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**
      $\text{First}^k(A) \leftarrow \text{First}^k(A) \cup \{x \in T^* \mid A \rightarrow Y_1 Y_2 \dots Y_n \wedge x \in \text{First}^k(Y_1) \oplus^k \text{First}^k(Y_2) \oplus^k \cdots \oplus^k \text{First}^k(Y_n)\}$
  until *stability*

$Follow^k$ sets construction algorithm

begin
  foreach $A \in V$ **do** $\text{Follow}^k(A) \leftarrow \emptyset$;
  repeat
    if $B \rightarrow \alpha A \beta \in P$ **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 $A \rightarrow \alpha$ **do**
    foreach $a \in First^1(\alpha)$ **do**
      $M[A,a] \leftarrow M[A,a] \cup \text{Produce}(A \rightarrow \alpha)$;
    if $\varepsilon \in First^1(\alpha)$ **then**
      foreach $a \in Follow^1(A)$ **do**
        $M[A,a] \leftarrow M[A,a] \cup \text{Produce}(A \rightarrow \alpha)$;
  foreach $a \in T$ **do** $M[a,a] \leftarrow \text{Match}$;
  $M[\$, \varepsilon] \leftarrow \text{Accept}$;

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 $\text{First}^2(\text{<expression>})$ and the $\text{Follow}^2(\text{<expression>})$ sets.

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

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

Figure 1: Grammar for exercises 1 and 4

1

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

2

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

3

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

4

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

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 C, C++, ...) for rules (14) through (21) of the grammar given by Figure 1.