Introduction to Language Theory and Compilation
Solutions
Session 6: First sets, Follow sets and LL(1) parsing

Solutions

Ex. 1. With regards to the grammar:

1. Give the $\text{First}^1(A)$ and the $\text{Follow}^1(A)$ sets for each $A \in V$.

Symbol	**First**$^1()$	**Follow**$^1()$
S	`begin`	
program	`begin`	$
statement list	`ID read write`	`end`
statement tail	`ID read write` ε	`end`
statement	`ID read write`	`ID read write end`
id list	`ID`	`)`
id tail	`,` ε	`)`
expr list	`( ID INTLIT`	`)`
expr tail	`,` ε	`)`
expression	`( ID INTLIT`	`; , )`
primary tail	`+ -` ε	`; , )`
primary	`( ID INTLIT`	`+ - ; , )`
add op	`+ -`	`( ID INTLIT`

2. Give the First^2(<expression>) and the Follow^2(<expression>) sets.

First$^2(< expression >) = \{$ *((, (ID, (INTLIT, ID+, ID-, INTLIT+, INTLIT-, ID, INTLIT* $\}$

Follow$^2(< expression >) = \{$ *,(, ,ID, ,INTLIT,);,)+,)-,),,)), ;read, ;write, ;ID, ;end* $\}$

Ex. 2. Which grammars are LL(1)?

1. not LL(1) because $a \in \text{Follow}^1(A)$, $a \in \text{First}^1(A)$ and we have the rule $A \to \varepsilon$. Thus, $M[A,a] = \{\text{Produce 2}, \text{Produce 3}\}$.
2. LL(1).
3. LL(1).
4. not LL(1) because $b \in \text{First}^1(A)$, $b \in \text{Follow}^1(B)$ and we have the rule $B \to \varepsilon$. Thus, $M[A,b] = \{\text{Produce 3}, \text{Produce 4}\}$.

Ex. 3. Give the action table of the grammar:

	`-`	`(`	`)`	`ID`	`$`
<S>	1	1		1	
<Expr>	2	3		4	
<ExprTail>	5		6		6
<Var>				7	
<VarTail>	9	8	9		9

Ex. 4. Program a recursive descent parser (in C, C++, ...) for rules (14) through (21).

```
void void expression(){
    primary(); primary_tail();
}
void primary_tail(){
    token tok = next_token();
    // primary_tail is nullable, we check with the follow set
    // and return if primary_tail has to be epsilon
    switch(tok){
    case ';':
    case ',':
    case '(': return;
    }
    // otherwise, we try to match the rule
    add_op(); primary(); primary_tail();
}
void primary(){
    token tok = next_token();
    switch(tok){
    case '(':
        match('('); expression(); match(')');
        break;
    case ID:
        match(ID);
        break;
    case INTLIT:
        match(INTLIT);
        break;
    default:
        syntax_error(tok); break;
    }
}
void add_op(){
    token tok = next_token();
    switch(tok){
    case '+':
        match('+');
        break;
    case '-':
        match('-');
        break;
    default:
        syntax_error(tok); break;
    }
}
```