Análisis Sintáctico del Compilador Mini-Python

Fabián Fernández Kevin Jiménez Justin Martínez

Compiladores e Intérpretes
Tecnológico de Costa Rica
Profesor: Oscar Mario Víquez Acuña
II Semestre 2024

Contents

1 Introducción

El proyecto se centra en la creación de un Analizador Sintáctico para un subconjunto de Python denominado Mini-Python, utilizando la herramienta ANTLR4 y el lenguaje de programación C#. El propósito es abordar las fases de análisis léxico (scanner) y análisis sintáctico (parser) del compilador, asegurando que el código fuente escrito en Mini-Python sea correctamente interpretado y procesado. Además, se implementa una interfaz gráfica en WPF que permite la ejecución y visualización de los errores, con diferentes funcionalidades, facilitando así la depuración y la prueba del código.

2 Análisis del Lenguaje

2.1 Gramática

El lenguaje Mini-Python sigue una gramática simplificada, presentada en formato EBNF para su implementación en ANTLR4. La gramática cubre las declaraciones y expresiones fundamentales del lenguaje, como funciones, ciclos y condiciones. Algunos aspectos clave de la gramática incluyen:

```
program: mainStatement* EOF;

mainStatement: defStatement | assignStatement;

defStatement: DEF ID '(' argList ')' ':' sequence;
assignStatement: ID ASSIGN expression NEWLINE;

ifStatement: IF expression ':' sequence ELSE ':' sequence;
whileStatement: WHILE expression ':' sequence;

sequence: INDENT statement+ DEDENT;

expression: additionExpression comparison?;
additionExpression: multiplicationExpression ((PLUS | MINUS) multiplicationExpression)*;
multiplicationExpression: elementExpression ((MULT | DIV) elementExpression)*;
```

2.2 Tokens

Los tokens fundamentales en Mini-Python, incluyendo palabras clave como `def`, `if`, operadores aritméticos, y delimitadores, son los siguientes:

Token	Descripción
IF	Palabra reservada ‘if”
DEF	Palabra reservada ‘def”
ASSIGN	Operador de asignación ‘=”
PLUS	Operador suma ‘+”
INT	Número entero
STRING	Cadena de texto

3 Soluciones e Implementación

3.1 Scanner

El *scanner* está implementado utilizando ANTLR4 para identificar los tokens. Una parte importante es el manejo de la indentación y dedentación, crucial en Mini-Python:

```
lexer grammar MiniPythonLexer;

tokens { INDENT, DEDENT }

@lexer::members {
    private DenterHelper denter;

    public override IToken NextToken() {
        if (denter == null) {
            denter = DenterHelper.Builder()
                .Nl(NEWLINE)
                .Indent(MiniPythonParser.INDENT)
                .Dedent(MiniPythonParser.DEDENT)
                .PullToken(base.NextToken);
        }
        return denter.NextToken();
    }
}

NEWLINE: ('\r'? '\n' ' '*);
COMMENT: '#' ~[\r\n]* -> channel(HIDDEN);
MULTILINE_COMMENT: '"""' .*? '"""' -> channel(HIDDEN);
WS: [ \t]+ -> skip;

DEF: 'def';
IF: 'if';
PRINT: 'print';
ASSIGN: '=';
PLUS: '+';
INT: [0-9]+;
STRING: '"' .*? '"';
```

3.2 Parser

El *parser* utiliza el enfoque de descenso recursivo proporcionado por ANTLR4, con un manejo especial de errores. A continuación, un fragmento del archivo de parser:

```
parser grammar MiniPythonParser;

options { tokenVocab = MiniPythonLexer; }

program: mainStatement* EOF;

mainStatement: defStatement | assignStatement;

defStatement: DEF ID '(' argList ')' ':' sequence;
assignStatement: ID ASSIGN expression NEWLINE;

ifStatement: IF expression ':' sequence ELSE ':' sequence;
sequence: INDENT statement+ DEDENT;
```

3.3 Interfaz Gráfica

La interfaz gráfica está implementada utilizando WPF. Incluye un editor de texto que permite visualizar el código, el numero de lineas existenten y un boton para ejecutar compilaciones, mostrando los errores en un panel separado. Además, los errores detectados en la compilación se muestran y proporcionan la linea del error, además permite abrir archivos locales, archivos desde web (en este caso se implementó la prueba con el uso de archivos desde github) ademas de la opción de guardar el archivo o simplemente desplegar un archivo nuevo sin titulo desde la misma interfaz.

