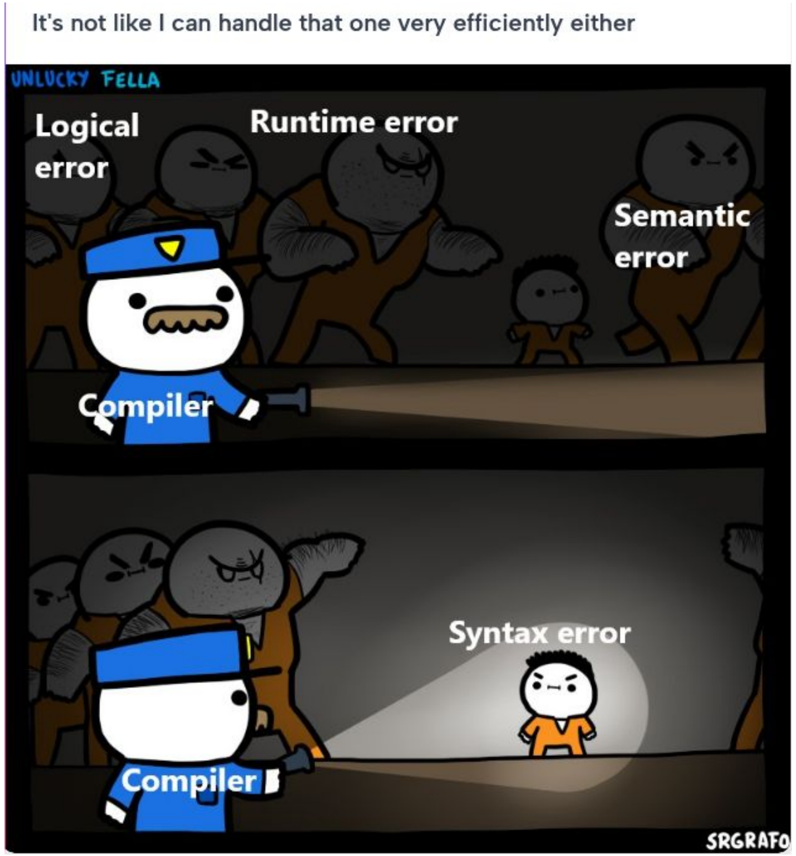
CSE 310 - Compiler Sessional
Syntax and Semantic Analysis

Nafis Tahmid

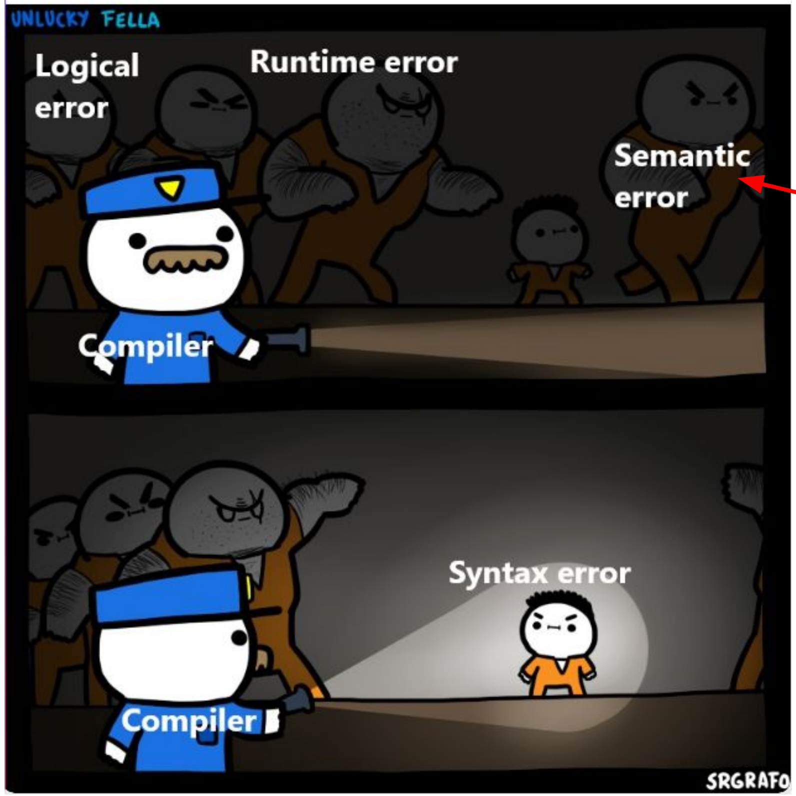
May 2025

What we do this time?


It's not like I can handle that one very efficiently either

What we do this time?

