

CS 4240 Phase 1

David Zhang

February 23, 2014

1 Lexical Rules

Pre-parse the input to remove comments with the following DFA. The resulting NOT-COMMENT tokens are concatenated together.

Everything under the token column represents either generated tokens or custom table actions on the character buffer used to produce the text associated with tokens.

| start state | symbol | next state | token |
|--------------|---------------------|--------------|-------------|
| START | $\Sigma - \{/, \}$ | START | NOT-COMMENT |
| START | " | STRING | |
| START | / | SLASH | |
| STRING | $\Sigma - \{\\, \}$ | STRING | |
| STRING | \ | STRING-SLASH | |
| STRING | " | START | NOT-COMMENT |
| STRING-SLASH | Σ | STRING | |
| SLASH | $\Sigma - \{*\}$ | START | NOT-COMMENT |
| SLASH | * | COMMENT | |
| COMMENT | $\Sigma - \{*\}$ | COMMENT | |
| COMMENT | * | COMMENT-END | |
| COMMENT-END | $\Sigma - \{*, /\}$ | COMMENT | |
| COMMENT-END | * | COMMENT-END | |
| COMMENT-END | / | START | COMMENT |

The DFA for uncommented code.

Note, any time backtracking is mentioned, it essentially is the same as treating the current state as the start state and doing the corresponding transitions or token generations. This is included to simplify the table by removing duplication of the start state transitions.

Note, all ids are later matched character by character with keywords to determine if they are keywords.

Note, drop character for the error state means that the last read character is ignored, and the state remains unchanged.

| start state | symbol | next state | token |
|-------------------|-----------------------------|-------------------|---------------------|
| START | + | START | PLUS |
| START | - | START | MIN |
| START | * | START | MULT |
| START | / | START | DIV |
| START | = | START | EQ |
| START | (| START | LPAREN |
| START |) | START | RPAREN |
| START | , | START | COMMA |
| START | & | START | AND |
| START | — | START | OR |
| START | [| START | LSQUARE |
| START |] | START | RSQUARE |
| START | ; | START | SEMI |
| START | < | LANGLE | |
| START | > | RANGLE | |
| START | : | COLON | |
| START | 0-9 | INT-LIT | |
| START | “ | STRING-LIT | |
| START | a-zA-Z | ID | |
| START | whitespace | START | ignore |
| START | others | ERROR | drop character |
| LANGLE | $\Sigma - \{=, \}$ | START | LESS , backtrack |
| LANGLE | > | START | NOTEQ |
| LANGLE | = | START | LESSEQ |
| RANGLE | $\Sigma - \{=\}$ | START | GREATER, backtrack |
| RANGLE | = | START | GREATEREQ |
| COLON | = | START | ASSIGN |
| COLON | $\Sigma - \{=\}$ | START | COLON, backtrack |
| INT-LIT | 0-9 | INT-LIT | |
| INT-LIT | $\Sigma - 0 - 9$ | START | INT-LIT, backtrack |
| ID | a-zA-Z0-9_ | ID | |
| ID | $\Sigma - a - zA - Z0 - 9_$ | START | ID, backtrack |
| STRING-LIT | $\Sigma - \backslash$ | STRING-LIT | |
| STRING-LIT | ” | START | STRING-LIT |
| STRING-LIT | \ | STRING-LIT-SLASH | |
| STRING-LIT-SLASH | n | STRING-LIT | |
| STRING-LIT-SLASH | t | STRING-LIT | |
| STRING-LIT-SLASH | ” | STRING-LIT | |
| STRING-LIT-SLASH | \ | STRING-LIT | |
| STRING-LIT-SLASH | ^ | STRING-LIT-CTL | |
| STRING-LIT-SLASH | 0-9 | STRING-LIT-CODE-1 | |
| STRING-LIT-SLASH | whitespace | STRING-LIT-SPACE | ignore 2 characters |
| STRING-LIT-SLASH | others | ERROR | drop character |
| STRING-LIT-CTL | @A-Z[\\]^_ . | STRING-LIT | |
| STRING-LIT-CTL | others | ERROR | drop character |
| STRING-LIT-CODE-1 | 0-9 | STRING-LIT-CODE-2 | |
| STRING-LIT-CODE-1 | others | ERROR | drop character |
| STRING-LIT-CODE-2 | 0-9 | STRING-LIT | |
| STRING-LIT-CODE-2 | others | ERROR | drop character |
| STRING-LIT-SPACE | whitespace | STRING-LIT-SPACE | ignore |
| STRING-LIT-SPACE | \ | STRING-LIT | ignore |
| STRING-LIT-SPACE | others | ERROR | drop character |

