

# CP471 Assignment 2

Aaron Exley

February 14, 2020

## LL(1) Language

```
<program> ::= <fdecls> <declarations> <statement_seq>.
<fdecls> ::= <fdec>; <fdecls> | ε
<fdec> ::= def <type> <fname> ( <params> ) <declarations> <statement_seq> fed
<params> ::= <type> <var><params_rest> |
<params_rest> ::= , <params> | ε
<fname> ::= <id>

<declarations> ::= <decl>; <declarations> | ε
<decl> ::= <type> <varlist>
<type> ::= int | double
<varlist> ::= <var><varlist_rest>
<varlist_rest> ::= , <varlist> | ε

<statement_seq> ::= <statement><statement_seq_rest>
<statement_seq_rest> ::= ; <statement_seq> | ε

<statement> ::= <var> = <expr> |
               if <bexpr> then <statement_seq> <if_rest> |
               while <bexpr> do <statement_seq> od |
               print <expr> |
               return <expr> |
<if_rest> ::= fi | else <statement_seq> fi |

<expr> ::= <term><expr_rest>
<expr_rest> ::= + <term><expr_rest> | - <term><expr_rest> | ε
<term> ::= <factor><term_rest>
<term_rest> ::= * <factor><term_rest> | / <factor><term_rest> | % <factor><term_rest> | ε
<factor> ::= <id><factor_rest> | <number> | (<expr>)
<factor_rest> ::= (<exprseq>) | <var_rest>
<exprseq> ::= <expr><exprseq_rest> |
<exprseq_rest> ::= , <exprseq> | ε

<bexpr> ::= <bterm><bexpr_rest>
<bexpr_rest> ::= or <bterm><bexpr_rest> | ε
<bterm> ::= <bfactor><bterm_rest>
<bterm_rest> ::= and <bfactor><bterm_rest> | ε
<bfactor> ::= (<bfactor_rest>) | not <bfactor>
<bfactor_rest> ::= <bexpr> | <expr> <comp> <expr>

<comp> ::= < | > | == | <= | >= | <>

<var> ::= <id><var_rest>
<var_rest> ::= [<expr>] | ε

<letter> ::= a | b | c | ... | z
<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<id> ::= <letter><id_rest>
```

$\langle \text{id\_rest} \rangle ::= \langle \text{letter} \rangle \langle \text{id\_rest} \rangle \mid \langle \text{digit} \rangle \langle \text{id\_rest} \rangle \mid \epsilon$   
 $\langle \text{number} \rangle ::= \langle \text{digit} \rangle \langle \text{number\_rest} \rangle$   
 $\langle \text{number\_rest} \rangle ::= \langle \text{digit} \rangle \langle \text{number\_rest} \rangle \mid . \langle \text{double} \rangle \mid e \langle \text{exp} \rangle \mid \epsilon$   
 $\langle \text{double} \rangle ::= \langle \text{digit} \rangle \langle \text{double\_rest} \rangle$   
 $\langle \text{double\_rest} \rangle ::= \langle \text{digit} \rangle \langle \text{double\_rest} \rangle \mid e \langle \text{exp} \rangle \mid \epsilon$   
 $\langle \text{exp} \rangle ::= \langle \text{digit} \rangle \langle \text{exp\_rest} \rangle \mid - \langle \text{digit} \rangle \langle \text{exp\_rest} \rangle$   
 $\langle \text{exp\_rest} \rangle ::= \langle \text{digit} \rangle \langle \text{exp\_rest} \rangle \mid \epsilon$

## First

$\text{First}(\langle \text{program} \rangle) = \{\text{def, int, double, a, b, c, ..., z, if, while, print, return, .}\}$

$\text{First}(\langle \text{fdecls} \rangle) = \{\text{def, } \epsilon\}$

$\text{First}(\langle \text{fdec} \rangle) = \{\text{def}\}$

$\text{First}(\langle \text{params} \rangle) = \{\text{int, double, } \epsilon\}$

$\text{First}(\langle \text{params\_rest} \rangle) = \{, , \epsilon\}$

$\text{First}(\langle \text{declarations} \rangle) = \{\text{int, double, } \epsilon\}$

$\text{First}(\langle \text{decl} \rangle) = \text{First}(\langle \text{type} \rangle) = \{\text{int, double}\}$

$\text{First}(\langle \text{varlist} \rangle) = \text{First}(\langle \text{var} \rangle) = \text{First}(\langle \text{fname} \rangle) = \{\text{id}\}$