It's not like I can handle that one very efficiently either



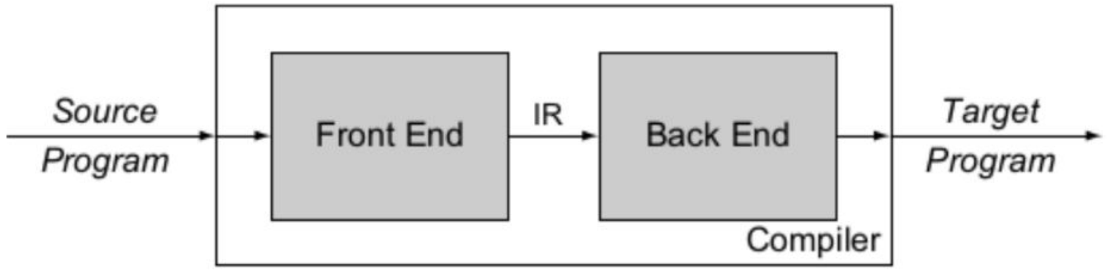
well, do not spare him either

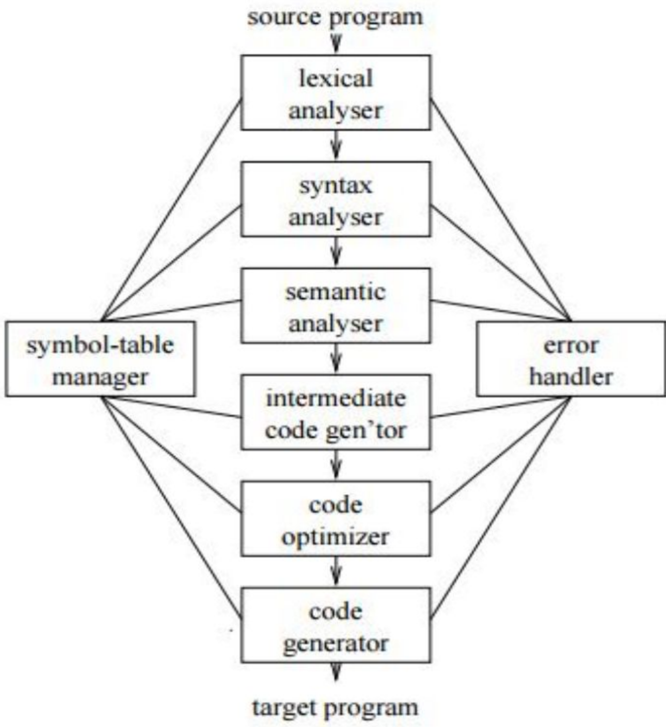
Syntax and Semantic Analysis

In the prev asm, we have constructed a lexical analyzer to generate token streams.

In this assignment, we will construct the last part of the front end of a compiler for a subset of the C language.

We will perform syntax analysis and semantic analysis with a grammar rule containing some rule actions.





Reporting errors with line number

Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'

Reporting errors with line number

```
Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'
```

"Well I don't know either"
Compiler: Error at line 308
Me: "What? How? My code only has 40 lines
Compiler:

Reporting errors with line number

```
Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of
FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of
FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'
```

Reporting errors with line number

```
Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'
```

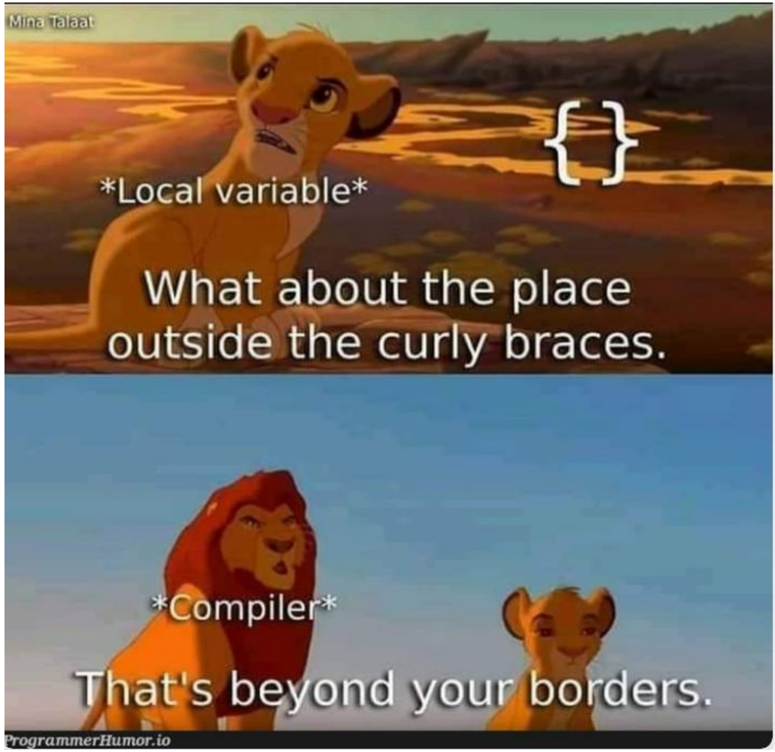
You all know the outcome
Me when my compiler asks me if I'm sure I want to continue with 800 warnings
I'll allow it.
made with mematic
ProgrammerHumor.io

Scope management (Usage of Grammars and Help of SymbolTable)

```
Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of
FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of
FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'
```

Scope management (Usage of Grammars and SymbolTable together)

Line# 14: Syntax error at expression of expression statement
Line# 15: Syntax error at expression of expression statement
Line# 17: Operands of modulus must be integers
Line# 18: Warning: possible loss of data in assignment of FLOAT to INT
Line# 20: Array subscript is not an integer
Line# 24: Conflicting types for 'i'
Line# 27: Undeclared variable 'j'
Line# 30: Syntax error at expression of expression statement
Line# 37: 'i' is not a function
Line# 44: Too few arguments to function 'mul_float'
Line# 45: Warning: possible loss of data in assignment of FLOAT to INT
Line# 47: Type mismatch for argument 1 of 'add_int'
Line# 47: Type mismatch for assignment operator
Line# 50: Void cannot be used in expression
Line# 55: Conflicting types for 'add_int'


Mina Talaat
{ }
Local variable
What about the place outside the curly braces.
Compiler
That's beyond your borders.
ProgrammerHumor.io

Ignore comments (but use them)

But in lexical analysis, we worked hard for comments...

But in lexical analysis, we worked hard for comments...

ANTLR

Download v4.13.2 · Dev Tools · Book · Doc · About ANTLR · Support · Bugs

What is ANTLR?

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.

Terence Parr is a tech lead at Google and until 2022 was a professor of data science / computer science at Univ. of San Francisco. He is the maniac behind ANTLR and has been working on language tools since 1989.

Check out Terence impersonating a machine learning droid: explained.ai

Quick Start

To try ANTLR immediately, jump to the *new* ANTLR Lab!

