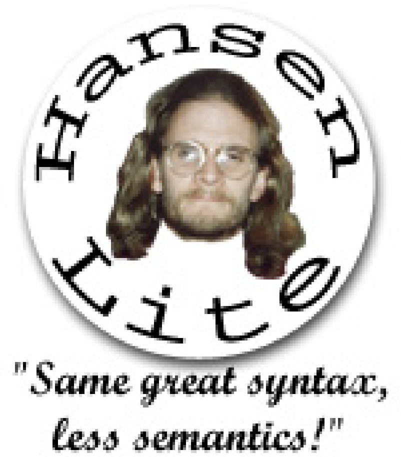
HansenLite — Grammar and Lexical Elements

CSIS 480 — Principles of Compiler Construction



1 Grammar for *HansenLite*

statement →
identifier assignment_operator *expression*
| if *boolean_expression* then *statement else_clause*
| while *boolean_expression* do *statement*
| print *print_expression*
| begin *statement_list* end
| variable identifier

else_clause →
else *statement*
| ϵ

statement_list →
statement separated_list

separated_list →
statement_separator *statement separated_list*
| ϵ

print_expression →
expression
| string_const

boolean_expression →
expression relational_operator *expression*

expression →
term addition

addition →
additive_operator *term addition*
| ϵ

term →
factor multiplication

multiplication →
multiplicative_operator *factor multiplication*
| ϵ

factor →
left_paren *expression* right_paren
| identifier
| number
| *signed_term*

signed_term →
additive_operator *term*

Note: variables must be declared before they are referenced and all variables are of type *number*.

2 Tokens

- variable
- identifier
- number
- print
- if
- then
- else
- while
- do
- begin
- end
- string_const
- multiplicative_operator
- relational_operator
- assignment_operator
- additive_operator
- statement_separator
- left_paren
- right_paren
- whitespace
- comment

3 Lexemes

delimiter → space | tab | newline

lowercase_letter → a | b | ... z

uppercase_letter → A | B | ... Z

letter → *lowercase_letter* | *uppercase_letter*

digit → 0 | 1 | ... 9

identifier → *letter* (*letter* | *digit* | _)*

number → *digit*$^+$

string_const → ”.*”

variable → variable

print → print

if → if

then → then

else → else

while → while

do → do

begin → begin

end → end

additive_operator → +|−

relational_operator →< | <= | <> | = | > | >=

multiplicative_operator → ∗|/

assignment_operator → :=

statement_separator → ;

left_paren → (

right_paren →)

whitespace → delimiter$^+$

comment → {.*}

Note: identifiers *are* case-sensitive and all reserved keywords are lowercase.