Laboratorio 1

Construcción de compiladores

Mauricio Julio Rodrigo Lemus Guzmán – 22461

Análisis

La gramática en ANTLR se define en archivos .g4 que separan las reglas léxicas (mayúsculas) de las reglas sintácticas (minúsculas). Estas gramáticas permiten estructurar el lenguaje que queremos reconocer. Las reglas léxicas identifican los componentes básicos como palabras clave, identificadores, operadores o números, mientras que las reglas sintácticas definen cómo se combinan esos componentes para formar estructuras válidas. Elementos como los literales entre comillas, los rangos con corchetes y el uso del símbolo | para alternancia ayudan a construir estas reglas. Además, se pueden usar directivas como -> skip para ignorar ciertos caracteres como espacios, y etiquetas con # que facilitan el uso de visitantes o listeners en fases más avanzadas del análisis.

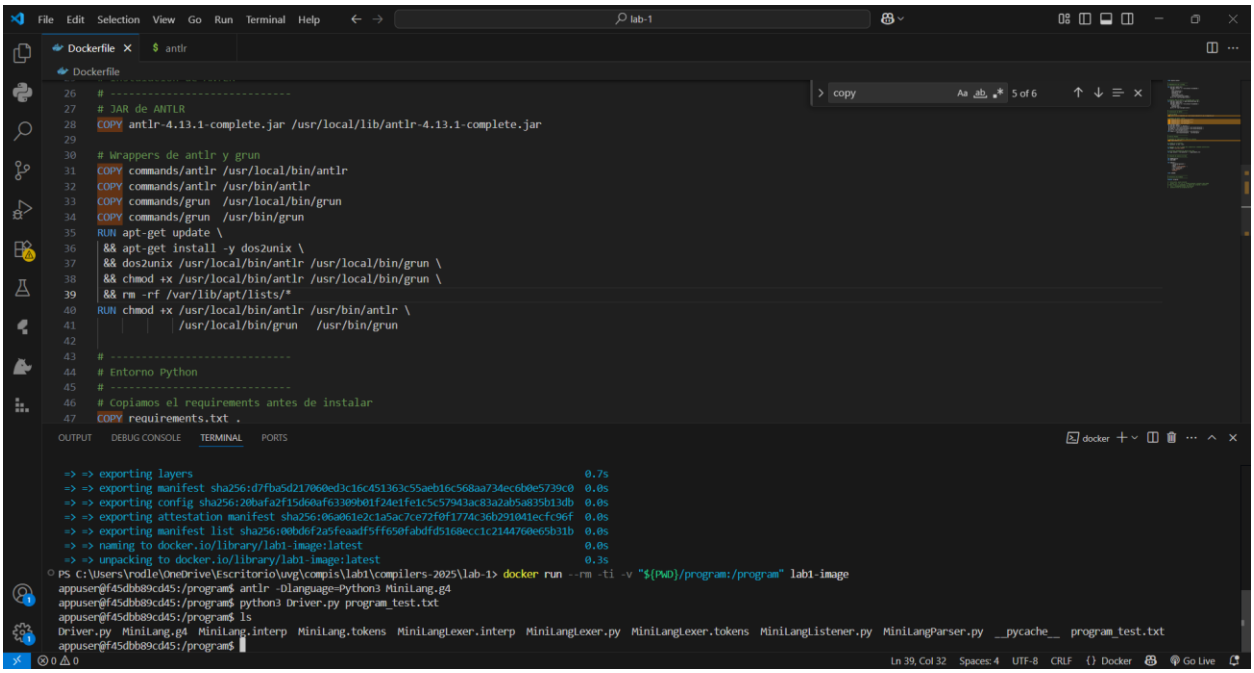
El archivo Driver.py actúa como el punto de entrada para probar la gramática. Su función principal es leer un archivo de entrada, pasar ese contenido al lexer y parser generados por ANTLR, y luego ejecutar la regla inicial definida en la gramática. Si el contenido cumple con la estructura esperada, no se muestra ningún mensaje, pero si hay errores sintácticos, ANTLR los reporta directamente en consola. Esto permite comprobar fácilmente si una entrada es válida según la gramática definida. En conjunto, la gramática y el driver permiten crear y probar lenguajes propios de manera sencilla, detectando errores sin necesidad de implementar un analizador manual desde cero.

