Parser Grammar

The below is the documentation related to the syntax of Differential Dynamic Logic (DL) and Relational Dynamic Logic (Rel DL). Please refer it and create the input files to pass to the tool and generate Abstract Syntax Trees (AST) or KeYmaeraX outputs.

Differential Dynamic Logic Grammar

formula:

- term COMPARISON_OPERATORS term
- BOOLEANS
- ! formula
- formula && formula
- formula || formula
- formula -> formula
- formula <-> formula
- [program] formula
- << program >> formula

program:

- IDENTIFIER := term;
- IDENTIFIER := **;
- program ; program
- program ++ program
- {program} **
- ? formula;
- { IDENTIFIER_PRIME = term && formula }

term:

- IDENTIFIER
- NUMBER
- term BINARY_EXPRESSION_OPERATORS term
- (term)

Lexer Rules:

- IDENTIFIER_PRIME: [a-zA-Z][a-zA-Z0-9]* '

- IDENTIFIER: [a-zA-Z][a-zA-Z0-9]*
- NUMBER: [0-9]+.[0-9]+
- BOOLEANS: true | false
- COMPARISON_OPERATORS: == (or) != (or) <= (or) >= (or) < (or) >
- BINARY_EXPRESSION_OPERATORS: + (or) – (or) * (or) /
- COMMENTS: // -> Will be skipped

Relational Dynamic Logic Grammar

We have imported all the grammar rules from the Differential Dynamic Logic (DL) grammar. We have used program, term, IDENTIFIER, and IDENTIFIER_PRIME from the DL grammar.

relFormula:

- relTerm REL_DL_COMPARISON_OPERATORS relTerm
- !# relFormula
- relFormula &&# relFormula
- relFormula ||# relFormula
- relFormula ->># relFormula
- relFormula <->># relFormula
- [# relProgram]# relFormula
- <<# relProgram >>># relFormula

relProgram:

- IDENTIFIER :=# term
- IDENTIFIER :=# **
- relProgram ;# relProgram
- relProgram ++# relProgram
- {# relProgram }# **
- ?# relFormula
- {# IDENTIFIER_PRIME =# relTerm &&# relFormula }#
- (# program ,# program)#

relTerm:

- term PROGRAM_CONSIDERED

Lexer Rules:

- REL_DL_COMPARISON_OPERATORS: ==# (or) !=# (or) <=# (or) >=# (or) <# (or) >#

- PROGRAM_CONSIDERED: @L (or) @R