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