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

CSEN1002 Compilers Lab, Spring Term 2022
Task 7: $LL(1)$ **Parsing**

Due: Week starting 15.05.2022

1 Objective

For this task you will implement an $LL(1)$ parser using pushdown automata (PDA) and predictive parsing tables. Given an input context-free grammar $G = (V, \Sigma, R, S)$, along with the *First* and *Follow* sets for all rules, you need to (i) construct the predictive parsing table for G, (ii) construct the PDA equivalent to G, and (iii) implement an $LL(1)$ parser for G which makes use of the table and the PDA to direct its decisions. Given an input string w, the parser should signal an error if $w \notin L(G)$ and produce a derivation of w from S if $w \in L(G)$.

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 other than "`e`".
  d) The letter "`e`" represents ε.
- You should implement a class constructor `LL1CFG` which takes an input string encoding a CFG G, together with *First* and *Follow* sets for its rules, and one method `parse` which takes an input string w and returns a string encoding a left-most derivation of w in G; in case $w \notin L(G)$, this derivation ends with "`ERROR`." The `parse` method should construct a PDA equivalent to G and use the PDA together with the $LL(1)$ parsing table to reach its decision. Note that we will be testing `parse` using only $LL(1)$ grammars. Hence, you do not need to include a search algorithm in your implementation; w either has no derivation in G or has exactly one.
- A string encoding a CFG together with its *First* and *Follow* sets is a #-separated sequence of three items. The first item is a string encoding of the CFG, the second item is a string encoding of the *First* sets, and the third is a string encoding of the *Follow* sets.
- A string encoding of 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.

- The *First* sets are encoded by a semi-colon-separated sequence of items. Each item corresponds to a variable of the CFG. Items appear in the order in which the corresponding variables appear. An item is a comma-separated sequence of items. The first item is the variable name and subsequent items are string encodings of the *First* sets of each right-hand side of a rule for the item's variable. These sets appear in the same order of the corresponding rules and are concatenations of the symbols making up the represented set.

- The *Follow* sets are encoded by a semi-colon-separated sequence of items. Each item corresponds to a variable of the CFG. Items appear in the order in which the corresponding variables appear. An item is a comma-separated sequence of two items. The first item is the variable name and the second item is a string encoding of its *Follow* set. These sets are encoded by concatenations of the symbols making up the represented set.

- For example, consider the CFG $(\{S, T\}, \{\texttt{a}, \texttt{c}, \texttt{i}\}, R, S)$, where R is given by the following productions.

$$\begin{array}{lcl} S & \longrightarrow & \texttt{i}\ S\ T | \ \varepsilon \\ T & \longrightarrow & \texttt{c}\ S \mid \texttt{a} \end{array}$$

  This CFG will have the following string encoding.

$$\texttt{S}, \texttt{iST}, \texttt{e}; \texttt{T}, \texttt{cS}, \texttt{a\#S}, \texttt{i}, \texttt{e}; \texttt{T}, \texttt{c}, \texttt{a\#S}, \texttt{ca\$}; \texttt{T}, \texttt{ca\$}$$

- A string encoding a derivation is a comma-separated sequence of items. Each item is a sentential form representing a step in the derivation. The first item is `S`. If $w \in L(G)$ the last item is w; otherwise, it is `ERROR`. For example, given the above CFG, on input string `iiac`, `parse` should print the following string.

$$\texttt{S}, \texttt{iST}, \texttt{iiSTT}, \texttt{iiTT}, \texttt{iiaT}, \texttt{iiacS}, \texttt{iiac}$$

  On the other hand, on input string `iia`, `parse` should print the following.

$$\texttt{S}, \texttt{iST}, \texttt{iiSTT}, \texttt{iiTT}, \texttt{iiaT}, \texttt{ERROR}$$

- Important Details:
  - Your implementation should be done within the template file "`LL1CFG.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 using two grammars and five input strings for each.
- You get one point for each correct output; hence, a maximum of ten points.

4 Online Submission

- You should submit your code at the following link.

  https://forms.gle/j5Xzd9FET4RZDJbX9

- Submit one Java file (`LL1CFG.java`) containing executable code.
- Online submission is due on Thursday, May 19, by 23:59.