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

CSEN1002 Compilers Lab, Spring Term 2024
Task 8: $LL(1)$ **Parsing**

Due: Week starting 29.04.2024

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 letters.
  b) The start variable is the symbol S.
  c) The set Σ of terminals consists of lower-case English letters (except the letter `e`).
  d) The letter "`e`" represents ε.
  e) We only consider $LL(1)$ CFGs.
- You should implement a class constructor `CfgLl1Parser` and a method `parse`.
- `CfgLl1Parser`, a class constructor, takes one parameter which is a string description of a CFG, together with *First* and *Follow* sets for its rules, and constructs a CFG instance. A string encoding a CFG is of the form $V\#T\#R\#I\#O$.
  - V is a string representation of the set of variables; a semicolon-separated sequence of upper-case English letters, starting with S.
  - T is a string representation of the set of terminals; a semicolon-separated sequence of alphabetically sorted lower-case English letters.
  - R is a string representation of the set of rules. R is a semicolon-separated sequence of pairs. Each pair represents the largest set of rules with the same left-hand side. Pairs are of the form i/j where i is a variable of V and j is a string representation of the set of right-hand sides—a comma-separated sequence of strings. These pairs are sorted by the common left-hand side i based on the ordering of V.

- I is a string representation of the $First$ set of each rule. I is a semicolon-separated sequence of pairs. Pairs are of the form i/j where i is a variable of V and j is the string representation of the $First$ sets of each right-hand side of a rule for i—a comma-separated sequence of strings. These sets appear in the same order of the corresponding rules and are concatenations of the symbols making up the represented set. These pairs are sorted by the common left-hand side i based on the ordering of V.
- O is a string representation of the $Follow$ set of each variable. O is a semicolon-separated sequence of pairs. Pairs are of the form i/j where i is a variable of V and j is the string representation of the $Follow$ set of i. These sets are encoded by concatenations of the symbols making up the represented set. These pairs are sorted by the common left-hand side i based on the ordering of V.

- For example, consider the CFG $G_1 = (\{S, T\}, \{\texttt{a}, \texttt{c}, \texttt{i}\}, R, S)$, where R is given by the following productions.

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

  This CFG will have the following string encoding.

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

- `parse` 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 derivation is a semicolon-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 G_1, on input string `iiac`, `parse` should return the following string.

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

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

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

- Important Details:
  - Your implementation should be done within the template file "`CfgLl1Parser.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.
- The evaluation will take place during your lab session of the week starting Monday, April 29.

4 Online Submission

- You should submit your code at the following link.

  `https://forms.gle/qnW99xUPVg8MDzNG7`

- Submit one Java file (`CfgLl1Parser.java`) containing executable code.
- **Online submission is due by the end of your lab session.**