Programming Languages and Computation

# Week 3: LL(1) Grammars

** 1. Consider the following grammar for arithmetic expressions:

$$
\begin{array}{rcll}
E & ::= & \mathsf{num} & (1) \\
 & | & \mathsf{id} & (2) \\
 & | & E + E & (3) \\
 & | & E * E & (4) \\
 & | & (E) & (5)
\end{array}
$$

The 6 terminal symbols of this grammar are:

$$\mathsf{num} \quad \mathsf{id} \quad + \quad * \quad ( \quad )$$

The nullable, first and follow sets for the nonterminals are:

| Nonterminal | Nullable? | First | Follow |
|---|---|---|---|
| E | × | num, id, ( | +, *, ) |

(a) For each of the rules (1) – (5), construct the first set of the right-hand side of the rule.

(b) Construct the parsing table for this grammar.

(c) What do the rule conflicts (cells of the parsing table containing more than one rule) of this gammar and the previous grammar have in common? Without constructing the parsing table, argue that the following grammar is sure to have a rule conflict:

$$
\begin{array}{rcl}
X & ::= & \mathsf{a} \\
 & | & \mathsf{b} \\
 & | & X\ \mathsf{c}
\end{array}
$$

Solution

(a)

$$
\begin{array}{rcl}
\mathsf{First(num)} & = & \mathsf{num} \\
\mathsf{First(id)} & = & \mathsf{id} \\
\mathsf{First(E + E)} & = & \mathsf{num, id, (} \\
\mathsf{First(E * E)} & = & \mathsf{num, id, (} \\
\mathsf{First}((E)) & = & \mathsf{(}
\end{array}
$$

(b)

| Nonterminal | num | id | + | * | ( | ) |
|---|---|---|---|---|---|---|
| E | 1,3,4 | 2,3,4 | | | 3,4,5 | |

(c) Whenever a non-terminal $X$ appears as the leftmost symbol in its own production rule $X ::= X\ \beta$, and First($X$) is non-empty, then there is certain to be a conflict. Since First($X$) is not empty, there must be some terminal symbol $a \in$ First($X$) and another rule $X ::= \gamma$ with $a \in$ First($\gamma$) – intuitively, this rule is the reason that $a$ is contained in First($X$). Then, since First($X\ \beta$) $\supseteq$ First($X$), it is certain that First($X\ \beta$) and First($\gamma$) will both contain $a$, and thus the cell of the parse table at coordinate $(X, a)$ will contain both rules.

* 2. Consider the following grammar for arithmetic expressions, whose terminals are the same as above:

$$\begin{array}{rcll} E & ::= & \mathsf{num} & (1) \\ & | & \mathsf{id} & (2) \\ & | & (\ F\ ) & (3) \\ F & ::= & +\ E\ E & (4) \\ & | & *\ E\ E & (5) \end{array}$$

The first, follow and nullable sets of the nonterminals can be described by:

| Nonterminal | Nullable? | First | Follow |
|---|---|---|---|
| E | × | num, id, ( | ), num, id, ( |
| F | × | +, * | ) |

(a) Construct the first set for each of the right-hand sides of rules in the grammar.

(b) Construct the parsing table for the grammar.

(c) Is the grammar LL(1)?

(d) Is num + num * id derivable in this grammar? If so, give a derivation.

Solution

(a)

$$\begin{array}{rcl} \mathsf{First}(\mathsf{num}) & = & \mathsf{num} \\ \mathsf{First}(\mathsf{id}) & = & \mathsf{id} \\ \mathsf{First}((\ F\ )) & = & ( \\ \mathsf{First}(+\ E\ E) & = & + \\ \mathsf{First}(*\ E\ E) & = & * \end{array}$$

(b)

| Nonterminal | num | id | + | * | ( | ) |
|---|---|---|---|---|---|---|
| E | 1 | 2 | | | 3 | |
| F | | | 4 | 5 | | |

(c) Yes.

(d) No.

* 3. Consider the following grammar for arithmetic expressions, whose terminals are as above:

$$\begin{array}{rcl} E & ::= & \mathsf{num}\ F \\ & | & \mathsf{id}\ F \\ & | & (\ E\ )\ F \\ F & ::= & +\ E \\ & | & *\ E \\ & | & \epsilon \end{array}$$

The nullable, first and follow sets can be described by:

| Nonterminal | Nullable? | First | Follow |
|---|---|---|---|
| E | × | num, id, ( | ) |
| F | ✓ | +, * | ) |

(a) Construct the first set of each of the right-hand sides of rules in the grammar.

(b) Construct the parsing table of the grammar.

(c) Is the grammar LL(1)?

(d) Is $\mathsf{num} + \mathsf{num} * \mathsf{id}$ derivable in this grammar? If so, give a derivation.

