Context-Free Grammar for a LTL wff

Here, ‘Next’, ‘Eventually’, ‘Always’, ‘Until’, and ‘Weak Until’ are represented by the letters ‘N’, ‘E’, ‘A’, ‘U’, and ‘W’.

Alphabet = { ‘0’, ‘1’, …, ‘9’, ‘p’, ‘(‘, ‘)’,

‘T’, ‘F’, ‘S’,

‘~’, ‘N’, ‘E’, ‘A’,

‘v’, ‘&’, ‘=’, ‘>’, ‘U’, ‘W’ }

Digit -> ‘0’ | ‘1’ | … |’9’

Num -> Digit Num | Digit

Prop\_var -> ‘p’ Num

Prop\_cons -> ‘T’ | ‘F’

Unary\_Prop\_conn -> ‘~’

Binary\_Prop\_conn -> ‘v’ | ‘&’ | ‘=’ | ‘>’

Temp\_cons -> ‘S’

Unary\_Temp\_conn -> ‘N’ | ‘E’ | ‘A’

Binary\_Temp\_conn -> ‘U’ | ‘W’

Wff -> Prop\_var | Prop\_cons | Temp\_cons

| Unary\_Prop\_conn Wff | Unary\_Temp\_conn Wff

| ‘(‘ Wff Binary\_Prop\_conn Wff ‘)’ | ‘(‘ Wff Binary\_Temp\_conn Wff ‘)’

C-code implementation of Prop\_var

bool Prop\_var(string s, int len){

if(s[0] == ‘p’){

return Num(s[1 : len - 1], len - 1);

}

}