2 Grammar Rules

Raw Grammar:

| symbol | rule |
|---|--|
| $\langle \text{tiger-program} \rangle$ | let $\langle \text{declaration-segment} \rangle$ in $\langle \text{stat-seq} \rangle$ end |
| $\langle \text{declaration-segment} \rangle$ | $\langle \text{type-declaration-list} \rangle \langle \text{var-declaration-list} \rangle \langle \text{funct-declaration-list} \rangle$ |
| $\langle \text{type-declaration-list} \rangle$ | $\langle \text{type-declaration} \rangle \langle \text{type-declaration-list} \rangle$ |
| $\langle \text{type-declaration-list} \rangle$ | NULL |
| $\langle \text{var-declaration-list} \rangle$ | $\langle \text{var-declaration} \rangle \langle \text{var-declaration-list} \rangle$ |
| $\langle \text{var-declaration-list} \rangle$ | NULL |
| $\langle \text{funct-declaration-list} \rangle$ | $\langle \text{funct-declaration} \rangle \langle \text{funct-declaration-list} \rangle$ |
| $\langle \text{type-declaration} \rangle$ | type id = $\langle \text{type} \rangle$; |
| $\langle \text{type} \rangle$ | $\langle \text{type-id} \rangle$ |
| $\langle \text{type} \rangle$ | array [INTLIT] of $\langle \text{type-dim} \rangle$ of $\langle \text{type-id} \rangle$ |
| $\langle \text{type-dim} \rangle$ | [INTLIT] $\langle \text{type-dim} \rangle$ |
| $\langle \text{type-dim} \rangle$ | NULL |
| $\langle \text{type-id} \rangle$ | int |
| $\langle \text{type-id} \rangle$ | string |
| $\langle \text{type-id} \rangle$ | id |
| $\langle \text{var-declaration} \rangle$ | var $\langle \text{id-list} \rangle$: $\langle \text{type-id} \rangle \langle \text{optional-init} \rangle$; |
| $\langle \text{id-list} \rangle$ | id |
| $\langle \text{id-list} \rangle$ | id , $\langle \text{id-list} \rangle$ |
| $\langle \text{optional-init} \rangle$ | NULL |
| $\langle \text{optional-init} \rangle$ | := $\langle \text{const} \rangle$ |
| $\langle \text{funct-declaration} \rangle$ | function id ($\langle \text{param-list} \rangle$) $\langle \text{ret-type} \rangle$ begin $\langle \text{stat-seq} \rangle$ end ; |
| $\langle \text{param-list} \rangle$ | NULL |
| $\langle \text{param-list} \rangle$ | $\langle \text{param} \rangle \langle \text{param-list-tail} \rangle$ |
| $\langle \text{param-list-tail} \rangle$ | NULL |
| $\langle \text{param-list-tail} \rangle$ | , $\langle \text{param} \rangle \langle \text{param-list-tail} \rangle$ |
| $\langle \text{ret-type} \rangle$ | NULL |
| $\langle \text{ret-type} \rangle$ | : $\langle \text{type-id} \rangle$ |
| $\langle \text{param} \rangle$ | id : $\langle \text{type-id} \rangle$ |
| $\langle \text{stat-seq} \rangle$ | $\langle \text{stat} \rangle \langle \text{stat-seq} \rangle$ |
| $\langle \text{stat-seq} \rangle$ | $\langle \text{stat} \rangle$ |
| $\langle \text{stat} \rangle$ | $\langle \text{lvalue} \rangle := \langle \text{expr} \rangle$; |
| $\langle \text{stat} \rangle$ | if $\langle \text{expr} \rangle$ then $\langle \text{stat-seq} \rangle$ endif ; |
| $\langle \text{stat} \rangle$ | if $\langle \text{expr} \rangle$ then $\langle \text{stat-seq} \rangle$ else $\langle \text{stat-seq} \rangle$ endif ; |
| $\langle \text{stat} \rangle$ | while $\langle \text{expr} \rangle$ do $\langle \text{stat-seq} \rangle$ enddo ; |
| $\langle \text{stat} \rangle$ | for id := $\langle \text{expr} \rangle$ to $\langle \text{expr} \rangle$ do $\langle \text{stat-seq} \rangle$ enddo ; |
| $\langle \text{stat} \rangle$ | $\langle \text{opt-prefix} \rangle$ id ($\langle \text{expr-list} \rangle$) ; |
| $\langle \text{stat} \rangle$ | break ; |
| $\langle \text{stat} \rangle$ | return $\langle \text{expr} \rangle$; |
| $\langle \text{expr} \rangle$ | $\langle \text{expr} \rangle \langle \text{binary-operator} \rangle \langle \text{expr} \rangle$ |
| $\langle \text{expr} \rangle$ | $\langle \text{const} \rangle$ |
| $\langle \text{expr} \rangle$ | $\langle \text{lvalue} \rangle$ |
| $\langle \text{expr} \rangle$ | - $\langle \text{expr} \rangle$ |
| $\langle \text{expr} \rangle$ | ($\langle \text{expr} \rangle$) |
| $\langle \text{binary-operator} \rangle$ | * |
| $\langle \text{binary-operator} \rangle$ | / |
| $\langle \text{binary-operator} \rangle$ | + |
| $\langle \text{binary-operator} \rangle$ | - |
| $\langle \text{binary-operator} \rangle$ | = |