$\text{First}(\langle \text{varlist\_rest} \rangle) = \{, , \epsilon\}$

$\text{First}(\langle \text{statement\_seq} \rangle) = \text{First}(\langle \text{statement} \rangle) = \{\text{a, b, c, ..., z, if, while, print, return, } \epsilon\}$

$\text{First}(\langle \text{statement\_seq\_rest} \rangle) = \{; , \epsilon\}$

$\text{First}(\langle \text{if\_rest} \rangle) = \{\text{fi, else}\}$

$\text{First}(\langle \text{expr} \rangle) = \text{First}(\langle \text{term} \rangle) = \text{First}(\langle \text{factor} \rangle) = \text{First}(\langle \text{exprseq} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, } \{ \}$

$\text{First}(\langle \text{expr\_rest} \rangle) = \{+, -, \epsilon\}$

$\text{First}(\langle \text{term\_rest} \rangle) = \{*, /, \%, \epsilon\} \mid$

$\text{First}(\langle \text{exprseq\_rest} \rangle) = \{, , \epsilon\}$

$\text{First}(\langle \text{bexpr} \rangle) = \text{First}(\langle \text{bterm} \rangle) = \text{First}(\langle \text{bfactor} \rangle) = \{ (, \text{not} \}$

$\text{First}(\langle \text{bexpr\_rest} \rangle) = \{\text{or, } \epsilon\}$

$\text{First}(\langle \text{bterm\_rest} \rangle) = \{\text{and, } \epsilon\}$

$\text{First}(\langle \text{bfactor\_rest} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, } (, \text{not}\}$

$\text{First}(\langle \text{comp} \rangle) = \{<, >, ==, <=, >=, <>\}$

$\text{First}(\langle \text{var\_rest} \rangle) = \{[, \epsilon\}$

$\text{First}(\langle \text{letter} \rangle) = \text{First}(\langle \text{id} \rangle) = \text{First}(\langle \text{number} \rangle) = \text{First}(\langle \text{double} \rangle) = \{\text{a, b, c, ..., z}\}$

$\text{First}(\langle \text{digit} \rangle) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$

$\text{First}(\langle \text{id\_rest} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, } \epsilon\}$

$\text{First}(\langle \text{number\_rest} \rangle) = \text{First}(\langle \text{double\_rest} \rangle) = \{\text{a, b, c, ..., z, ., e, } \epsilon\}$

$\text{First}(\langle \text{exp} \rangle) = \{\text{a, b, c, ..., z, -}\}$

$\text{First}(\langle \text{exp\_rest} \rangle) = \{\text{a, b, c, ..., z, } \epsilon\}$

## Follow

Follow(<program>) = {.  
Follow(<fdecls>) = {int, double, a, b, c, ..., z, if, while, print, return, .}  
Follow(<fdec>) = Follow(<decl>) = Follow(<varlist>) = Follow(<varlist\_rest>) = {;  
Follow(<params>) = Follow(<params\_rest>) = Follow(<exprseq>) = Follow(<exprseq\_rest>) = Follow(<bfactor\_rest>)  
= {  
Follow(<fname>) = {(

Follow(<declarations>) = {int, double, a, b, c, ..., z, if, while, print, return, .}  
Follow(<type>) = {a, b, c, ..., z}

Follow(<statement\_seq>) = Follow(<statement\_seq\_rest>) = {fi, else, od, fed, .}

Follow(<statement>) = Follow(<if\_rest>) = {;, fi, else, od, fed, .}

Follow(<expr>) = Follow(<expr\_rest>) = Follow(<comp>) = {), , , ;, ], fi, else, od, fed, .}  
Follow(<term>) = Follow(<term\_rest>) = {+, -, ), , , ;, fi, else, od, fed, .}  
Follow(<factor>) = Follow(<factor\_rest>) = Follow(<number>) = Follow(<number\_rest>) = Follow(<double>)  
= Follow(<double\_rest>) = Follow(<exp>) = Follow(<exp\_rest>) = {\*, /, %, +, -, ), , , ;, fi, else, od, fed, .}

Follow(<bexpr>) = Follow(<bexpr\_rest>) = {then, do}  
Follow(<bterm>) = Follow(<bterm\_rest>) = {or, then, do}  
Follow(<bfactor>) = {and, then, do}

Follow(<var>) = Follow(<var\_rest>) = {=, ), ;, \*, /, %, +, -, ), , , ;, fi, else, od, fed, .}  
Follow(<letter>) = Follow(<digit>) = {a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, (, [, ), ;, =, \*, /, %, +, -, ), , , ;, fi,  
else, od, fed, .}

Follow(<id>) = Follow(<id\_rest>) = {(, [, ), ;, =, \*, /, %, +, -, ), , , ;, fi, else, od, fed, .}