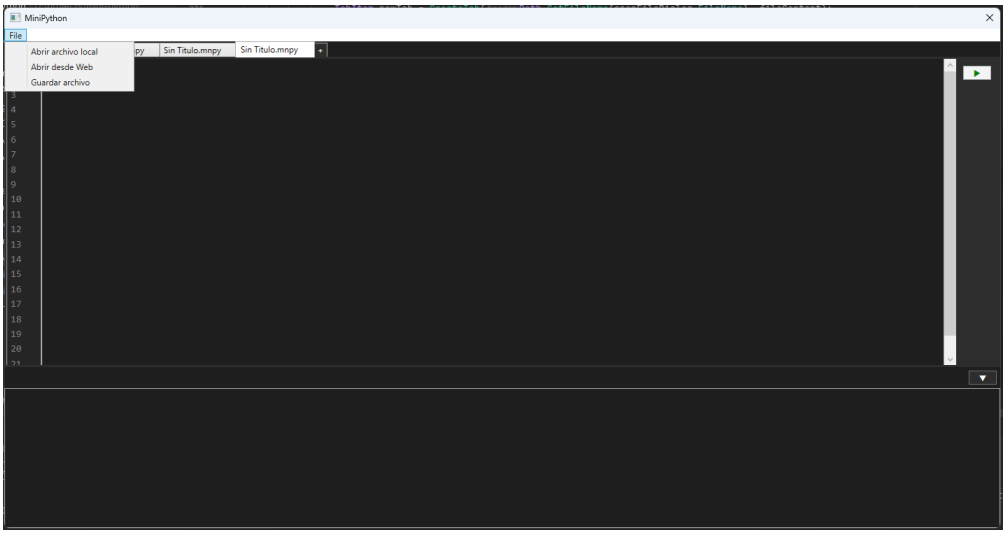

Figure 1: Interfaz gráfica del compilador Mini-Python

3.4 Manejo de Errores

El manejo de errores es crítico tanto en el scanner como en el parser. Se implementó una clase `CustomErrorListener` que intercepta los errores léxicos y sintácticos, mostrando mensajes con línea y columna del error:

```
public void SyntaxError(TextWriter output, IRecognizer recognizer, IToken offendingSymbo
    int line, int charPositionInLine, string msg, RecognitionException e) {
    ErrorMsgs.Add($"PARSER ERROR - line {line}:{charPositionInLine + 1} {msg}");
}
```

4 Resultados Obtenidos

Para esta primera entrega, se han logrado los siguientes hitos:

Funcionalidad	Estado
Análisis léxico	Completado
Análisis sintáctico	Completado
Interfaz gráfica	Completada
Manejo de errores	Implementado pero no completo

5 Problemas Identificados

5.1 Interfaz de Depuración

No se implementó una interfaz de depuración que permita ejecutar el código línea por línea. Esta es una característica pendiente.

5.2 Hipervínculo en la Consola de Errores

Aunque salen los mensajes de error en la interfaz y dicen la linea, no permiten navegar a la línea correspondiente, el sistema de hipervínculos aún no está completamente optimizado para todas las situaciones.

5.3 Error en el Parser

Se detectó un error en el parser relacionado con la detección incorrecta de una línea adicional al final del archivo (EOF):

```
PARSER ERROR - line 2:1 missing NEWLINE at '<EOF>'
```

Este problema surge por el manejo inadecuado del token de EOF, lo que lee y genera una línea adicional al final de cualquier archivo .mnpy

6 Conclusiones

Este proyecto ha sido útil para entender las complejidades de construir un analizador léxico y sintáctico utilizando ANTLR4 y C#. A pesar de los desafíos, como el manejo de indentación y errores específicos de la sintaxis de Python, se ha avanzado significativamente en la implementación de un compilador funcional con una interfaz gráfica completa. Se implementará lo que sea necesario para iniciar con la segunda etapa del proyecto.

7 Bibliografía

References

[1] Terrence Parr. *The Definitive ANTLR 4 Reference*. 2013.

[2] Antlr-Denter, `https://github.com/yshavit/antlr-denter`.