Introduction to Language Theory and Compilation Exercises

Session 6: First sets, Follow sets and LL(1) parsing

Reminders

First^k sets construction algorithm

```
begin
    foreach $a \in T$ do  $\text{First}^k(a) \leftarrow \{a\}$ ;
    foreach  $A \in V$ do  $\text{First}^k(A) \leftarrow \emptyset$ ;
    repeat
        foreach $A \in V$ do
            $\text{First}^k(A) \leftarrow \text{First}^k(A) \cup \{x \in T^* \mid A \to Y_1 Y_2 \ldots Y_n \wedge x \in$
              $\text{First}^k(Y_1) \oplus^k \text{First}^k(Y_2) \oplus^k \cdots \oplus^k \text{First}^k(Y_n)\}$ ;
    until stability;
```

Follow^k sets construction algorithm

```
begin
    foreach $A \in V \setminus \{S\}$ do $\text{Follow}^k(A) \leftarrow \emptyset$ ;
    $\text{Follow}^k(S) \leftarrow \{\varepsilon\}$ ;
    repeat
        if $B \to \alpha A \beta \in P$ (with $B \in V$ and $\alpha, \beta \in (V \cup T)^*$) then
            $\text{Follow}^k(A) \leftarrow \text{Follow}^k(A) \cup \{\text{First}^k(\beta) \oplus^k \text{Follow}^k(B)\}$ ;
    until stability;
```

Action table construction algorithm

```
begin
    $M \leftarrow \times$ ;
    foreach $A \to \alpha$ do
        foreach $a \in First^1(\alpha)$ do
            $M[A,a] \leftarrow M[A,a] \cup \text{Produce}(A \to \alpha)$ ;
        if $\varepsilon \in First^1(\alpha)$ then
            foreach $a \in Follow^1(A)$ do
                $M[A,a] \leftarrow M[A,a] \cup \text{Produce}(A \to \alpha)$ ;
    foreach $a \in T$ do $M[a,a] \leftarrow$ Match ;
    $M[\$, \varepsilon] \leftarrow$ Accept ;
```

Exercises

Ex. 1. With regards to the grammar given by Figure 1:

1. Give the $\text{First}^1(A)$ and the $\text{Follow}^1(A)$ sets for each $A \in V$.
2. Give the First^2(<expression>) and the Follow^2(<expression>) sets.

(1)	<S>	→	<program> $
(2)	<program>	→	begin <statement list> end
(3)	<statement list>	→	<statement> <statement tail>
(4)	<statement tail>	→	<statement> <statement tail>
(5)	<statement tail>	→	ε
(6)	<statement>	→	ID := <expression> ;
(7)	<statement>	→	read (<id list>) ;
(8)	<statement>	→	write (<expr list>) ;
(9)	<id list>	→	ID <id tail>
(10)	<id tail>	→	, ID <id tail>
(11)	<id tail>	→	ε
(12)	<expr list>	→	<expression> <expr tail>
(13)	<expr tail>	→	, <expression> <expr tail>
(14)	<expr tail>	→	ε
(15)	<expression>	→	<primary> <primary tail>
(16)	<primary tail>	→	<add op> <primary> <primary tail>
(17)	<primary tail>	→	ε
(18)	<primary>	→	(<expression>)
(19)	<primary>	→	ID
(20)	<primary>	→	INTLIT
(21)	<add op>	→	$+$
(22)	<add op>	→	$-$

Figure 1: Grammar for exercises 1 and 4

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

$$\begin{array}{lcl} S & \rightarrow & ABBA \\ A & \rightarrow & a \\ & & \varepsilon \\ B & \rightarrow & b \\ & & \varepsilon \end{array}$$

(a)

$$\begin{array}{lcl} S & \rightarrow & aSe \\ & & B \\ B & \rightarrow & bBe \\ & & C \\ C & \rightarrow & cCe \\ & & d \end{array}$$

(b)

$$\begin{array}{lcl} S & \rightarrow & ABc \\ A & \rightarrow & a \\ & & \varepsilon \\ B & \rightarrow & b \\ & & \varepsilon \end{array}$$

(c)

$$\begin{array}{lcl} S & \rightarrow & Ab \\ A & \rightarrow & a \\ & & B \\ & & \varepsilon \\ B & \rightarrow & b \\ & & \varepsilon \end{array}$$

(d)

Figure 2: Grammars for exercise 2

Ex. 3. Give the action table for the following grammar:

(1)	<S>	→	<expr> $
(2)	<expr>	→	− <expr>
(3)	<expr>	→	(<expr>)
(4)	<expr>	→	<var> <expr-tail>
(5)	<expr-tail>	→	− <expr>
(6)	<expr-tail>	→	ε
(7)	<var>	→	ID <var-tail>
(8)	<var-tail>	→	(<expr>)
(9)	<var-tail>	→	ε

Ex. 4. Program a recursive descent parser (in Java, C, C++, . . .) for rules (15) through (22) of the grammar given by Figure 1.