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>
```

<id_rest> ::= <letter><id_rest> | <digit><id_rest> | ϵ

<number> ::= <digit><number_rest>
<number_rest> ::= <digit><number_rest> | .<double> | e<exp> | ϵ
<double> ::= <digit><double_rest>
<double_rest> ::= <digit><double_rest> | e<exp> | ϵ
<exp> ::= <digit><exp_rest> | -<digit><exp_rest>
<exp_rest> ::= <digit><exp_rest> | ϵ

First

First(<program>) = {def, int, double, a, b, c, ..., z, if, while, print, return, .}
First(<fdecls>) = {def, ϵ}
First(<fdec>) = {def}
First(<params>) = {int, double, ϵ}
First(<params_rest) = {, , ϵ }

First(<declarations>) = {int, double, ϵ}
First(<decl>) = First(<type>) = {int, double}

First(<varlist>) = First(<var>) = First(<fname>) = {id}
First(<varlist_rest>) = {, , ϵ}

First(<statement_seq>) = First(<statement>) = {a, b, c, ..., z, if, while, print, return, ϵ}
First(<statement_seq_rest>) = {; , ϵ}

First(<if_rest>) = {fi, else}

First(<expr>) = First(<term>) = First(<factor>) = First(<exprseq>) = {a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, (}
First(<expr_rest>) = {+, -, ϵ}
First(<term_rest>) = {*, /, %, ϵ} |
First(<exprseq_rest>) = {, , ϵ}

First(<bexpr>) = First(<bterm>) = First(<bfactor>) = {(, not}
First(<bexpr_rest>) = {or, ϵ}
First(<bterm_rest>) = {and, ϵ}
First(<bfactor_rest>) = {a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, (, not}

First(<comp>) = {<, >, ==, <=, >=, <>}

First(<var_rest>) = {[, ϵ}

First(<letter>) = First(<id>) = First(<number>) = First(<double>) = {a, b, c, ..., z}
First(<digit>) = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}

First(<id_rest>) = {a, b, c, ..., z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ϵ}
First(<number_rest>) = First(<double_rest>) = {a, b, c, ..., z, ., e, ϵ}
First(<exp>) = {a, b, c, ..., z, -}
First(<exp_rest>) = {a, b, c, ..., z, ϵ}

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, .}