JÓVENES EN LA CIENCIA
REVISTA DE DIVULGACIÓN CIENTÍFICA

METACOMPILADOR DIDÁCTICO GENERADOR DE CÓDIGO JAVA

Erick Leonel Rico Preciado[1], Ana Cristina Bueno Campos[1], José Gerardo Carpio Flores[2], Ruth Sáez de Nanclares Rodríguez[2], Martha Alicia Rocha Sánchez[2]

Resumen

La investigación se enfoca en la construcción de un metacompilador, el cual es una herramienta especializada en el área de compiladores, en la categoría de gramáticas LL(1), siendo una investigación de desarrollo tecnológico, así como en el diseño de un lenguaje de programación. El metacompilador será utilizado de manera práctica y como caso de estudio en la impartición de la asignatura de Lenguajes y Autómatas II, perteneciente al plan de estudios de la carrera de Ingeniería en Sistemas Computacionales. El metacompilador es un software generador de analizadores sintácticos escritos en lenguaje java, que incluye un ambiente de desarrollo integrado, así como un módulo de detección de errores, y la generación de código java.

Palabras Clave Compiladores, Teoría de la Computación, IDE, Ingeniaría de Software, Lenguaje.

[1] Ingeniería en Sistemas Computacionales, Instituto Tecnológico de León, Av. Tecnológico s/n, Fracc. Industrial Julián de Obregón, C.P. 37280, Guanajuato, León, Teléfono (477) 710 5200.

[2] Instituto Tecnológico de León, Departamento de Sistemas y Computación, Av. Tecnológico s/n, Fracc. Industrial Julián de Obregón, C.P. 37280, Guanajuato, León, Teléfono (477) 710 5200; Fax: 711-2072; *gcarpio@ymail.com , r.saezdenanclares@yahoo.com.mx , martaalicia.rocha@itl.edu.mx*

INTRODUCCIÓN

Los compiladores son fundamentales en la computación, actúan como traductores, transforman los lenguajes de programación orientados a los humanos en lenguajes orientados al computador (Lemone, 1999). Para la mayoría de los usuarios, un compilador es visto como una caja negra que realiza la siguiente transformación (ver Figura 1):

Figura 1. Proceso de Compilación

Un compilador permite que la mayoría de los usuarios de computadoras ignoren los detalles del lenguaje máquina; los compiladores permiten que los programas y los programadores sean independientes de la plataforma computacional que ejecutará el código. El término compilador fue inventado a inicios de la década de los 50's por Grace Murray Hopper. La traducción fue vista como la compilación de una secuencia de subprogramas seleccionados de una biblioteca. La compilación fue llamada programación automática por la respuesta a la ejecución de código. Entre los primeros compiladores reales, están los compiladores de FORTRAN de finales de los 50's. Uno de los primeros compiladores de FORTRAN tomo 18 años hombre en su construcción (Lemone, 1999). El proceso de traducción puede describirse como una secuencia de fases seriadas que se describen a continuación (Aho, 2008):

- Análisis Léxico: Reconoce componentes léxicos (tokens).
- Análisis Sintáctico: Verifica mediante estructuras de datos que la secuencia de tokens se apegue a la definición del lenguaje.
- Análisis Semántico: Mediante un árbol de análisis gramatical, se verifica que la entrada sea lógicamente correcta.
- Optimización: Procura que la generación de código se ejecute más rápido y ocupe menos espacio.
- Generación de Código: selección de secuencias de instrucciones adecuadas para obtener código legible.

El término metacompilador se refiere a un programa de computadora que recibe como entrada las especificaciones del lenguaje para el que se desea obtener un compilador, y genera como salida el compilador para ese lenguaje. El desarrollo de los metacompiladores se encuentra con la dificultad de unir las dos fases de atención en la compilación de programas: la fase de análisis y la fase de síntesis, que es la generación de código (Tremblay, 1985).

