

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

### $Follow^k$ sets construction algorithm

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

### Action table construction algorithm

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

## Exercises

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

1. Give the  $First^1(A)$  and the  $Follow^1(A)$  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>	→	$\epsilon$
(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>	→	$\epsilon$
(12)	<expr list>	→	<expression> <expr tail>
(13)	<expr tail>	→	, <expression> <expr tail>
(14)	<expr tail>	→	$\epsilon$
(15)	<expression>	→	<primary> <primary tail>
(16)	<primary tail>	→	<add op> <primary> <primary tail>
(17)	<primary tail>	→	$\epsilon$
(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)?

$S \rightarrow ABBA$	$S \rightarrow aSe$	$S \rightarrow ABc$	$S \rightarrow Ab$
$A \rightarrow a$	$B$	$A \rightarrow a$	$A \rightarrow a$
$\epsilon$	$B \rightarrow bBe$	$\epsilon$	$B$
$B \rightarrow b$	$C$	$B \rightarrow b$	$\epsilon$
$\epsilon$	$C \rightarrow cCe$	$\epsilon$	$B \rightarrow b$
	$d$		$\epsilon$
(a)	(b)	(c)	(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>	→	$\epsilon$
(7)	<var>	→	ID <var-tail>
(8)	<var-tail>	→	( <expr> )
(9)	<var-tail>	→	$\epsilon$

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