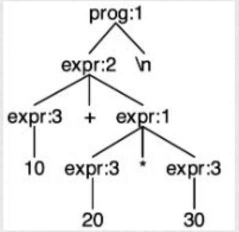
To install locally, use antlr4-tools, which installs Java and ANTLR if needed and creates `antlr4` and `antlr4-parse` executables:

```
$ pip install antlr4-tools
```

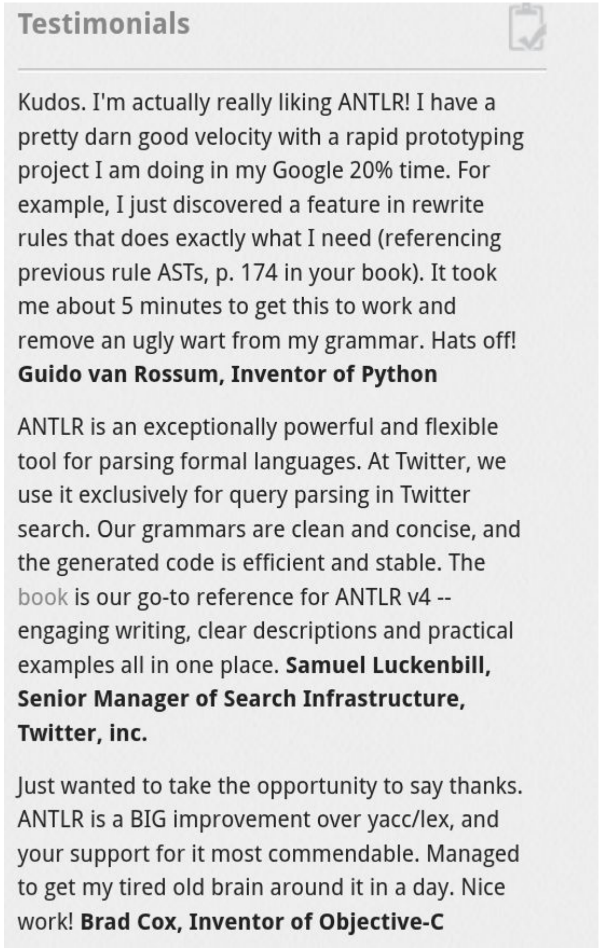
(Windows must add `..\LocalCache\local-packages\Python310\Scripts` to the PATH). See the Getting Started doc. Paste the following grammar into file `Expr.g4` and, from that directory, run the `antlr4-parse` command. Hit control-D on Unix (or control-Z on Windows) to indicate end-of-input. A window showing the parse tree will appear.

```
grammar Expr;
prog:   (expr NEWLINE)* ;
expr:   expr ('*'|'/') expr
    |   expr ('+'|'-') expr
    |   INT
    |   '(' expr ')'
    ;
NEWLINE : [\r\n]+ ;
INT     : [0-9]+ ;
```

```
$ antlr4-parse Expr.g4 prog -gui
10+20*30
^D
$ antlr4 Expr.g4  # gen code
$ ls ExprParser.java
ExprParser.java
```



The reference book

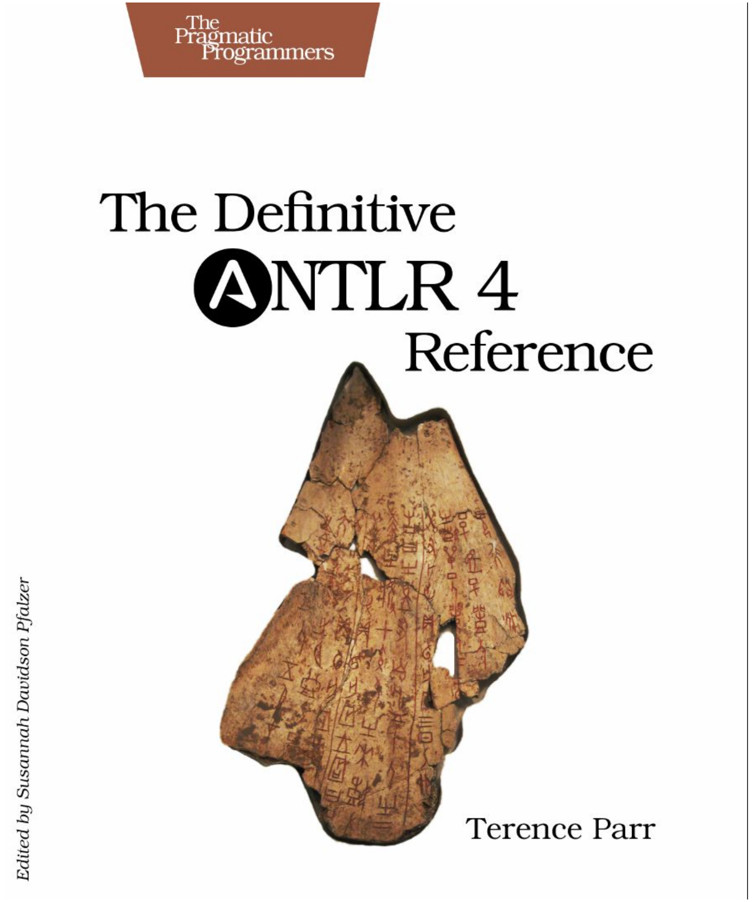


Testimonials

Kudos. I'm actually really liking ANTLR! I have a pretty darn good velocity with a rapid prototyping project I am doing in my Google 20% time. For example, I just discovered a feature in rewrite rules that does exactly what I need (referencing previous rule ASTs, p. 174 in your book). It took me about 5 minutes to get this to work and remove an ugly wart from my grammar. Hats off! **Guido van Rossum, Inventor of Python**

ANTLR is an exceptionally powerful and flexible tool for parsing formal languages. At Twitter, we use it exclusively for query parsing in Twitter search. Our grammars are clean and concise, and the generated code is efficient and stable. The book is our go-to reference for ANTLR v4 -- engaging writing, clear descriptions and practical examples all in one place. **Samuel Luckenbill, Senior Manager of Search Infrastructure, Twitter, inc.**