Las gramáticas de contexto libre son útiles como mecanismos definicionales. Son usadas con frecuencia para usar parsers. (Aho, 2008)La idea es construir un generador de parsers que tome como entrada una clase de gramática y produzca como salida un parser que analice correctamente el lenguaje definido por la gramática. Este concepto es el de un compilador con definiciones de alto nivel que se traduce a formas ejecutables también de alto nivel. La facilidad que una herramienta de estas características aporta para modificar gramáticas y tener un analizador en forma automática es formidable desde el punto de vista didáctico.

Los modelos educativos actuales requieren del aprendizaje centrado en el alumno, por lo cual el desarrollo del metacompilador auxiliará al alumno en la comprensión de la teoría de autómatas y lenguajes formales. La

herramienta se enfoca en la enseñanza a través de la implementación computacional de los conceptos, y así el alumno no deje de lado la programación de software de base.

MATERIALES Y MÉTODOS

El metacompilador realiza una traducción dirigida por sintaxis (parsing), así mismo se ha programado bajo el paradigma orientado a objetos del lenguaje Java, utilizando como herramienta de software el entorno integrado de desarrollo NetBeans. El desarrollo se realizó bajo la siguiente metodología:

Actualización Teórico-Práctica

Es necesario repasar algunos conceptos previos como:

- Actualización de los conocimientos teóricos necesarios para desarrollar el meta compilador.
- Desarrollo de la revisión técnica que permita conocer en dónde se encuentra en este momento la frontera del conocimiento en desarrollo y la aplicación de meta compiladores en el estudio y desarrollo de software de base.
- Comprender el uso y construcción de los meta compiladores más utilizados actualmente.
- Analizar los Ambientes de Desarrollo Integrados más populares en este momento.

Implementación Computacional

Para la parte de implementación, se desarrolló las siguientes partes que componen al metacompilador:

- Desarrollo de las especificaciones léxicas y sintácticas que deberá observar el usuario final del metacompilador.
- Elección de los algoritmos apropiados para el desarrollo de un traductor que utilice gramáticas Libres de contexto y Drivers LL(1).
- Desarrollo e implementación de los algoritmos elegidos.
- Elección de las estrategias de generación de código más adecuadas y accesibles para el tipo de traductor que nos proponemos desarrollar.
- Codificación de los algoritmos requeridos para la generación de código objeto.
- Elección de las funcionalidades que el IDE debe ofrecer a los usuarios finales.
- Desarrollo de los algoritmos necesarios para implantar las funcionales del IDE.
- Codificación de algoritmos para la el IDE.

Corrección y Pruebas

Testeo de los algoritmos desarrollados, análisis y corrección de errores, tales como:

- Verificación del funcionamiento del analizador léxico-sintáctico.
- Pruebas con diversas gramáticas de contexto libre elegidas.
- Pruebas de rendimiento y detección de errores para el IDE.
- Pruebas de integración entre el metacompilador y el IDE.

RESULTADOS Y DISCUSIÓN

Se ha creado un lenguaje de programación nombrado "*MCD"*, que define una estructura genérica que le indica al usuario como es que debe escribir sus gramáticas, en la figura 2 se muestra un Subset de Pascal de la manera en la que se desea sea introducida por el usuario (ver Figura 2).

```
Language SubsetPascal {

        Special_Symbols_Punctuation {
                Equ = '=' ;
                DosP =':' ;
                Coma =',' ;
                pAbre ='(' ;
                pCierra =')' ;
                Ptocoma =';' .
        }
        Grammar {
                system_goal> program;
                program > begin statement_list end;
                statement_list > statement statement_tail;
                statement_tail > statement statement_tail ;
                statement_tail > ;
                statement > id ':=' expression ';' ;
                statement > read '('id_list')'';' ;
                statement > write '('expr_list')'';' ;
                id_list > id id_tail ;
                id_tail > ','id id_tail ;
                id_tail > ;
                expr_list > expression expr_tail ;
                expr_tail > ','expression expr_tail ;
                expr_tail > ;
                expression > primary primary_tail ;
                primary_tail > addop primary primary_tail ;
                primary_tail > ;
                primary > '('expression')' ;
                primary > id ;
                primary > intLiteral ;
                primary > RealNum ;
                addop > '+' ;
                addop > '-' .
        }
}
```

Figura 2.Gramatica Subset de Pascal

Implementación de un autómata genérico capaz de tokenizar una entrada cualquiera, siendo este la parte primordial del analizador léxico, así como la implantación computacional de los algoritmos first, follow para obtener finalmente la matriz predictiva automáticamente, siendo parte esencial para el correcto funcionamiento del LLDriver, en la figura 3 se puede observar una corrida con la gramática subset de Pascal, mostrando la generación automática de los algoritmos antes mencionados (ver Figura 3).