| | |
|--|---|
| $\langle \text{binary-operator} \rangle$ | $<$ |
| $\langle \text{binary-operator} \rangle$ | $>$ |
| $\langle \text{binary-operator} \rangle$ | $<=$ |
| $\langle \text{binary-operator} \rangle$ | $>=$ |
| $\langle \text{binary-operator} \rangle$ | $<>$ |
| $\langle \text{binary-operator} \rangle$ | $\&$ |
| $\langle \text{binary-operator} \rangle$ | $ $ |
| $\langle \text{binary-operator} \rangle$ | $:=$ |
| $\langle \text{opt-prefix} \rangle$ | $\langle \text{lvalue} \rangle :=$ |
| $\langle \text{opt-prefix} \rangle$ | NULL |
| $\langle \text{const} \rangle$ | INTLIT |
| $\langle \text{const} \rangle$ | STRLIT |
| $\langle \text{const} \rangle$ | nil |
| $\langle \text{expr-list} \rangle$ | $\langle \text{expr} \rangle \langle \text{expr-list-tail} \rangle$ |
| $\langle \text{expr-list} \rangle$ | NULL |
| $\langle \text{expr-list-tail} \rangle$ | $, \langle \text{expr} \rangle \langle \text{expr-list-tail} \rangle$ |
| $\langle \text{expr-list-tail} \rangle$ | NULL |
| $\langle \text{lvalue} \rangle$ | id $\langle \text{lvalue-tail} \rangle$ |
| $\langle \text{lvalue-tail} \rangle$ | $[\langle \text{expr} \rangle] \langle \text{lvalue-tail} \rangle$ |
| $\langle \text{lvalue-tail} \rangle$ | NULL |

Modified Grammar:

