Compilers
Lab II

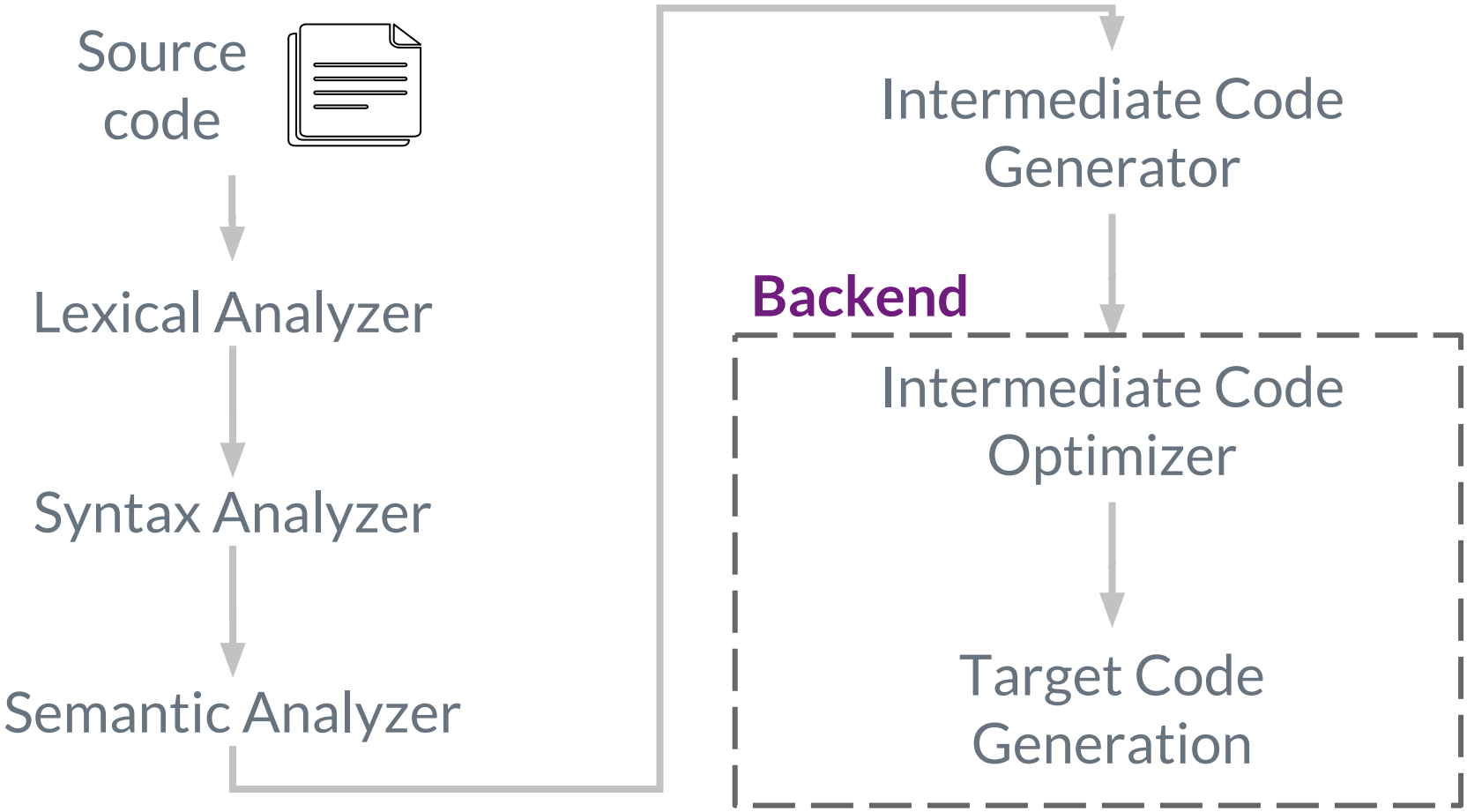
Mo.Agamia

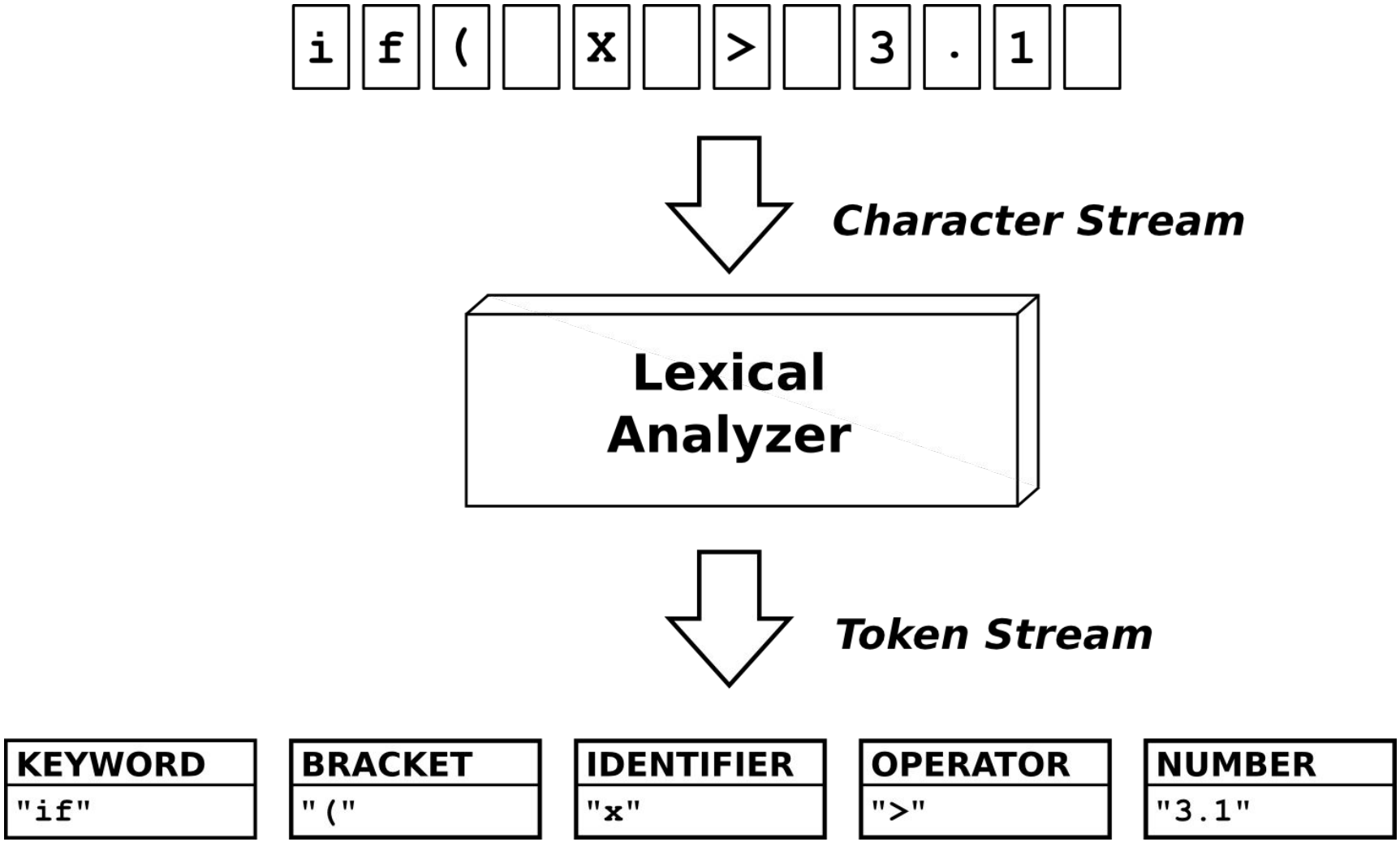
Plan

- Overview
- ε-NFA to DFA
- ANTLR

1.
Overview

Compiler phases



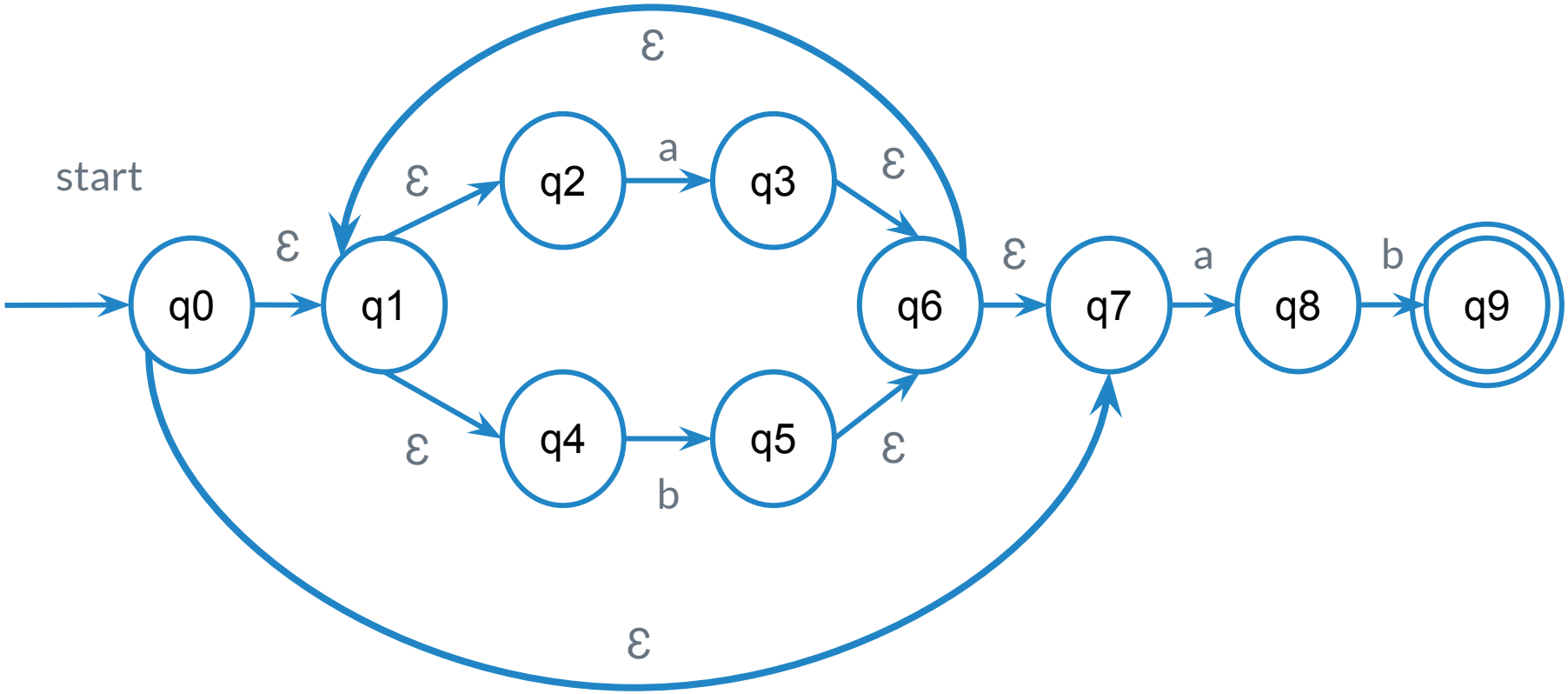

i f (x > 3 . 1
Character Stream
Lexical Analyzer
Token Stream
KEYWORD
"if"
BRACKET
"("
IDENTIFIER
"x"
OPERATOR
">"
NUMBER
"3.1"

1. *Write regular definition*
2. *Compile corresponding regular expression*
3. *Convert expression to NFA*
4. *Convert NFA to DFA*

2.
ε-NFA to DFA

NFA Examples

(a|b)*ab



NFA Examples

(a|b)*ab

DFA state	NFA state	a	b
A	{q0, q1, q2, q4, q7}	B	C
B	{q1, q2 ,q3 ,q4 ,q6 ,q7 ,q8}	B	D
C	{q1, q2, q4, q5, q6, q7}	B	C
D	{q1, q2, q4, q5, q6, q7, q9}	B	C

3.
ANTLR

ANother Tool for Language Recognition, is a powerful parser generator for reading, processing, executing, or translating structured text or binary files.

ANTLR

Download v4.7.2 | 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.

Check out Terence's latest adventure explained.ai

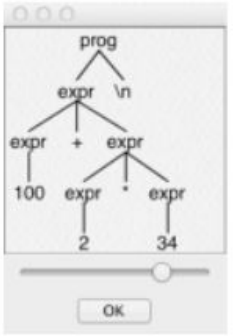
Terence Parr is the maniac behind ANTLR and has been working on language tools since 1989. He is a professor of computer science at the University of San Francisco.

