* 1. non-vowel = [b-d f-h j-n p-t v-z]

string = non-vowel\* a+ non-vowel\* e+ non-vowel\* I+ non-vowel\* o+ non-vowel\* u+ non-vowel\*

* 1. (.\* ‘ .\* ‘ .\*)\*
  2. b\*(a\*b)\*a\*
  3. T = (aabb+abab+abba+bbaa+baba+baab+bb+aa)\*

string = T\*bT\*

1. 1. E -> DAD | D | A

A -> ‘ E ‘

D ->aD | bD | a | b

* 1. The language is not a regular language.

Proof: Assume the language is regular, then the grammar can be encoded into a finite-state automaton. We can construct a string that consists of n left and n right quotation marks, of which the length is longer than the number of states in the FA. That means that some states in the FA must be repeated let W = xy^iz, where |xy| <= n. We know that y consists of only left parenthesis. Let y^i be the substring that can be accepted by the repeated states in FA, then xy^iz can be accepted by the FA for (i >= 0). We can choose a arbitrary positive integer for i and the string will still get accepted by the finite-state automata. However, there will be more left parentheses in the string than the right parentheses. Thus, there is a contradiction. Thus, the assumption that the language is regular cannot be held.

1. 1. Because of strong typing in C programming language, it is a not context-free language. All of its variables’ types would have to be determined in compile-time. Its grammar only allows an expression to be derived in a certain context, which is in the format:

αAβ -> αYβ.

Assume we have context-free grammar

T -> SE

S -> int x; | int y;

E -> x = 1; | y = 1;

Then the following illegal C code could be matched by the grammar:

int x;

y = 1;

* 1. it is not ambiguous, since the output of parser is the parse tree. If a string generates more than one possible parse tree. Then semantic analysis could have more than one possible evaluation of the same string, which is not allowed in C.



G1: ambiguous

“aa” can be generated by

S S

/ \ / \

a S S a

| |

a a

G2: ambiguous

“abab” can be generated by

S S

/ \ / \

aS bS aS bS

/ \ | | / \

bS aS ε ε aS bS

| | | |

ε ε ε ε

G3: ambiguous

“a + aa + a” can be generated by

R R

/ \ / \

R + R R R

| / \. / \ / \

a R + R R+RR+R

/ \ | | | | |

R R a a a a a

| |

a a

G1:

S –> aS | ε

G2:

S -> aTbR | bTaR

R -> S | ε

T -> S | ε

G3:

R -> R + S | S

S -> S\* | T

T -> TA | A

A -> a | b | c

* 1. Digits = {1+2+3+4+5+6+7+8+9}

Integers = Digits (Digits + 0)\*+ 0