COMPI, una herramienta interactiva para la enseñanza de construcción de compiladores

Francisco Ibañez, Daniel Díaz, Sandra Oviedo, Alejandra Otazu, , Maximiliano Alves Piñeiro
LISI- Instituto de Informática – Dpto. de Informática
FCEFyN - Universidad Nacional de San Juan
CUIM – Av. Ignacio de la Roza 590 (O), Rivadavia – J5402DCS San Juan
{ fibanez, ddiaz, soviedo, aleotazu, maxiap}@iinfo.unsj.edu.ar

Resumen

El objetivo es investigar sobre métodos, técnicas y herramientas para construir compiladores en pos de diseñar una herramienta interactiva y visual que se adapte a nuestro entorno cultural y que ayude al alumno en el proceso de enseñanza-aprendizaje, inherente al desarrollo de un compilador.

Palabras clave: Compiladores, Generación de compiladores, Técnicas de visualización de compilación.

Contexto

Este trabajo está inserto en la línea de investigación Enseñar, Aprender y Desarrollar Nuevos Lenguajes de Programación. Se ejecuta en el Instituto de Informática y en el Departamento de Informática de la Universidad Nacional de San Juan, y es parte del proyecto denominado "COMPI, una herramienta interactiva para la enseñanza de construcción de compiladores".

Introducción

Los educadores de las Ciencias de la Computación han puesto en cuestionamiento el contenido de los cursos tradicionales que forman parte de la columna vertebral del plan de estudios. Cursos tales como construcción de compiladores, deben proporcionar una visión coherente con el resto del plan de estudios de la institución [1].

Si observamos el plan de estudio de la carrera Ciencias de la Computación, las técnicas y metodología de enseñanza de la materia Construcción de Compiladores ha evolucionado notablemente. Los primeros cursos sobre compiladores ponían mucho énfasis en la teoría de *Parsing* y la traducción sintáctica directa. Sin embargo, con el pasar de los años se llegó a la conclusión que centrándose solamente en la teoría, no contribuye necesariamente a los estudiantes a construir compiladores útiles [2].

Bajo este propósito, se han desarrollado muchas herramientas para ayudar a construir compiladores. Algunas herramientas son de uso académico, y fueron explícitamente desarrolladas para educación, mientras que otras son de uso comercial. La mayoría de estas herramientas no presentan una interfaz gráfica, sino que son utilizables mediante consola de comandos. Es necesario un entrenamiento previo para poder utilizarla, consumiendo parte del semestre que dura el curso. Muchas de ellas no son interactivas y por lo tanto carecen de la opción de "paso a paso", que constituye una ayuda en el proceso de aprendizaje del alumno. Otro aspecto no menos importante es el cultural. Es difícil encontrar alguna herramienta desarrollada

que se adapte a nuestra cultura y/o nuestro plan de estudio.

Todas estas razones motivan el presente proyecto que tiene por objetivo investigar métodos, técnicas y herramientas para construir compiladores en pos de diseñar una herramienta interactiva y visual que se adapte a nuestro entorno cultural y que ayude al alumno en el proceso de enseñanza-aprendizaje inherente al desarrollo de un compilador.

Estructura de un Compilador

Los compiladores son programas de computadora que traducen de un lenguaje a otro. Un compilador toma como entrada un programa escrito en lenguaje fuente y produce un programa equivalente escrito en un lenguaje objeto. Un compilador se compone internamente de varias etapas, o fases, que realizan operaciones lógicas. Es útil pensar en estas fases como piezas separadas dentro del compilador, y pueden en realidad escribirse como operaciones codificadas separadamente aunque en la práctica a menudo se integran.

En la figura 1 se muestran las fases de un compilador tradicional y a continuación se describen brevemente cada una de ellas

Analizador léxico: lee la secuencia de caracteres de izquierda a derecha del programa fuente y agrupa las secuencias de caracteres en unidades con significado propio (componentes léxicos o "tokens" en inglés). Esta tarea es llevada a cabo por un proceso denominado comúnmente *scanner*.

Análisis sintáctico: determina si la secuencia de componentes léxicos sigue la sintaxis del lenguaje y obtiene la estructura jerárquica del programa en forma de árbol, donde los nodos son las construcciones de alto nivel del lenguaje. Esta tarea es llevada a cabo por un proceso denominado comúnmente como parser.

Análisis semántico: realiza las comprobaciones necesarias sobre el árbol sintáctico para determinar el correcto significado del programa. Esta tarea es llevada a cabo por un proceso denominado comúnmente como Análisis Semántico.