| symbol | rule |
|---|--|
| $\langle \text{tiger-program} \rangle$ | let $\langle \text{declaration-segment} \rangle$ in $\langle \text{stat-seq} \rangle$ end |
| $\langle \text{declaration-segment} \rangle$ | $\langle \text{type-declaration-list} \rangle \langle \text{var-declaration-list} \rangle \langle \text{funct-declaration-list} \rangle$ |
| $\langle \text{type-declaration-list} \rangle$ | $\langle \text{type-declaration} \rangle \langle \text{type-declaration-list} \rangle$ |
| $\langle \text{type-declaration-list} \rangle$ | NULL |
| $\langle \text{var-declaration-list} \rangle$ | $\langle \text{var-declaration} \rangle \langle \text{var-declaration-list} \rangle$ |
| $\langle \text{var-declaration-list} \rangle$ | NULL |
| $\langle \text{funct-declaration-list} \rangle$ | $\langle \text{funct-declaration} \rangle \langle \text{funct-declaration-list} \rangle$ |
| $\langle \text{type-declaration} \rangle$ | type id = $\langle \text{type} \rangle$; |
| $\langle \text{type} \rangle$ | $\langle \text{type-id} \rangle$ |
| $\langle \text{type} \rangle$ | array [INTLIT] of $\langle \text{type-dim} \rangle$ of $\langle \text{type-id} \rangle$ |
| $\langle \text{type-dim} \rangle$ | [INTLIT] $\langle \text{type-dim} \rangle$ |
| $\langle \text{type-dim} \rangle$ | NULL |
| $\langle \text{type-id} \rangle$ | id |
| $\langle \text{var-declaration} \rangle$ | var $\langle \text{id-list} \rangle$: $\langle \text{type-id} \rangle$ $\langle \text{optional-init} \rangle$; |
| $\langle \text{id-list} \rangle$ | id $\langle \text{id-list-tail} \rangle$ |
| $\langle \text{id-list-tail} \rangle$ | , id $\langle \text{id-list-tail} \rangle$ |
| $\langle \text{id-list-tail} \rangle$ | NULL |
| $\langle \text{optional-init} \rangle$ | $:= \langle \text{const} \rangle$ |
| $\langle \text{optional-init} \rangle$ | NULL |
| $\langle \text{funct-declaration} \rangle$ | function id ($\langle \text{param-list} \rangle$) $\langle \text{ret-type} \rangle$ begin $\langle \text{stat-seq} \rangle$ end ; |
| $\langle \text{param-list} \rangle$ | $\langle \text{param} \rangle \langle \text{param-list-tail} \rangle$ |
| $\langle \text{param-list} \rangle$ | NULL |
| $\langle \text{param-list-tail} \rangle$ | , $\langle \text{param} \rangle \langle \text{param-list-tail} \rangle$ |
| $\langle \text{param-list-tail} \rangle$ | NULL |
| $\langle \text{ret-type} \rangle$ | : $\langle \text{type-id} \rangle$ |
| $\langle \text{ret-type} \rangle$ | NULL |
| $\langle \text{param} \rangle$ | id : $\langle \text{type-id} \rangle$ |
| $\langle \text{stat-seq} \rangle$ | $\langle \text{stat} \rangle \langle \text{stat-seq-tail} \rangle$ |
| $\langle \text{stat-seq-tail} \rangle$ | $\langle \text{stat} \rangle \langle \text{stat-seq-tail} \rangle$ |