Quick Start | Samples

Samples

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

```
$ antlr4 Expr.g4
$ javac Expr*.java
$ grun Expr prog -gui
100+2*34
^D
```



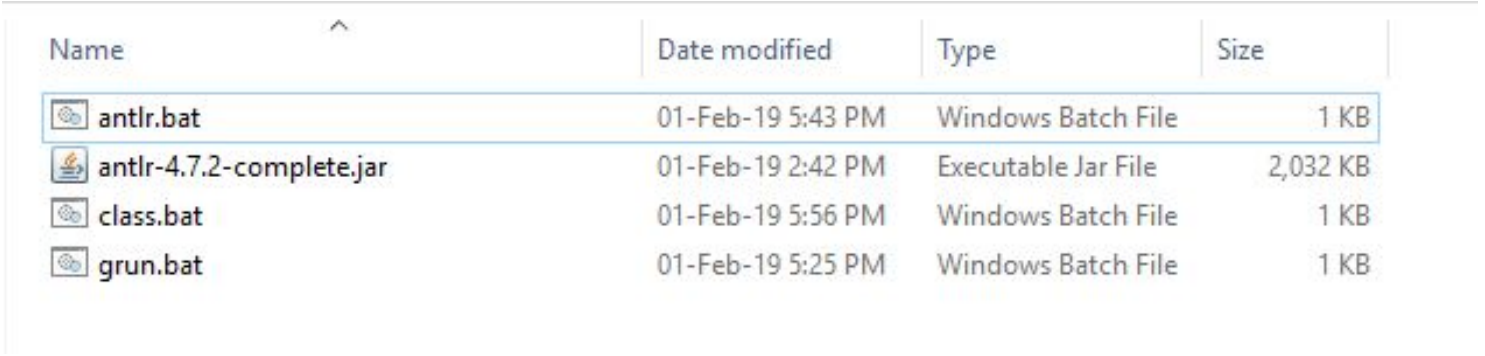
ANTLR

ANTLR takes as input a grammar that specifies a language and generates as output source code for a recognizer of that language.

ANTLR can generate lexers, parsers, tree parsers, and combined lexer-parsers.

- Linear approximate lookahead
- Semantic and Syntactic predicates
- ANTLRWorks
- Tree parsing
- LL(*)
- Adaptive LL(*) in ANTLR v4

ANTLR Setup



Name	Date modified	Type	Size
antlr.bat	01-Feb-19 5:43 PM	Windows Batch File	1 KB
antlr-4.7.2-complete.jar	01-Feb-19 2:42 PM	Executable Jar File	2,032 KB
class.bat	01-Feb-19 5:56 PM	Windows Batch File	1 KB
grun.bat	01-Feb-19 5:25 PM	Windows Batch File	1 KB

antlr.bat

```
1  java org.antlr.v4.Tool %*
```

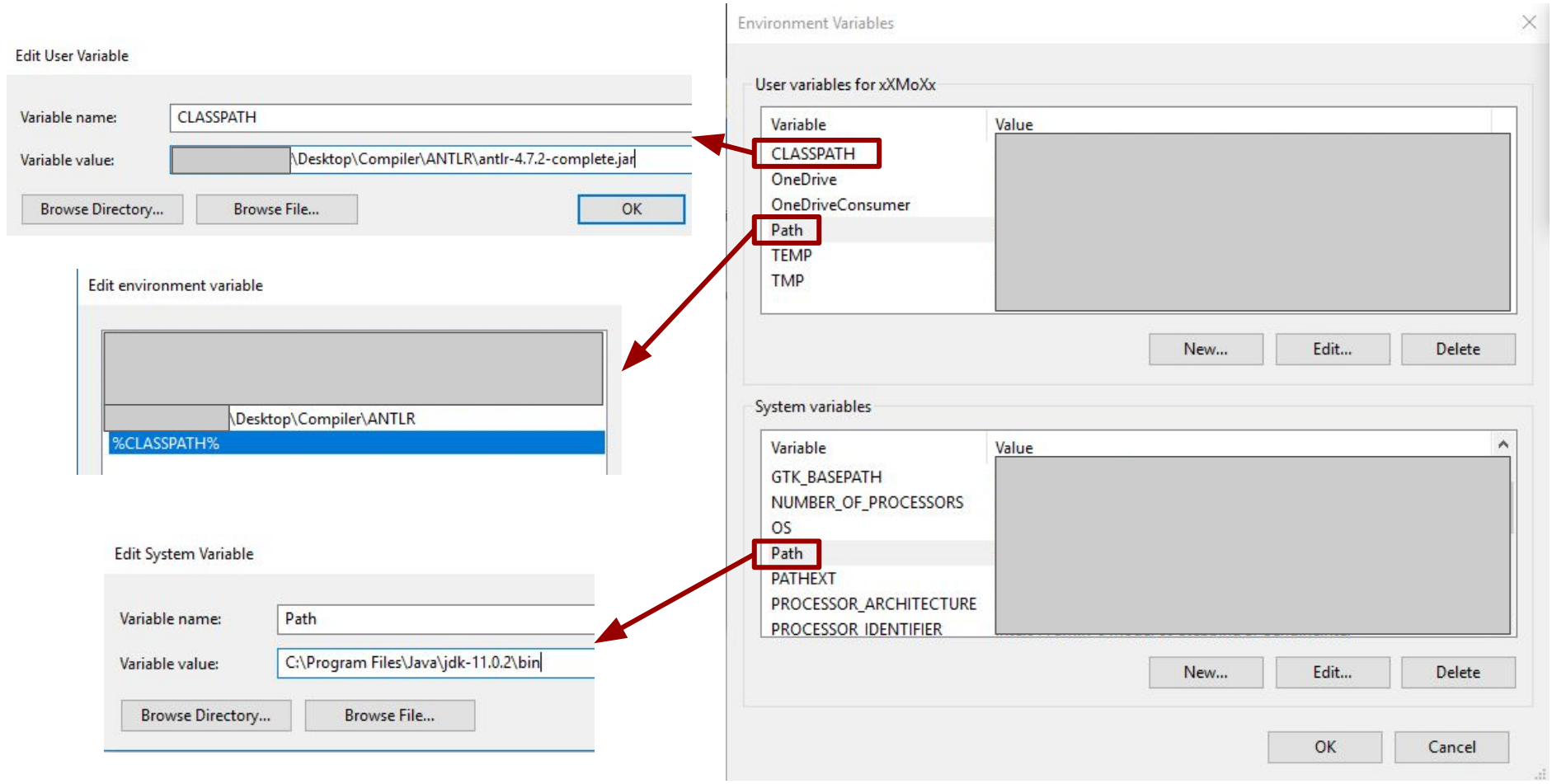
grun.bat

```
1  java org.antlr.v4.gui.TestRig %*
```