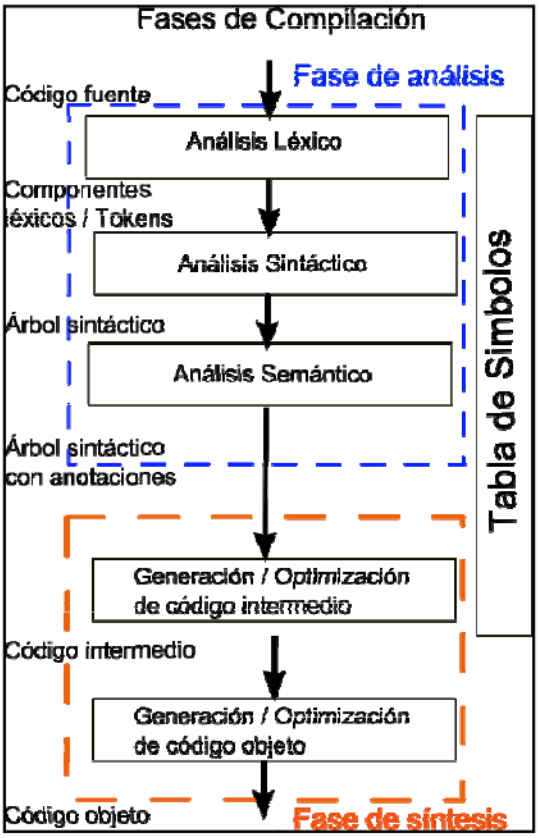

Figura 1. Fases de compilación

Generación y optimización de código intermedio: la optimización consiste en la calibración del árbol sintáctico donde ya no aparecen construcciones de alto nivel. Generando un código mejorado, ya no estructurado, más fácil de traducir directamente a código ensamblador o máquina, compuesto de un código de tres direcciones (cada instrucción tiene un operador, y la dirección de dos operandos y un lugar donde guardar el resultado),

también conocida como código intermedio.

Generación de código objeto: toma como entrada la representación intermedia y genera el código objeto. La optimización depende de la máquina. Es necesario conocer el conjunto de instrucciones, la representación de los datos (número de bytes), modos de direccionamiento, número y propósito de registros, jerarquía de memoria, encauzamientos, etc.

Evolución de los compiladores y sus técnicas de enseñanza

El contraste entre los primeros y los modernos compiladores es profundo. Los primeros compiladores se llevaron a cabo con sólo miles de líneas de código, por lo general fueron escritos en lenguajes de bajo nivel como el lenguaje ensamblador o C; los compiladores modernos normalmente constan de millones de líneas de código escritas en una variedad de lenguajes de programación (C, C + +, C #, F #, Java y Ocaml son actualmente las opciones más populares). Los primeros compiladores a menudo eran hechos a mano por una sola persona, los compiladores modernos son típicamente grandes proyectos de ingeniería de software que involucran equipos de programadores que utilizan componentes prediseñados y frameworks específicos.

En los primeros días de la informática había relativamente pocos lenguajes de programación, hoy en día hay miles. También en aquellos días existían relativamente pocas arquitecturas de máquina, algunas de ellas tales como CISC y RISC subsisten hasta hoy pero muchas otras han surgido, entre las cuales podemos destacar Vector, VLIW, y Multicore. Como consecuencia de esta evolución de los lenguajes y máquinas, no es posible cubrir en un curso de compiladores de un semestre la diversidad de tecnologías y algoritmos utilizados en un moderno compilador comercial.

Segun Aho [2] , uno de los autores del famoso libro de compiladores, "el libro del dragon" [3], a pesar de esta diversidad desconcertante de lenguajes y maquinas, todavía es posible dar un curso de compiladores que le otorgue al alumno varios beneficios educativos y una enorme satisfacción. A pesar de esta evolución, los conceptos básicos de análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio, entornos de ejecución, gestión de recursos y la generación de códigos están presentes en prácticamente todos los compiladores.

Breve descripción de COMPI

COMPI intenta ser una herramienta interactiva y visual para nuestro entorno cultural cuyo objetivo es ayudar al alumno en el proceso de enseñanza-aprendizaje inherente al desarrollo de un compilador.

COMPI está basado en el curso presentado [4]. Este curso presenta cada una de las fases de un compilador. Muestra los conceptos teóricos que subyacen a cada fase, así como la manera de ponerlos en práctica de manera eficiente. Los estudiantes deben escribir un pequeño compilador para el lenguaje Z#, el cual es un subconjunto del lenguaje de programación C#. El lenguaje de implementación usado es C #.

COMPI esta embebido en el código del compilador del lenguaje Z#, esencialmente sirve como entorno de programación, simulación y visualización de cada una de las fase de compilación. Así el alumno puede escribir un programa

fuente en Z# y con COMPI compilar y ejecutar paso a paso el programa. Tal como se muestra en la figura 2, se puede observar, la sentencia actual que se está ejecutando coloreada en rojo (ventana 1), la producción de la gramática (ventana 2), y el árbol de derivación (ventana 3).

