Syntax Analysis

Sudakshina Dutta

IIT Goa

26^{th} March, 2022

LR parsing

Note that in LL(1) parsing, 1 means Number of Lookahead symbols from input being considered.

- Most prevalent type of bottom-up parser
- Shift-reduce is a form of bottom-up parsing which can be implemented by LR method
- A table-driven form of parsing and similar to LL(1) parsing
- Almost all programming language constructs are covered
- Non-LR CFG exists; but are avoided for programming language constructs
- The LR method cover more language than LL method

Shift-reduce parsing

- The shift-reduce parser needs to know when to shift and when to reduce
- For understanding whether to shift or to reduce, it needs to keep track of how much has been parsed and how much is yet to be parsed

The shift-reduce parser

- A shift-reduce parser needs to know when to shift and when to reduce
- A stack is maintained which keeps track of $\$T$ and input symbols
  - The challenge is to know that T is not a handle, so the appropriate action is to shift $*$ and not to reduce T to E
- The LR parser makes shift-reduce decisions by maintaining states to keep track of where we are in a parse

LR(0) means no input symbol is guiding about whether something can be in stack or not.

- States represent sets of "items"
- An $LR(0)$ item is a production with a dot at some position of the body. The production $A \to XYZ$ yields the four items

$$A \to .XYZ$$
$$A \to X.YZ$$
$$A \to XY.Z$$
$$A \to XYZ.$$

The prodution $A \to \epsilon$ generates only one item, $A \to .$

- Intuitively, an item indicates how much of a production we have seen at a given point in parsing process

—The item $A \to .XYZ$ indicates that we hope to see a string derivable from XYZ next on the input

- We shall consider sets of $LR(0)$ items which is called *canonical LR(0) collection* which provides the basis for constructing a DFA
- For this purpose, we define an augmented grammar with a new start symbol S' and production $S' \to S$

  — Acceptance occurs when and only when the parser is about to reduce $S' \to S$
- We also define two functions, CLOSURE and GOTO

- **Closure of Item Sets**
- Let I be the set of items for a grammar G.
  1. Initially, add every item in I to $CLOSURE(I)$
  2. If $A \rightarrow \alpha.B\beta$ and $B \rightarrow \gamma$ is a production, then add the itemm $B \rightarrow .\gamma$ to $CLOSURE(I)$, if it is not already there. Apply this rule until no more new items can be added to $CLOSURE(I)$
- **The function GOTO**
- $GOTO(I, X)$, where I is a set of items and X is a grammar symbol, is defined to be closure of the set of all items $[A \rightarrow \alpha X.\beta]$ such that $[A \rightarrow \alpha.X\beta]$

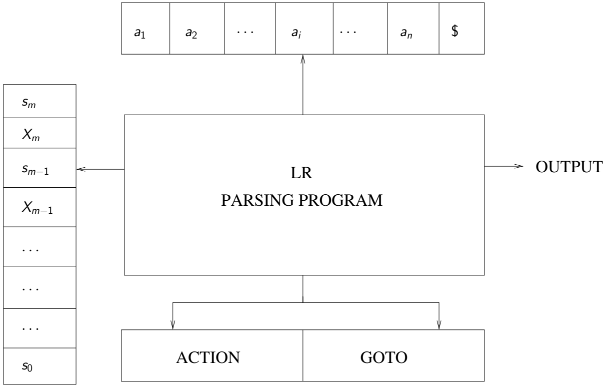
- **Example**
- Consider the augmented expression grammar

$$E' \rightarrow E$$
$$E \rightarrow E + T | T$$
$$T \rightarrow T * F | F$$
$$F \rightarrow (E) | id$$

If I is the set of one item $\{[E' \rightarrow .E]\}$, then $CLOSURE(I)$ contains the set of items as given below

$$E' \rightarrow .E$$
$$E \rightarrow .E + T$$
$$E \rightarrow .T$$
$$T \rightarrow .T * F$$
$$T \rightarrow .F$$
$$F \rightarrow .(E)$$
$$F \rightarrow .id$$

and $GOTO(I, E)$ is the set $\{E' \rightarrow E., E \rightarrow E. + T\}$



a_1
a_2
. . .
a_i
. . .
a_n
\$
s_m
X_m
s_{m-1}
X_{m-1}
. . .
. . .
. . .
s_0
LR
PARSING PROGRAM
OUTPUT
ACTION
GOTO

- Consider the following grammar for arithmetic expression $+$ and $*$

$$
\begin{gathered}
E \to E + T \\
E \to T \\
T \to T * F \\
T \to F \\
F \to (E) \\
F \to id
\end{gathered}
$$

Canonical collection of $LR(0)$ items

$$
\begin{aligned}
I_0 :\ & E' \rightarrow .E \\
& E \rightarrow .E + T \\
& E \rightarrow .T \\
& T \rightarrow .T * F \\
& T \rightarrow .F \\
& F \rightarrow .(E) \\
& F \rightarrow .id
\end{aligned}
$$

$$
\begin{aligned}
I_1 :\ & E' \rightarrow E. \\
& E \rightarrow E. + T
\end{aligned}
$$

$$
\begin{aligned}
I_2 :\ & E \rightarrow T. \\
& T \rightarrow T. * F
\end{aligned}
$$

$$
I_3 :\ T \rightarrow F.
$$

$$
\begin{aligned}
I_4 :\ & F \rightarrow (.E) \\
& E \rightarrow .E + T \\
& E \rightarrow .T \\
& T \rightarrow .T * F \\
& T \rightarrow .F \\
& F \rightarrow .(E) \\
& F \rightarrow .id
\end{aligned}
$$

$$
I_5 :\ F \rightarrow id.
$$

$$
\begin{aligned}
I_6 :\ & E \rightarrow E + .T \\
& T \rightarrow .T * F \\
& T \rightarrow .F \\
& F \rightarrow .(E) \\
& F \rightarrow .id
\end{aligned}
$$

$$
\begin{aligned}
I_7 :\ & T \rightarrow T * .F \\
& F \rightarrow .(E) \\
& F \rightarrow .id
\end{aligned}
$$

$$
\begin{aligned}
I_8 :\ & F \rightarrow (E.) \\
& E \rightarrow E. + T
\end{aligned}
$$

$$
\begin{aligned}
I_9 :\ & E \rightarrow E + T. \\
& T \rightarrow T. * F
\end{aligned}
$$

$$
I_{10} :\ T \rightarrow T * F.
$$

$$
I_{11} :\ F \rightarrow (E).
$$