

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

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

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

$\text{First}(\langle \text{varlist} \rangle) = \text{First}(\langle \text{var} \rangle) = \text{First}(\langle \text{fname} \rangle) = \{\text{a, b, c, ..., z}\}$
 $\text{First}(\langle \text{varlist_rest} \rangle) = \{, , \epsilon\}$

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

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

$\text{First}(\langle \text{expr} \rangle) = \text{First}(\langle \text{term} \rangle) = \text{First}(\langle \text{factor} \rangle) = \text{First}(\langle \text{exprseq} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, \{ \}}$
 $\text{First}(\langle \text{expr_rest} \rangle) = \{+, -, \epsilon\}$
 $\text{First}(\langle \text{term_rest} \rangle) = \{*, /, \%, \epsilon\}$
 $\text{First}(\langle \text{factor_rest} \rangle) = \{ (, [, \epsilon \}$
 $\text{First}(\langle \text{exprseq_rest} \rangle) = \{, , \epsilon\}$

$\text{First}(\langle \text{bexpr} \rangle) = \text{First}(\langle \text{bterm} \rangle) = \text{First}(\langle \text{bfactor} \rangle) = \{ (, \text{not} \}$
 $\text{First}(\langle \text{bexpr_rest} \rangle) = \{\text{or, } \epsilon\}$
 $\text{First}(\langle \text{bterm_rest} \rangle) = \{\text{and, } \epsilon\}$
 $\text{First}(\langle \text{bfactor_rest} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, (, not}\}$

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

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

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

$\text{First}(\langle \text{id_rest} \rangle) = \{\text{a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, } \epsilon\}$
 $\text{First}(\langle \text{number_rest} \rangle) = \text{First}(\langle \text{double_rest} \rangle) = \{\text{a, b, c, ..., z, ., e, } \epsilon\}$
 $\text{First}(\langle \text{exp} \rangle) = \{\text{a, b, c, ..., z, -}\}$
 $\text{First}(\langle \text{exp_rest} \rangle) = \{\text{a, b, c, ..., z, } \epsilon\}$

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, fed, 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, .}