Parsing anomalies

If an expression is well formed, sometimes it parses with an exception raised. This results from pairs of operators preceding variables only and not expressions. For example %(#(%(p=p)))=*(p=p)*, rewrite the "#(%" as ((r=%(p=p))>%#r)=*(p=p)*.

If this happens, we suggest a work around by embedding the expression as an antecedent (or consequent) with a connective to tautology such as "*p=p*". T tautology is the designated proof value, F contradiction, N value for truth, and C value for falsity.

The resulting truth table of 16-values is presented horizontally as row-major. We found these examples with work arounds.

1. #(~(%~p>~%r)) > #(~(#r>%p)) ; ngp (not good parse)
   #(~(%~p>~%r) > ~(#r>%p))=*(p=p)* ; then extract the truth table for the antecedent as: `NNNN NNNN NNNN NNNN`

2. # ~(%~p>~%r) > #~(#r>%p) ; ngp [modal operator with negagtion preceding parenthetical literal]
   #(~(%~p>~%r) > ~(#r>%p))=*(p=p)* ; then extract the truth table for the antecedent as: `NNNN NNNN NNNN NNNN`

3. (~(#(%~p>~%r))) > (~(#(#r>%p))) ; ngp
   (~(#(%~p>~%r) > #(#r>%p)))=*(p=p)*; then extract the truth table for the antecedent as: `FFFF FFFF FFFF FFFF`

4. ~(#(%~p>~%r)) > ~(#(#r>%p)) ; ngp
   ~(#(%~p>~%r) > #(#r>%p))=*(p=p)* ; then extract the truth table for the antecedent as: `FFFF FFFF FFFF FFFF`

5. ~ #(%~p>~%r) > ~#(#r>%p) ; ngp
   ~(#(%~p>~%r) > #(#r>%p))=*(p=p)* ; then extract the truth table for the antecedent as: `FFFF FFFF FFFF FFFF`

6. # (%~p>~%r) > #(#r>%p) ; validated as tautologous: `TTTT TTTT TTTT TTTT`
   #((%~p>~%r) > (#r>%p))=*(p=p)* ; however, *not* distributing the quantifier produces: `NNNN NNNN NNNN NNNN`