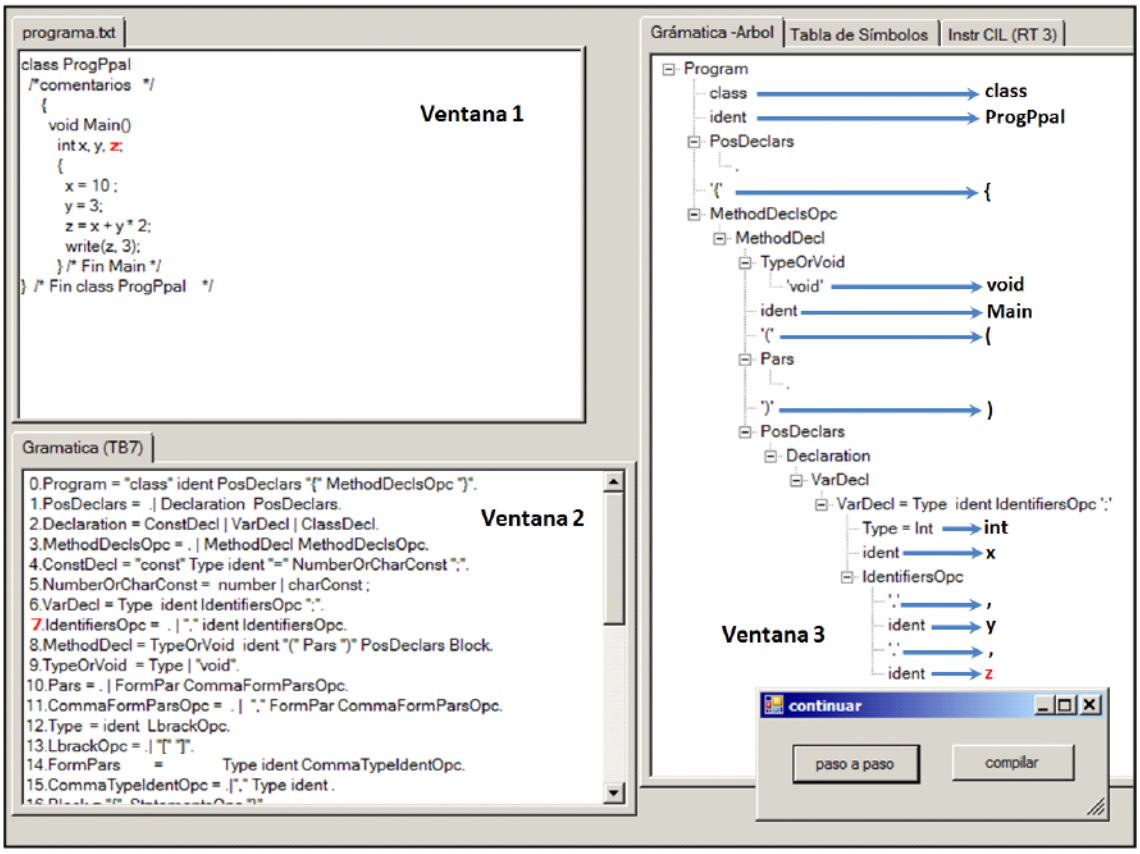

Figura 2. Ventanas principal de COMPI

Las pestañas *Tabla de Símbolos* e *Instrucciones CIL*, en la ventana 3, corresponden a desarrollos en los que estamos trabajando actualmente.

Líneas de Investigación, Desarrollo e Innovación

Las líneas de investigación están dedicadas a apoyar los procesos de Enseñar, Aprender y Desarrollar Nuevos Lenguajes de Programación. El eje principal es el desarrollo de una herramienta interactiva que apoye el proceso de enseñanza-aprendizaje, inherente al desarrollo de un compilador. Otros ejes incluyen, desarrollo de lenguaje de dominios específicos y técnicas de metaprogramación en el contexto del paradigma de programación orientado a características.

Resultados y Objetivos

Actualmente la herramienta soporta las fases de análisis léxico, sintáctico y

semántico. La herramienta es usada para dictar el curso de compiladores de la carrera de Ciencias de la Computación de nuestra facultad. El trabajo futuro, consiste en desarrollar la etapa de la generación de código. Está previsto también incluir un simulador de la maquina virtual, que ayude a comprender el lenguaje CIL (Common Intermediate Language), mediante la simulación de la ejecución del código CIL.

Formación de Recursos Humanos

El equipo de trabajo está constituido por 4 docentes y 2 alumnos avanzados. Existe una tesis de grado en curso.

Referencias

[1] W. M. Waite, "The Compiler Course in Today's Curriculum: Three Strategies," 2006, pp. 87-91.

[2] A. V. Aho, "Teaching the Compilers Course," *SIGCSE Bull.*, vol. 40, pp. 6-8, 2013/12/24/14:18:01 2008.

[3] A. V. Aho and R. U. J. D. Sethi, *Compilers: principles, techniques, and tools*. Reading, Mass. [u.a.: Addison-Wesley, 1988.

[4] W. Albrecht and M. Hanspeter. (2004, 15/03/2015). *Compiler Construction Course Concepts and Practical Application to .NET*. Available: http://dotnet.jku.at/courses/CC/#description