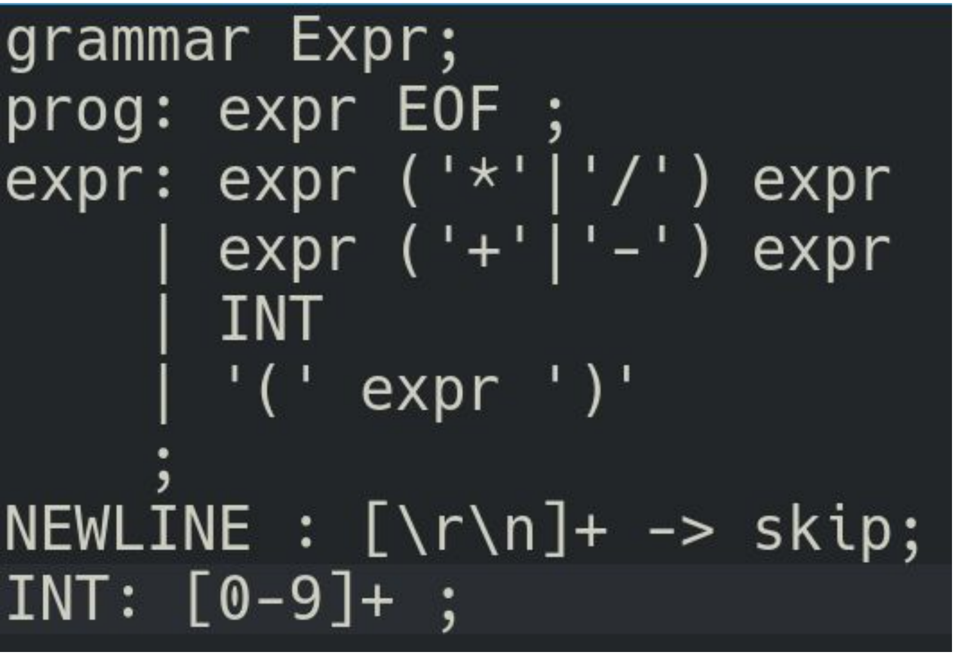
Just wanted to take the opportunity to say thanks. ANTLR is a BIG improvement over yacc/lex, and your support for it most commendable. Managed to get my tired old brain around it in a day. Nice work! **Brad Cox, Inventor of Objective-C**


The Pragmatic Programmers
The Definitive ANTLR 4 Reference
Terence Parr
Edited by Susannah Davidson Pfalzer

Language?

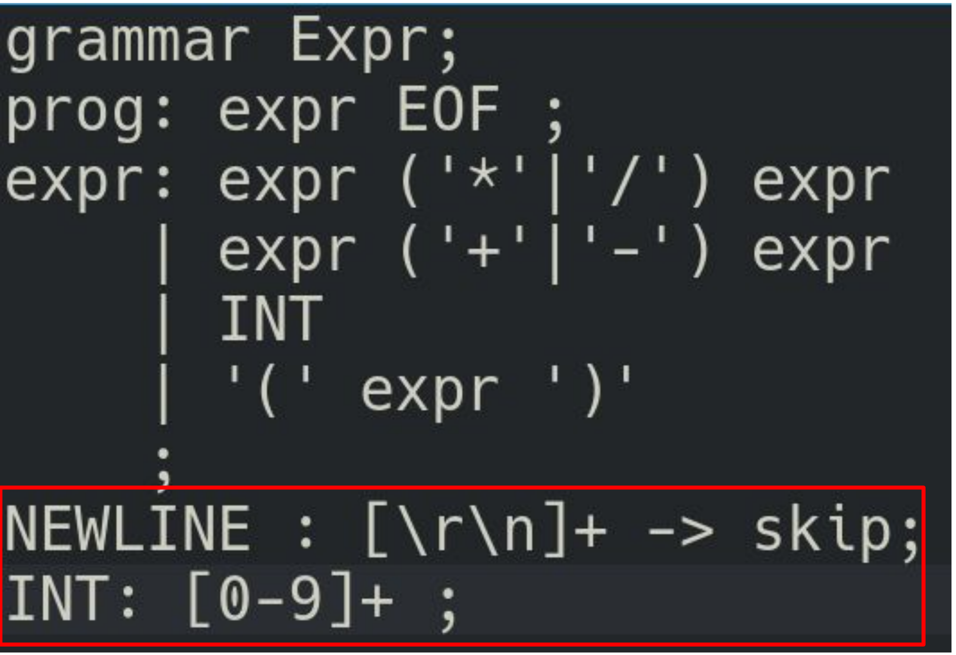
```
antlr4-skeletons
├── cpp
│   ├── C8086Lexer.g4
│   ├── C8086Parser.g4
│   ├── clean-script.sh
│   ├── Ctester.cpp
│   ├── input
│   │   ├── test.c
│   │   └── test_syntax_error.c
│   └── run-script.sh
└── java
    ├── C8086Lexer.g4
    ├── C8086Parser.g4
    ├── clean-script.sh
    ├── input
    │   ├── test.c
    │   └── test_syntax_error.c
    ├── Main.java
    └── run-script.sh
```

Let's see installation and do some parsing



```
grammar Expr;
prog: expr EOF ;
expr: expr ('*'|'/') expr
    | expr ('+'|'-') expr
    | INT
    | '(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

Let's see installation and do some parsing

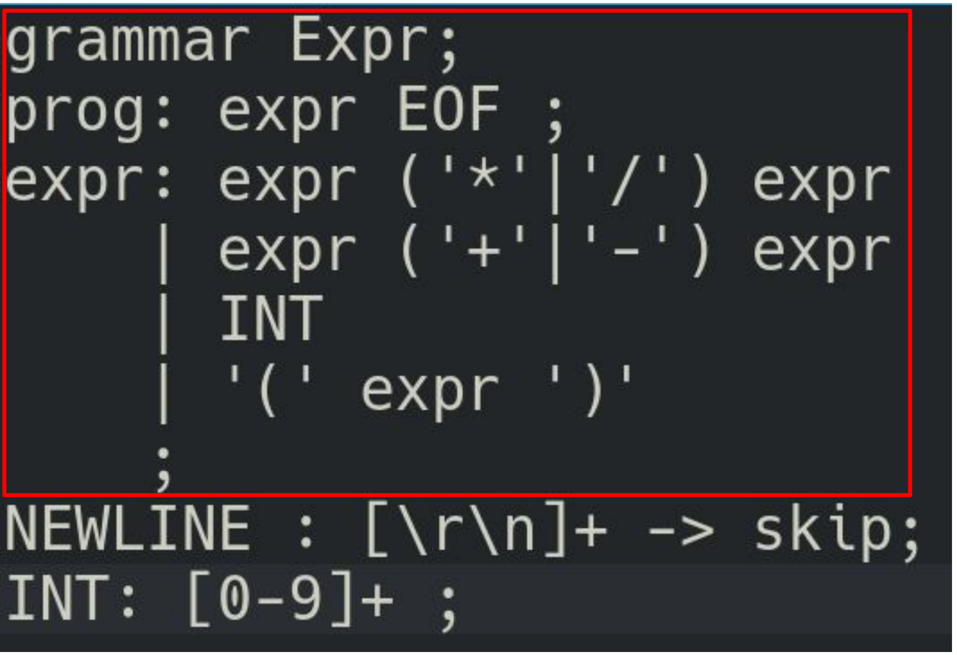