Solution

(a)

$$\begin{array}{rcl} \mathsf{First}(\mathsf{num}\ F) & = & \mathsf{num} \\ \mathsf{First}(\mathsf{id}\ F) & = & \mathsf{id} \\ \mathsf{First}((\ E\ )\ F) & = & ( \\ \mathsf{First}(+\ E) & = & + \\ \mathsf{First}(*\ E) & = & * \\ \mathsf{First}(\epsilon) & = & \end{array}$$

(b)

| Nonterminal | num | id | + | * | ( | ) |
|---|---|---|---|---|---|---|
| E | 1 | 2 | | | 3 | |
| F | | | 4 | 5 | | 6 |

(c) Yes.

(d) Yes:

$$\begin{aligned} E &\to \mathsf{num}\ F \\ &\to \mathsf{num} + E \\ &\to \mathsf{num} + \mathsf{num}\ F \\ &\to \mathsf{num} + \mathsf{num} * E \\ &\to \mathsf{num} + \mathsf{num} * \mathsf{id}\ F \\ &\to \mathsf{num} + \mathsf{num} * \mathsf{id} \end{aligned}$$

** 4. Give a CFG that describes the language of all subsequences of all permutations (i.e. no repetition) of the following keywords:

final static synchronized

Solution

$$\begin{array}{lcl} S & ::= & ABC \mid ACB \mid BAC \mid BCA \mid CAB \mid CBA \\ A & ::= & \mathsf{final} \mid \epsilon \\ B & ::= & \mathsf{static} \mid \epsilon \\ C & ::= & \mathsf{synchronized} \mid \epsilon \end{array}$$

*** 5. An $\epsilon$-production is a rule of shape $X ::= \epsilon$ (for some nonterminal $X$). A unit production is a rule of shape $X ::= Y$ (for some non-terminals $X$ and $Y$). Consider the following grammar $G$ with start symbol $S$:

$$\begin{array}{lcl} S & ::= & aSbb \mid T \\ T & ::= & bTaa \mid S \mid \epsilon \end{array}$$

This grammar has two unit productions $S ::= T$ and $T ::= S$, and it has an epsilon production $T ::= \epsilon$.

Give a grammar with *no* unit productions and *no* $\epsilon$-productions for the language $L(G) \setminus \{\epsilon\}$.

Solution

The following grammar describes $L(G) \setminus \{\epsilon\}$ without using unit or $\epsilon$-productions:

$$S ::= aSbb \mid bSaa \mid abb \mid baa$$

We can obtain it in the following way (which more-or-less follows a general approach to eliminating mutually recursive definitions in programming). First, since $S ::= T$ and $T ::= S$, we can replace the production $T ::= bTaa$ by $T ::= bSaa$ without changing the language of the grammar. The new grammar, call it $G'$, is:

$$\begin{array}{lcl} S & ::= & aSbb \mid T \\ T & ::= & bSaa \mid S \mid \epsilon \end{array}$$

The reasoning for why this does not change the language is as follows.

- The new grammar (with this production replaced) cannot derive any strings that were not already derivable in the old grammar, because we could already derive $T \to bTaa \to bSaa$ in the old grammar.
- The new grammar (with this production replaced) can derive all the string that were derivable in the old grammar because we can derive $T \to bSaa \to bTaa$ in the new grammar.

Therefore, the new grammar and the old grammar derive the same strings. Now, we have eliminated the directly recursive part of the $T$-rules, we can look to eliminate $T$ altogether from $G'$. We can replace the production $S ::= T$ in $G'$ by the three productions $S ::= bSaa \mid S \mid \epsilon$. This does not change the language of the grammar because if we use $S ::= T$, then we must then replace $T$ by something, and these are the three possibilities (so we are just removing a step from the derivation). This gives equivalent grammar $G''$:

$$\begin{array}{lll} S & ::= & aSbb \mid bSaa \mid S \mid \epsilon \\ T & ::= & bSaa \mid S \mid \epsilon \end{array}$$

But note that $S ::= S$ is a useless production which contributes nothing to the language described, so we can remove it. Then note that it is impossible to introduce a $T$ nonterminal when starting from $S$, so all the $T$ rules are useless in $G''$. Hence, we obtain the equivalent grammar $G'''$:

$$S ::= aSbb \mid bSaa \mid \epsilon$$

Now we still want to remove the $\epsilon$-production. In general this will change the language, because $S \to \epsilon$ is a valid derivation. However, we want to describe the language $L(G) \setminus \{\epsilon\}$ anyway, so this derivation should be impossible. We just observe that, in a derivation of a string in $L(G''') \setminus \{\epsilon\}$, the production $S ::= \epsilon$ will only be used after one of the other two productions. Thus we can combine $S \to aSbb \to abb$ and $S \to bSaa \to baa$ into two single rules and get rid of the $\epsilon$-production:

$$S ::= aSbb \mid bSaa \mid abb \mid baa$$