Context-Free Grammar for a MLTL wff.

Here, ‘Eventually’, ‘Always’, ‘Until’, and ‘Weak Until’ are represented by the letters ‘F’, ‘G’, ‘U’, and ‘W’.

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

‘T’, ‘!’,

‘~’, ‘F’, ‘G’,

‘v’, ‘&’, ‘=’, ‘>’, ‘U’, ‘R’ }

Digit -> ‘0’ | ‘1’ | … |’9’

Num -> Digit Num | Digit

Interval -> ‘[’ Num ‘:’ Num ‘]’

Prop\_var -> ‘p’ Num

Prop\_cons -> ‘T’ | ‘!’

Unary\_Prop\_conn -> ‘~’

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

Assoc\_Prop\_conn -> ‘v’ | ‘&’ | ‘=’

Array\_entry -> Wff ‘,’ Array\_entry | Wff

Unary\_Temp\_conn -> ‘F’ | ‘G’

Binary\_Temp\_conn -> ‘U’ | ‘R’

Wff -> Prop\_var | Prop\_cons

| Unary\_Prop\_conn Wff

| Unary\_Temp\_conn Interval Wff

| ‘(‘ Assoc\_Prop\_conn ‘[‘ Array\_entry ‘]’ ‘)’

| ‘(‘ Wff Binary\_Prop\_conn Wff ‘)’

| ‘(‘ Wff Binary\_Temp\_conn Interval Wff ‘)

(a & (b & c))

(& [a, b, c] )

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);

}

}