```
grammar Expr;
prog: expr EOF ;
expr: expr ('*'|'/') expr
    | expr ('+'|'-') expr
    | INT
    | '(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

Lexer

directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing

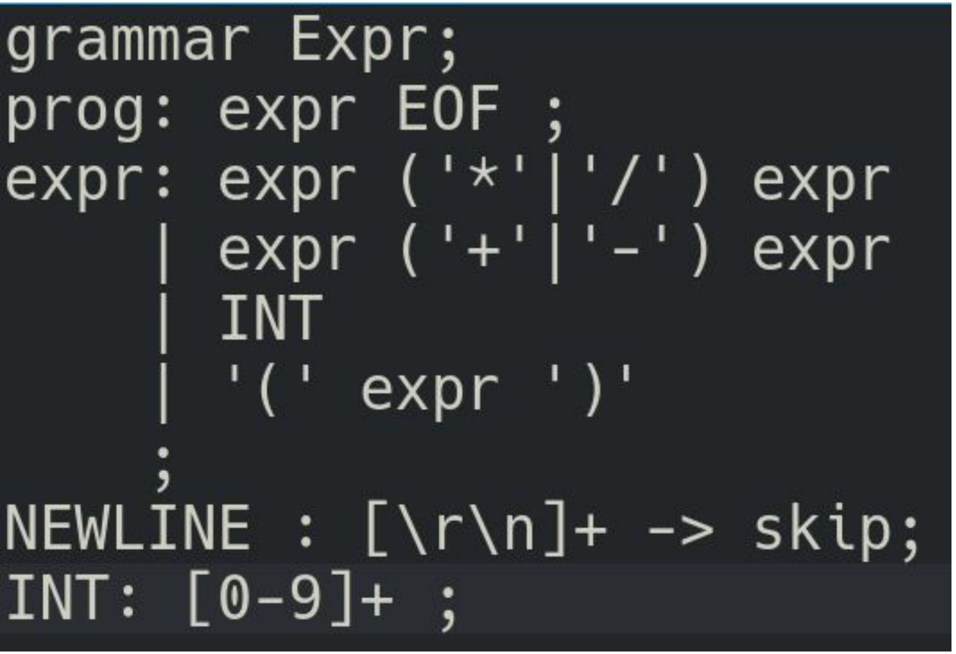


```
grammar Expr;
prog: expr EOF ;
expr: expr ('*'|'/') expr
    | expr ('+'|'-') expr
    | INT
    | '(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

Parser

directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing

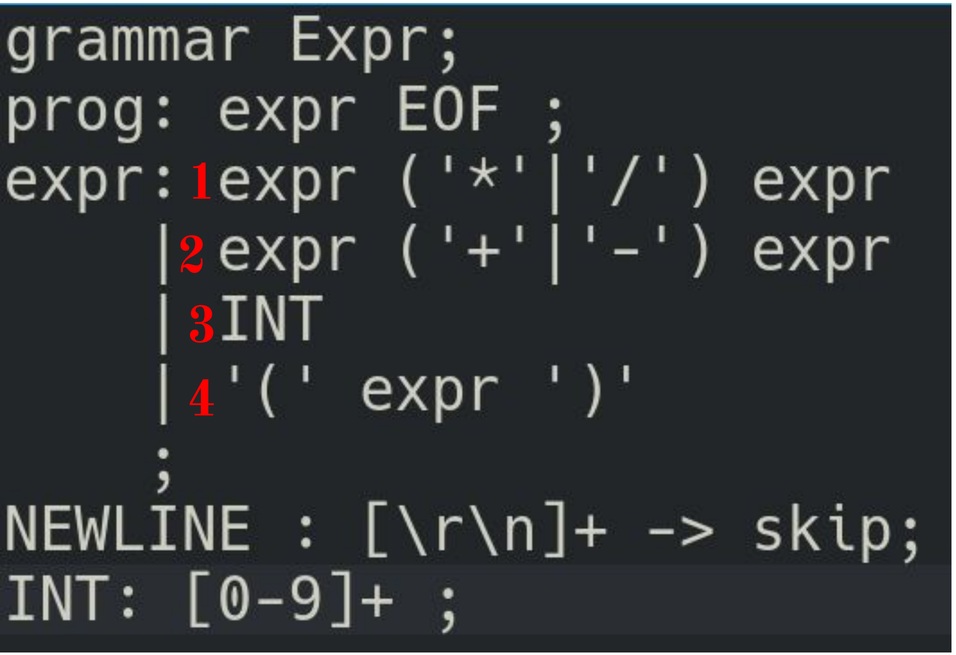


```
grammar Expr;
prog: expr EOF ;
expr: expr ('*'|'/') expr
    | expr ('+'|'-') expr
    | INT
    | '(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$10 + 20 * 30$

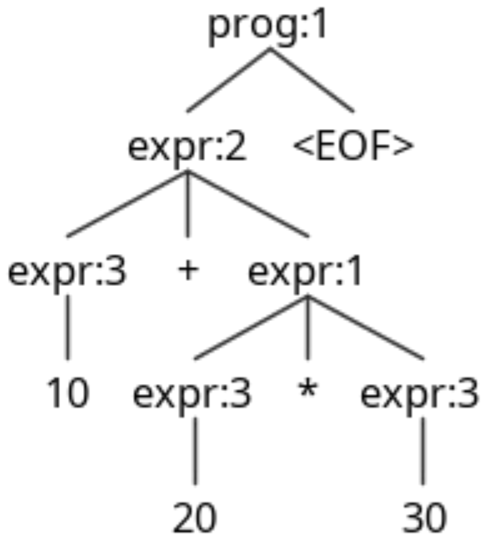
directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing



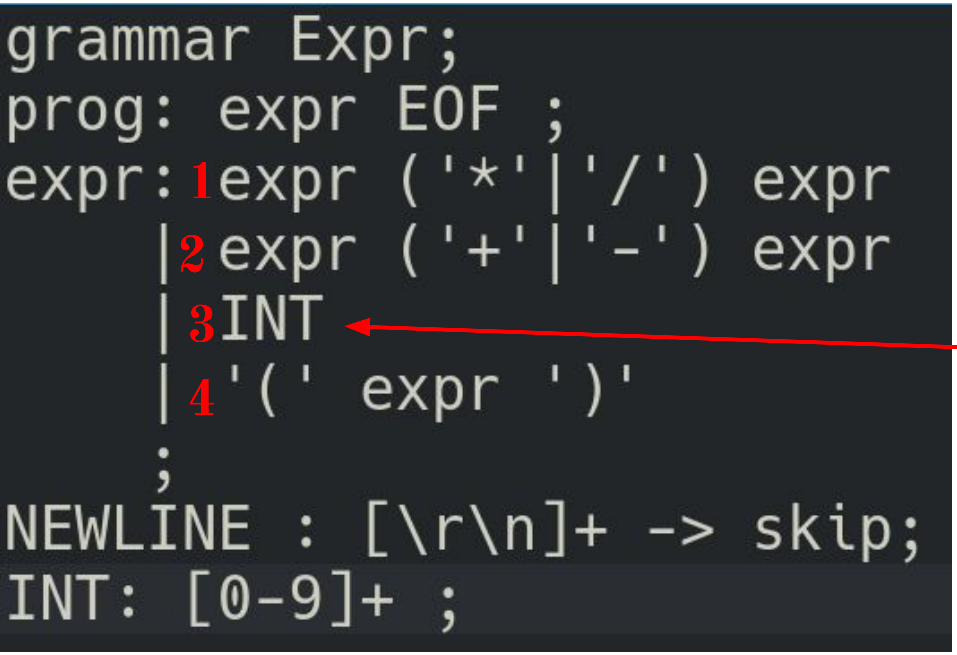
```
grammar Expr;
prog: expr EOF ;
expr:1expr ('*'|'/') expr
    |2expr ('+'|'-') expr
    |3INT
    |4'(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$$10 + 20 * 30$$



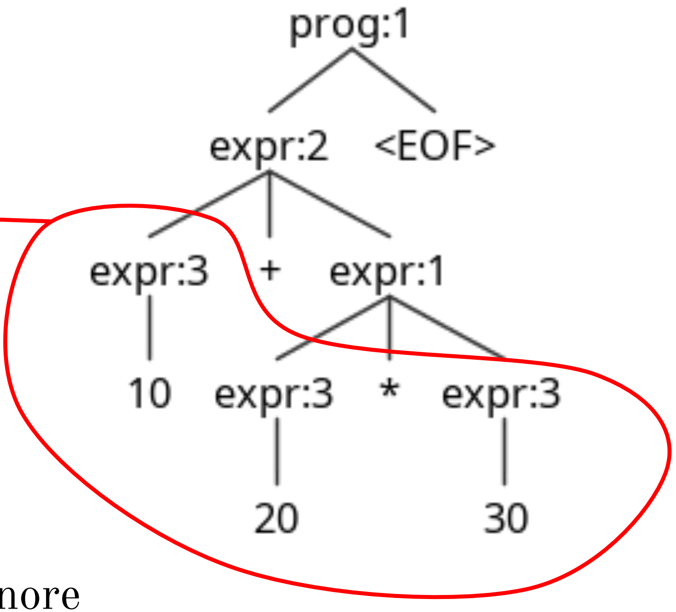
directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing



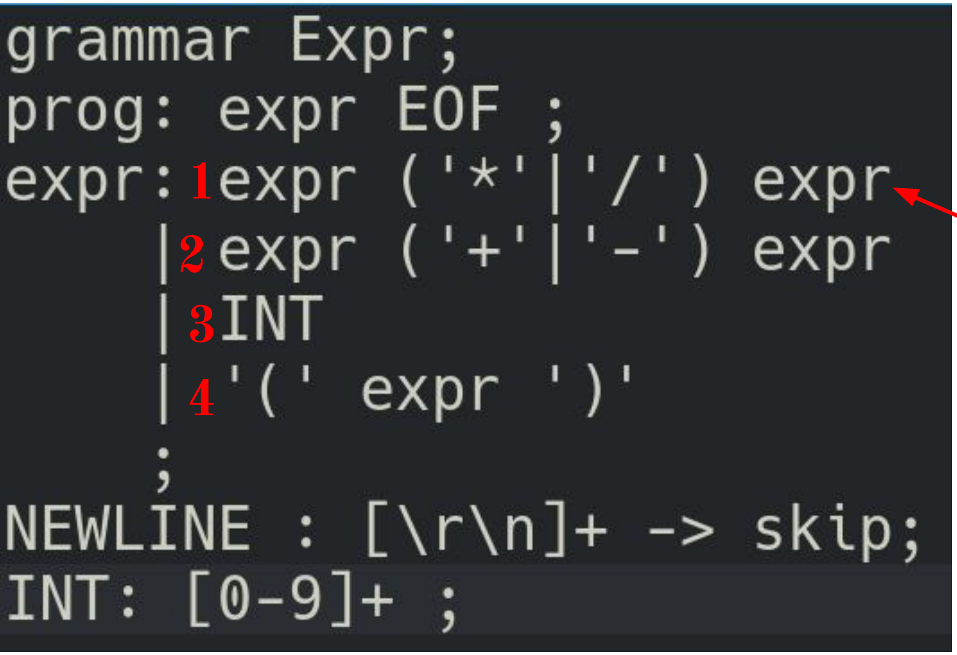
```
grammar Expr;
prog: expr EOF ;
expr:1expr ('*'|'/') expr
     |2expr ('+'|'-') expr
     |3INT
     |4'(' expr ')'
     ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$10 + 20 * 30$