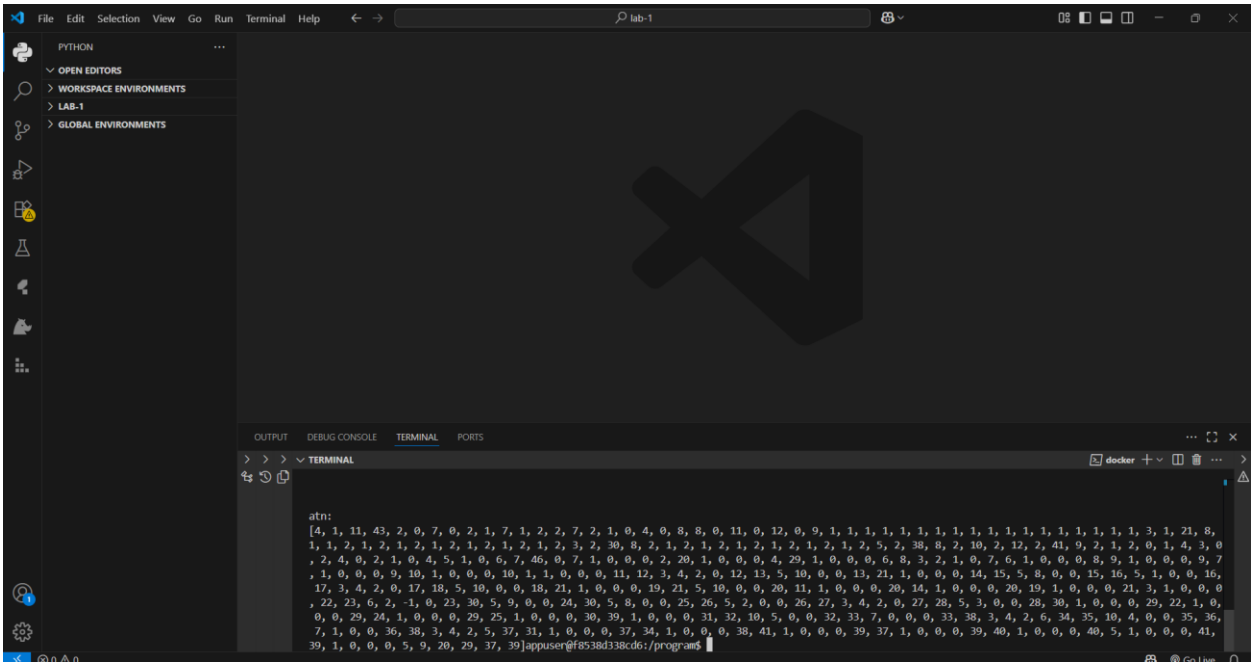
Video

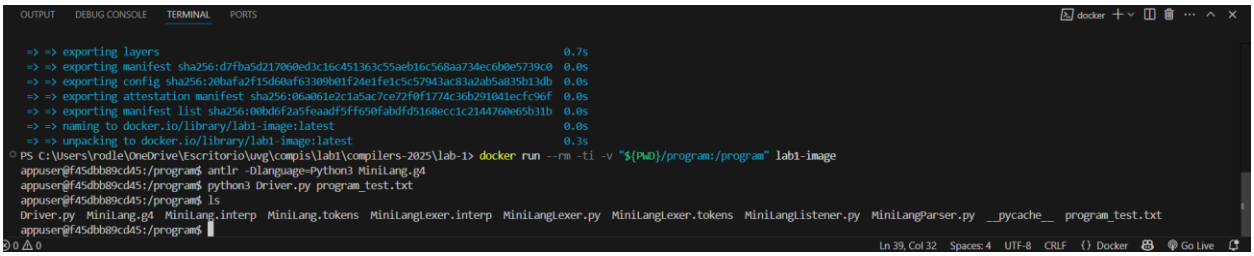
Enlace al video

Repositorio

Enlace al repositorio


File Edit Selection View Go Run Terminal Help
lab-1
Dockerfile
antlr
copy
5 of 6
JAR de ANTLR
COPY antlr-4.13.1-complete.jar /usr/local/lib/antlr-4.13.1-complete.jar
Wrappers de antlr y grun
COPY commands/antlr /usr/local/bin/antlr
COPY commands/antlr /usr/bin/antlr
COPY commands/grun /usr/local/bin/grun
COPY commands/grun /usr/bin/grun
RUN apt-get update \
&& apt-get install -y dos2unix \
&& dos2unix /usr/local/bin/antlr /usr/local/bin/grun \
&& chmod +x /usr/local/bin/antlr /usr/local/bin/grun \
&& rm -rf /var/lib/apt/lists/*
RUN chmod +x /usr/local/bin/antlr /usr/bin/antlr \
/usr/local/bin/grun /usr/bin/grun
Entorno Python
Copiamos el requirements antes de instalar
COPY requirements.txt .
OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker
=> => exporting layers 0.7s
=> => exporting manifest sha256:d7fba5d217060ed3c16c451363c55aeb16c568aa734ec6b0e5739c0 0.0s
=> => exporting config sha256:20bafa2f15d60af63309b01f24e1fe1c5c57943ac83a2ab5a835b13db 0.0s
=> => exporting attestation manifest sha256:06a061e2c1a5ac7ce72f0f1774c36b291041ecfc96f 0.0s
=> => exporting manifest list sha256:00bd6f2a5feaadf5ff650fabdfd5168ecc1c2144760e65b31b 0.0s
=> => naming to docker.io/library/lab1-image:latest 0.0s
=> => unpacking to docker.io/library/lab1-image:latest 0.3s
PS C:\Users\rodle\OneDrive\Escritorio\uvg\compis\lab1\compilers-2025\lab-1> docker run --rm -ti -v "${PWD}/program:/program" lab1-image
appuser@f45dbb89cd45:/program$ antlr -Dlanguage=Python3 MiniLang.g4
appuser@f45dbb89cd45:/program$ python3 Driver.py program_test.txt
appuser@f45dbb89cd45:/program$ ls
Driver.py MiniLang.g4 MiniLang.interp MiniLang.tokens MiniLangLexer.interp MiniLangLexer.py MiniLangLexer.tokens MiniLangListener.py MiniLangParser.py __pycache__ program_test.txt
appuser@f45dbb89cd45:/program$
Ln 39, Col 32 Spaces: 4 UTF-8 CRLF {} Docker Go Live


