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

CSEN1002 Compilers Lab, Spring Term 2020
Task 5: CFG Left-Recursion Elimination

Due: Week starting 17.03.2020

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

- You may use the programming language of your choice.
- 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.
  d) We only consider CFGs with no cycles and no ε-rules.
- You should implement a function `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.

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

  This CFG will have the following string encoding.

$$\mathtt{S}, \mathtt{ScT}, \mathtt{T}; \mathtt{T}, \mathtt{aSb}, \mathtt{iaLb}, \mathtt{i}; \mathtt{L}, \mathtt{SdL}, \mathtt{S}$$

- 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'. Thus, for the above example, the output should be as follows.

$$\texttt{S}, \texttt{TS}'; \texttt{S}', \texttt{cTS}', ; \texttt{T}, \texttt{aSb}, \texttt{iaLb}, \texttt{i}; \texttt{L}, \texttt{aSbS}'\texttt{dL}, \texttt{iaLbS}'\texttt{dL}, \texttt{iS}'\texttt{dL}, \texttt{aSbS}', \texttt{iaLbS}', \texttt{iS}'$$

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.