| | |
|---|---|
| $\langle \text{stat-seq-tail} \rangle$ | NULL |
| $\langle \text{stat} \rangle$ | if $\langle \text{expr} \rangle$ then $\langle \text{stat-seq} \rangle$ $\langle \text{stat-if-tail} \rangle$ |
| $\langle \text{stat} \rangle$ | while $\langle \text{expr} \rangle$ do $\langle \text{stat-seq} \rangle$ enddo ; |
| $\langle \text{stat} \rangle$ | for id := $\langle \text{expr} \rangle$ to $\langle \text{expr} \rangle$ do $\langle \text{stat-seq} \rangle$ enddo ; |
| $\langle \text{stat} \rangle$ | $\langle \text{lvalue} \rangle := \langle \text{stat-assign} \rangle$ |
| $\langle \text{stat} \rangle$ | id ($\langle \text{expr-list} \rangle$) ; |
| $\langle \text{stat} \rangle$ | break ; |
| $\langle \text{stat} \rangle$ | return $\langle \text{expr} \rangle$; |
| $\langle \text{stat-if-tail} \rangle$ | else $\langle \text{stat-seq} \rangle$ endif ; |
| $\langle \text{stat-if-tail} \rangle$ | endif ; |
| $\langle \text{stat-assign} \rangle$ | $\langle \text{expr} \rangle$; |
| $\langle \text{stat-assign} \rangle$ | id ($\langle \text{expr-list} \rangle$) ; |
| $\langle \text{opt-prefix} \rangle$ | $\langle \text{lvalue} \rangle :=$ |
| $\langle \text{opt-prefix} \rangle$ | NULL |
| $\langle \text{expr} \rangle$ | $\langle \text{orexpr} \rangle \langle \text{orop} \rangle \langle \text{expr} \rangle$ |
| $\langle \text{expr} \rangle$ | $\langle \text{orexpr} \rangle$ |
| $\langle \text{orop} \rangle$ | |
| $\langle \text{orexpr} \rangle$ | $\langle \text{andexpr} \rangle \langle \text{andop} \rangle \langle \text{orexpr} \rangle$ |
| $\langle \text{orexpr} \rangle$ | $\langle \text{andexpr} \rangle$ |
| $\langle \text{andop} \rangle$ | & |
| $\langle \text{andexpr} \rangle$ | $\langle \text{compare} \rangle \langle \text{compop} \rangle \langle \text{and} \rangle$ |
| $\langle \text{andexpr} \rangle$ | $\langle \text{compare} \rangle$ |
| $\langle \text{const} \rangle$ | INTLIT |
| $\langle \text{const} \rangle$ | STRLIT |
| $\langle \text{const} \rangle$ | nil |
| $\langle \text{compop} \rangle$ | = |
| $\langle \text{compop} \rangle$ | <> |
| $\langle \text{compop} \rangle$ | > |
| $\langle \text{compop} \rangle$ | < |
| $\langle \text{compop} \rangle$ | >= |
| $\langle \text{compop} \rangle$ | <= |
| $\langle \text{compare} \rangle$ | $\langle \text{term} \rangle \langle \text{addop} \rangle \langle \text{compare} \rangle$ |
| $\langle \text{compare} \rangle$ | $\langle \text{term} \rangle$ |
| $\langle \text{addop} \rangle$ | + |
| $\langle \text{addop} \rangle$ | - |
| $\langle \text{term} \rangle$ | $\langle \text{factor} \rangle \langle \text{mulop} \rangle \langle \text{term} \rangle$ |
| $\langle \text{term} \rangle$ | $\langle \text{factor} \rangle$ |
| $\langle \text{mulop} \rangle$ | * |
| $\langle \text{mulop} \rangle$ | / |
| $\langle \text{factor} \rangle$ | $\langle \text{unaryminus} \rangle$ |
| $\langle \text{factor} \rangle$ | - $\langle \text{unaryminus} \rangle$ |
| $\langle \text{unaryminus} \rangle$ | ($\langle \text{expr} \rangle$) |
| $\langle \text{unaryminus} \rangle$ | $\langle \text{const} \rangle$ |
| $\langle \text{unaryminus} \rangle$ | $\langle \text{lvalue} \rangle$ |
| $\langle \text{opt-prefix} \rangle$ | $\langle \text{lvalue} \rangle :=$ |
| $\langle \text{opt-prefix} \rangle$ | NULL |
| $\langle \text{const} \rangle$ | INTLIT |
| $\langle \text{const} \rangle$ | STRLIT |
| $\langle \text{const} \rangle$ | nil |
| $\langle \text{expr-list} \rangle$ | $\langle \text{expr} \rangle \langle \text{expr-list-tail} \rangle$ |
| $\langle \text{expr-list} \rangle$ | NULL |
| $\langle \text{expr-list-tail} \rangle$ | , $\langle \text{expr} \rangle \langle \text{expr-list-tail} \rangle$ |
| $\langle \text{expr-list-tail} \rangle$ | NULL |

| | |
|--------------------------------------|--|
| $\langle \text{lvalue} \rangle$ | id $\langle \text{lvalue-tail} \rangle$ |
| $\langle \text{lvalue-tail} \rangle$ | [$\langle \text{expr} \rangle$] $\langle \text{lvalue-tail} \rangle$ |
| $\langle \text{lvalue-tail} \rangle$ | NULL |

Tokens for the parser table:

PLUS, MINUS, MULT, DIV, EQ, LPAREN, RPAREN, COMMA, AND, OR, LBRACK, RBRACK, SEMI, NEQ, LESSEREQ, LESS, GREATEREQ, GREATER, ASSIGN, COLON, INTLIT, ID, STRLIT, RETURN, ARRAY, BREAK, DO, ELSE, END, FOR, FUNC, IF, IN, LET, NIL, OF, THEN, TO, TYPE, VAR, WHILE, ENDIF, BEGIN, ENDDO