File Edit Selection View Go Run Terminal Help
lab-1
PYTHON
OPEN EDITORS
WORKSPACE ENVIRONMENTS
LAB-1
GLOBAL ENVIRONMENTS
OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
docker
atn:
[4, 1, 11, 43, 2, 0, 7, 0, 2, 1, 7, 1, 2, 2, 7, 2, 1, 0, 4, 0, 8, 8, 0, 11, 0, 12, 0, 9, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 21, 8,
1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 3, 2, 30, 8, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 5, 2, 38, 8, 2, 10, 2, 12, 2, 41, 9, 2, 1, 2, 0, 1, 4, 3, 0
, 2, 4, 0, 2, 1, 0, 4, 5, 1, 0, 6, 7, 46, 0, 7, 1, 0, 0, 0, 2, 20, 1, 0, 0, 0, 4, 29, 1, 0, 0, 0, 6, 8, 3, 2, 1, 0, 7, 6, 1, 0, 0, 0, 8, 9, 1, 0, 0, 0, 9, 7
, 1, 0, 0, 0, 9, 10, 1, 0, 0, 0, 10, 1, 1, 0, 0, 0, 11, 12, 3, 4, 2, 0, 12, 13, 5, 10, 0, 0, 13, 21, 1, 0, 0, 0, 14, 15, 5, 8, 0, 0, 15, 16, 5, 1, 0, 0, 16,
17, 3, 4, 2, 0, 17, 18, 5, 10, 0, 0, 18, 21, 1, 0, 0, 0, 19, 21, 5, 10, 0, 0, 20, 11, 1, 0, 0, 0, 20, 14, 1, 0, 0, 0, 20, 19, 1, 0, 0, 0, 21, 3, 1, 0, 0, 0
, 22, 23, 6, 2, -1, 0, 23, 30, 5, 9, 0, 0, 24, 30, 5, 8, 0, 0, 25, 26, 5, 2, 0, 0, 26, 27, 3, 4, 2, 0, 27, 28, 5, 3, 0, 0, 28, 30, 1, 0, 0, 0, 29, 22, 1, 0,
0, 0, 29, 24, 1, 0, 0, 0, 29, 25, 1, 0, 0, 0, 30, 39, 1, 0, 0, 0, 31, 32, 10, 5, 0, 0, 32, 33, 7, 0, 0, 0, 33, 38, 3, 4, 2, 6, 34, 35, 10, 4, 0, 0, 35, 36,
7, 1, 0, 0, 36, 38, 3, 4, 2, 5, 37, 31, 1, 0, 0, 0, 37, 34, 1, 0, 0, 0, 38, 41, 1, 0, 0, 0, 39, 37, 1, 0, 0, 0, 39, 40, 1, 0, 0, 0, 40, 5, 1, 0, 0, 0, 41,
39, 1, 0, 0, 0, 5, 9, 20, 29, 37, 39]appuser@f8538d338cd6:/program$


OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker
=> => exporting layers 0.7s
=> => exporting manifest sha256:d7fba5d217060ed3c16c451363c55aeb16c568aa734ec6b0e5739c0 0.0s
=> => exporting config sha256:20bafa2f15d60af63309b01f24e1fe1c5c57943ac83a2ab5a835b13db 0.0s
=> => exporting attestation manifest sha256:06a061e2c1a5ac7ce72f0f1774c36b291041ecfc96f 0.0s
=> => exporting manifest list sha256:00bd6f2a5feaadf5ff650fabdfd5168ecc1c2144760e65b31b 0.0s
=> => naming to docker.io/library/lab1-image:latest 0.0s
=> => unpacking to docker.io/library/lab1-image:latest 0.3s
PS C:\Users\rodle\OneDrive\Escritorio\uvg\compis\lab1\compilers-2025\lab-1> docker run --rm -ti -v "${PWD}/program:/program" lab1-image
appuser@f45dbb89cd45:/program$ antlr -Dlanguage=Python3 MiniLang.g4
appuser@f45dbb89cd45:/program$ python3 Driver.py program_test.txt
appuser@f45dbb89cd45:/program$ ls
Driver.py MiniLang.g4 MiniLang.interp MiniLang.tokens MiniLangLexer.interp MiniLangLexer.py MiniLangLexer.tokens MiniLangListener.py MiniLangParser.py __pycache__ program_test.txt
appuser@f45dbb89cd45:/program$
Ln 39, Col 32 Spaces: 4 UTF-8 CRLF {} Docker Go Live