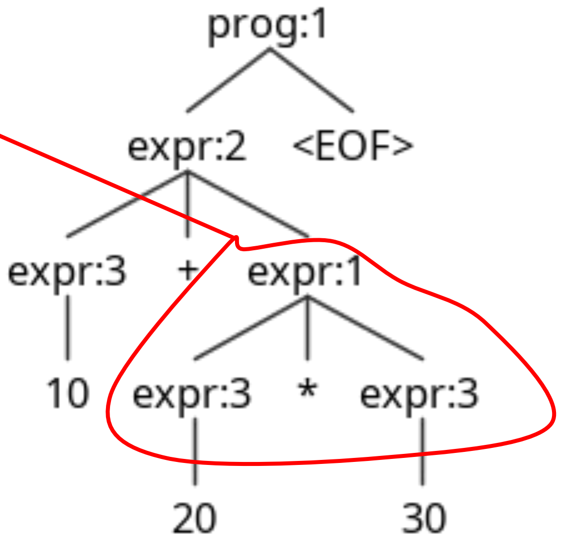
directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing



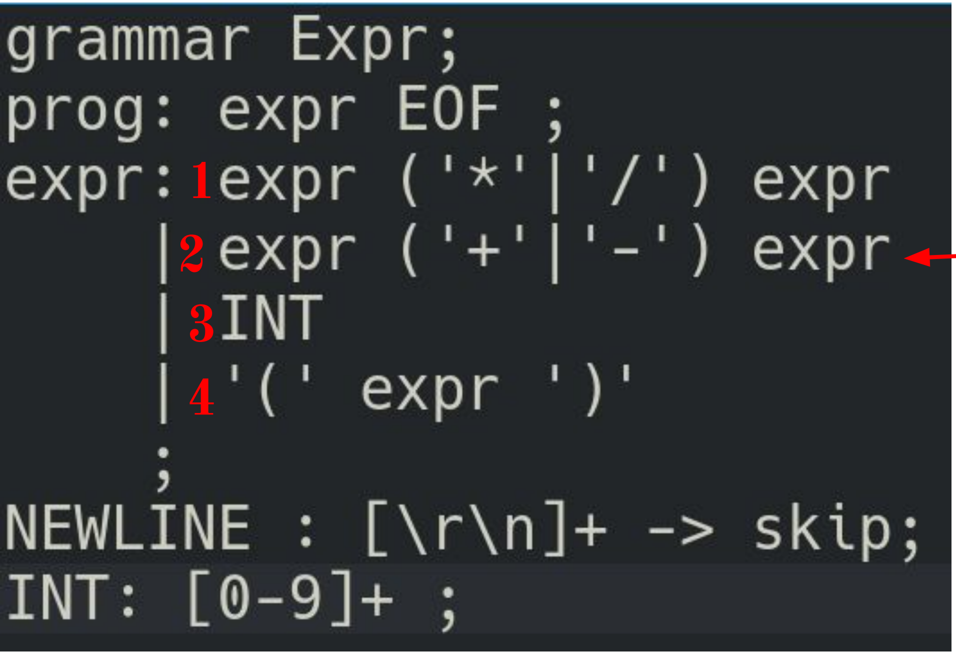
```
grammar Expr;
prog: expr EOF ;
expr:1expr ('*'|'/') expr
    |2expr ('+'|'-') expr
    |3INT
    |4'(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$$10 + 20 * 30$$



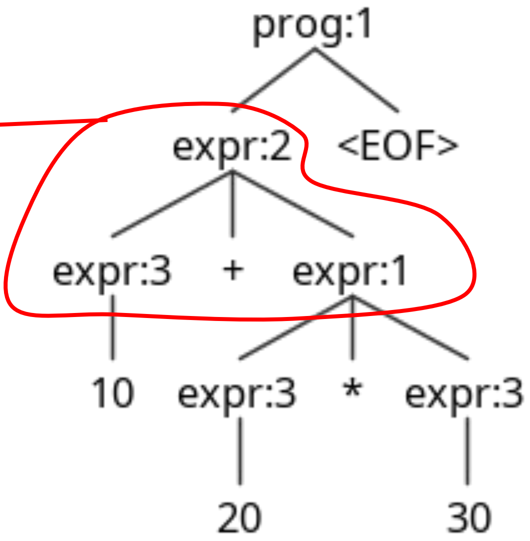
directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing



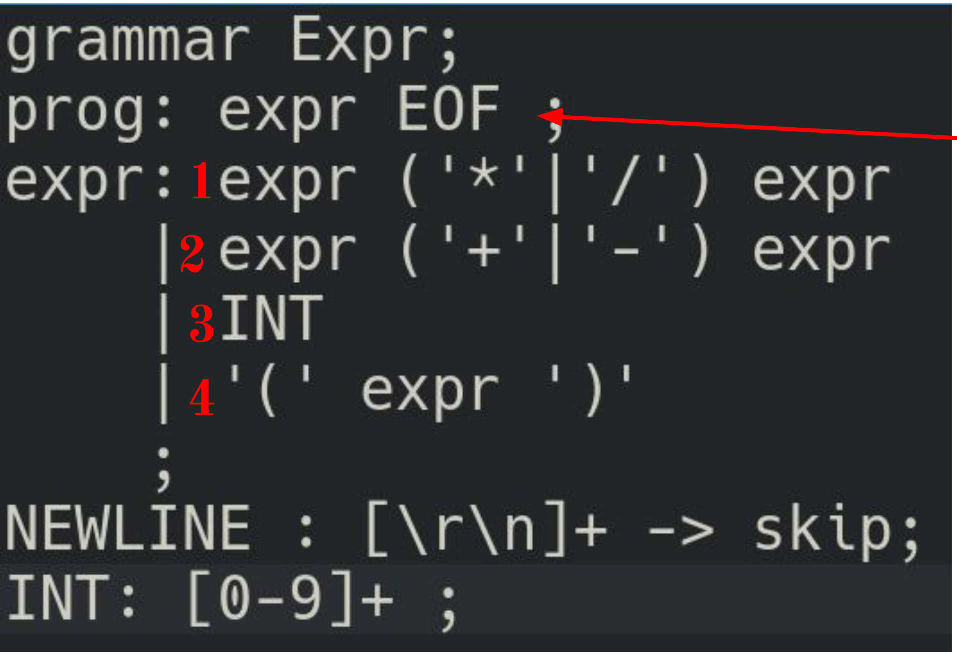
```
grammar Expr;
prog: expr EOF ;
expr:1expr ('*'|'/') expr
    |2expr ('+'|'-') expr
    |3INT
    |4'(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$$10 + 20 * 30$$



