to change package, file, constructor, or method names/signatures.
  - You are allowed to implement as many helper classes/methods within the same file (if needed).
  - Public test cases have been provided on the CMS for you to test your implementation.
  - Please ensure that the public test cases run correctly without modification before coming to the lab to maintain a smooth evaluation process.
  - Private test cases will be uploaded before your session and will have the same structure as the public test cases.

3 Evaluation

- Your implementation will be tested by running `lre` on five CFGs.
- You get two points for each correct output of `lre`; hence, a maximum of ten points.

4 Online Submission

- You should submit your code at the following link.

https://forms.gle/jhUfCyHvpz1is6xz8

- Submit one Java file (`CFG.java`) containing executable code.
- Online submission is due on Thursday, April 7, by 23:59.