class.bat

```
1  SET CLASSPATH=.;%CLASSPATH%
```

ANTLR Setup


Edit User Variable
Variable name:
CLASSPATH
Variable value:
\Desktop\Compiler\ANTLR\antlr-4.7.2-complete.jar
Browse Directory...
Browse File...
OK
Edit environment variable
\Desktop\Compiler\ANTLR
%CLASSPATH%
Edit System Variable
Variable name:
Path
Variable value:
C:\Program Files\Java\jdk-11.0.2\bin
Browse Directory...
Browse File...
Environment Variables
User variables for xXMoXx
Variable
Value
CLASSPATH
OneDrive
OneDriveConsumer
Path
TEMP
TMP
New...
Edit...
Delete
System variables
Variable
Value
GTK_BASEPATH
NUMBER_OF_PROCESSORS
OS
Path
PATHEXT
PROCESSOR_ARCHITECTURE
PROCESSOR_IDENTIFIER
New...
Edit...
Delete
OK
Cancel

ANTLR Setup

Unix:

1. sudo apt update
2. sudo apt install antlr4
3. sudo apt install default-jdk
4. sudo apt install python3-pip
5. pip3 install --user antr4-python3-runtime

```
grammar test;

start:   (expr NEWLINE)* ;

expr:    expr OPERATOR expr
     |   INT
     |   '(' expr ')'
     ;

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

```
100+2*24
```

```
import argparse
from antlr4 import *
from testLexer import testLexer
from testListener import testListener
from testParser import testParser
from antlr4.tree.Trees import Trees

def get_token_type(token):
    if token.type == testLexer.INT:
        return "INT"
    elif token.type == testLexer.NEWLINE:
        return "NEWLINE"
    elif token.type == testLexer.OPERATOR:
        return "OPERATOR"
    else:
        return "ERROR UNKNOWN TOKEN"

def main():

    with open(args.file, "r") as file:
        lines = file.read()
    input_stream = InputStream(lines)
    lexer = testLexer(input_stream)
    token_stream = CommonTokenStream(lexer)
    parser = testParser(token_stream)

#   tree = parser.start()
#   print(Trees.toStringTree(tree,None, parser))

    token = lexer.nextToken()

    while not token.type == Token.EOF:
        print(get_token_type(token), token.text)
        token = lexer.nextToken()
```

```
sample.txt
ANTLR runtime and generated code versions disagree: 4.7.2!=4.5.1
ANTLR runtime and generated code versions disagree: 4.7.2!=4.5.1
INT 100
OPERATOR +
INT 2
OPERATOR *
INT 24
NEWLINE
```

ANTLR Setup

To Run:

- antlr test.g4
- antlr test.g4 -Dlanguage=Python3
- javac test*.java
- class
- grun test start sample.txt -gui

Thanks!

Any questions?

You can find me at:

@piazza

mohammed.agamia@guc.edu.eg