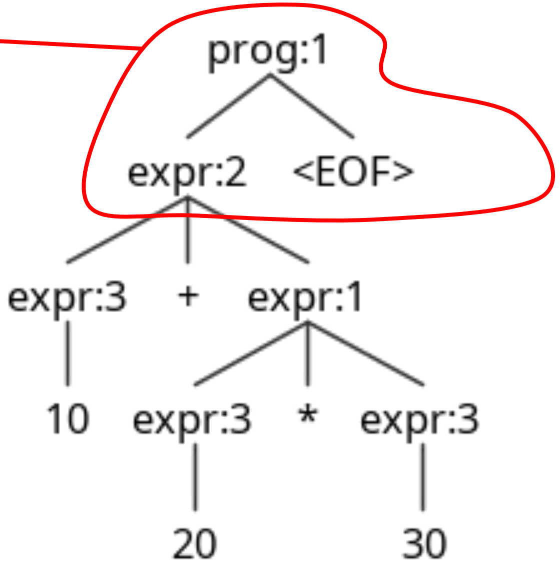
directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Let's see installation and do some parsing



```
grammar Expr;
prog: expr EOF ;
expr:1expr ('*'|'/') expr
    |2expr ('+'|'-') expr
    |3INT
    |4'(' expr ')'
    ;
NEWLINE : [\r\n]+ -> skip;
INT: [0-9]+ ;
```

$10 + 20 * 30$



directive ***->skip***; tells the lexer to completely ignore the matched token—it will not emit it to the parser

Lex Attributes

```
$INT->getText()
$INT->getLine()

INT: [0-9]+ ;
```

Attribute	Type	Description
text	String	The text matched for the token; translates to a call to getText(). Example: $ID.text.
type	int	The token type (nonzero positive integer) of the token such as INT; translates to a call to getType(). Example: $ID.type.
line	int	The line number on which the token occurs, counting from 1; translates to a call to getLine(). Example: $ID.line.

Attribute	Type	Description
pos	int	The character position within the line at which the token's first character occurs counting from zero; translates to a call to getCharPositionInLine(). Example: $ID.pos.
index	int	The overall index of this token in the token stream, counting from zero; translates to a call to getTokenIndex(). Example: $ID.index.
channel	int	The token's channel number. The parser tunes to only one channel, effectively ignoring off-channel tokens. The default channel is 0 (Token.DEFAULT_CHANNEL), and the default hidden channel is Token.HIDDEN_CHANNEL. Translates to a call to getChannel(). Example: $ID.channel.
int	int	The integer value of the text held by this token; it assumes that the text is a valid numeric string. Handy for building calculators and so on. Translates to Integer.valueOf(text-of-token). Example: $INT.int.

Basic Structure of Files

C8086Lexer.g4

lexer grammar C8086Lexer;

@lexer::header {
Commonly used to include necessary imports, std libraries or other dependencies.
Placed at the top of the generated lexer class, so that the necessary headers are included.
}

@lexer::members {
Declare variables, helper functions, utility functions used within the lexer rules.
Becomes a member of the generated lexer class and can be invoked within lexer rules to log messages.
}

// lexer rules here

Basic Structure of Files

C8086Lexer.g4

```
lexer grammar C8086Lexer;

@lexer::header {
Commonly used to include necessary imports, std
libraries or other dependencies.
Placed at the top of the generated lexer class, so that
the necessary headers are included.
}

@lexer::members {
Declare variables, helper functions, utility functions
used within the lexer rules.
Becomes a member of the generated lexer class and can
be invoked within lexer rules to log messages.
}

// lexer rules here
```

C8086Parser.g4

```
parser grammar C8086Parser;
options {
    tokenVocab = C8086Lexer;
}

@parser::header {
Commonly used to include necessary imports, std
libraries or other dependencies.
Placed at the top of the generated parser class, so that
the necessary headers are included.
}

@parser::members {
Declare variables, helper functions, utility functions
used within the grammar rules.
Becomes a member of the generated parser class and
can be invoked within parser rules to log messages.
}
// parser rules here
```

Basic Structure of Files

C8086Lexer.g4

lexer grammar C8086Lexer;

@lexer::header {
Commonly used to include necessary imports, std libraries or other dependencies.
Placed at the top of the generated lexer class, so that the necessary headers are included.
}

@lexer::members {
Declare variables, helper functions, utility functions used within the lexer rules.
Becomes a member of the generated lexer class and can be invoked within lexer rules to log messages.
}

// lexer rules here

C8086Parser.g4

parser grammar C8086Parser;

options {
    tokenVocab = C8086Lexer;
}

Instruct the parser grammar to use tokens defined in a separate lexer grammar.

@parser::header {
Commonly used to include necessary imports, std libraries or other dependencies.
Placed at the top of the generated parser class, so that the necessary headers are included.
}

@parser::members {
Declare variables, helper functions, utility functions used within the grammar rules.
Becomes a member of the generated parser class and can be invoked within parser rules to log messages.
}

// parser rules here

Embedded actions

However, you can give some embedded actions in grammar rules.

```
A : B C {printf(“hello i am embedded”)} D
        ;
```

Let's see some real stuffs