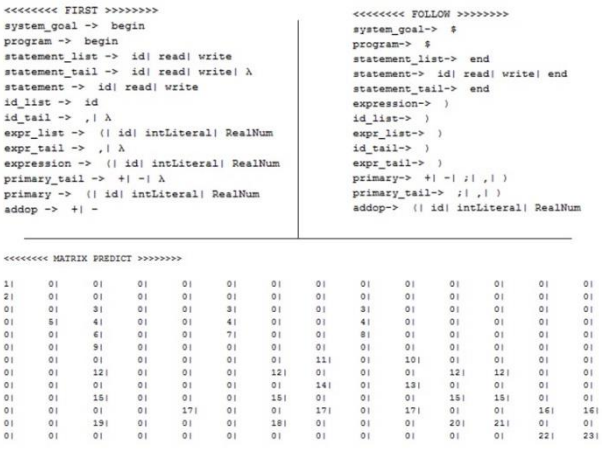

Figura 3. Ejecución de cálculo first, follow y matriz predictiva del Subset de Pascal.

La implementación del algoritmo del LLDriver, es la parte fundamental del metacompilador, ya que mediante él se verifica que la entrada proporcionada por el usuario, contenga la estructura definida por el lenguaje, se ha agregado el módulo de detección de errores para auxiliar al LLDriver en su función, además de reportar al

usuario los tokens en los que se encuentran dichos errores y la capacidad de continuar con la ejecución y encontrar otros posibles errores (ver Figura 4).

```
..::: Errors :::...

-31, tId, tLanguage, Discarded, Token not found!,
-17, tId, Unknown, Discarded, Lexical Error!,
-31, MoreThan, tId, Discarded, Token not found!,
```

Figura 4 Reporte de errores

En la segunda fase de la investigación se diseñó de la interfaz del entorno de desarrollo, así mismo se establecieron las funcionalidades básicas para una primera integración con el metacompilador. Obteniendo como resultado una herramienta en versión beta, estable, de funcionalidad básica (ver Figura 5).

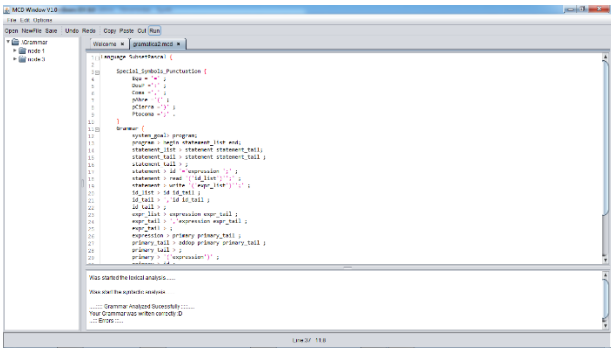

Figura 5. Entorno de Desarrollo Integrado

CONCLUSIONES

Durante el proceso de desarrollo del metacompilador, se considera que existen dos aportaciones sobresalientes. La primera de ellas es la creación de un lenguaje de programación que se acoplara a las necesidades de la herramienta. La segunda, es el diseño y creación de un entorno de desarrollo integrado, ya que la mayoría de herramientas similares, no cuentan con un IDE, por lo que el contar con un IDE, hará el proceso educativo más sencillo y práctico.

REFERENCIAS

Libro:

Aho Alfred V., Sethi Ravi, Lam Monica S., Ullman Jeffrey D., Compiladores principios, técnicas y herramientas, Edit. Pearson Addison Wesley, Segunda Edición 2008.

Lemone Karen A., Fundamentos de Compiladores, como traducir al lenguaje de computadora, Editorial CECSA, 1999.

Tremblay Jean-Paul, Sorenson Paul G., The Theory and Practice of Compiler Writing, McGrawHil 1985.