PROCESADORES DE LENGUAJE
Código: 13048. Titulación: Ingeniería Informática
Tipo: Troncal. Créditos: 6 Teoría + 3 Prácticas
Curso: 4. Anual. Año: 2005-2006
Profesores: Elena Díaz, Ariadna Fuertes, José Garcia
Departamento: Informática

Objetivos:

1. Conocer los componentes, estructura de un traductor y modo de operación, los conceptos teóricos, estructuras de datos, algoritmos y técnicas sobre los que se basan su diseño e implementación. Desde la fase de análisis con el análisis léxico, sintáctico y el semántico a la fase de síntesis con la generación y optimización del código, junto con la tabla de símbolos y la detección de errores interactuando entre estas fases.

2. Conocer cómo se ha formado el área que estudia el diseño de traductores y en qué otras se basa, especialmente su relación con la Teoría de Autómatas y Lenguajes Formales y los Lenguajes de Programación.

3. Mostrar la utilidad de los conceptos y técnicas que se abordan en la asignatura, por un lado no sólo por su aplicabilidad en el proceso de traducción, sino a cualquier problema que se pueda plantear en términos de una gramática y su reconocimiento y por otro lado porque ayudarán a conocer mejor los problemas que se plantean en el diseño e implementación de los lenguajes de programación y por tanto a hacer un mejor uso del lenguaje.

4. Conocer algunas de las herramientas que facilitan la generación de traductores, tanto de analizadores léxicos y sintácticos como para la generación de código.

5. Poner en práctica los conceptos y técnicas implementado en el laboratorio un traductor de un sencillo pero representativo lenguaje de programación, de manera que tengan la oportunidad de aplicar las técnicas aprendidas en cursos anteriores, de escribir un programa de cierta complejidad en equipo, lo que requiere que se mantenga el software.

Prerrequisitos:
La asignatura supone conocimientos previos sobre Teoría de Autómatas y Lenguajes Formales y Estructuras de Datos, así como del lenguaje de programación C/C++.

Temario:

Tema 1. Introducción.(4 horas)

1. ¿Qué es un traductor?. Tipos de traductores.
2. Programas del sistema relacionados con los traductores.
3. Fases del proceso de traducción.
4. Mil lenguajes y mil máquinas, ¿un millón de traductores?:
   a) *Front-end* y *Back-end*. Traducción cruzada.
   b) *Bootstraping.*
5. Principales estructuras de datos en un traductor.
6. Herramientas de ayuda a la traducción.
7. Relación con otras áreas.
8. Aplicaciones de los sistemas de traducción.

Tema 2. Análisis Léxico. (5 horas)

1. Funciones del analizador léxico.
2. Especificación de los componentes léxicos: expresiones regulares.

3. Reconocimiento de los componentes léxicos: autómatas finitos.
4. Implementación de un analizador léxico:
    a) Dirigido por tabla.
    b) Mediante bucles anidados.
5. Aspectos prácticos en la implementación de un análisis léxico:
    a) Representación de los componentes léxicos.
    b) Principio de máxima longitud.
    c) Palabras reservadas versus identificadores.
    d) Gestión del buffer de entrada.
6. Errores léxicos y su tratamiento.
7. Generadores automáticos de analizadores léxicos: *Lex.*

Tema 3. Introducción al Análisis Sintáctico.(5 horas)

1. El proceso de análisis sintáctico.
2. Especificación sintáctica de los lenguajes de programación:
    a) Gramáticas independientes del contexto versus expresiones regulares.
    b) Gramáticas independientes del contexto versus gramáticas dependientes del contexto.
3. Gramáticas limpias y bien formadas.
4. Derivaciones y árboles sintácticos.
5. El problema de la ambigüedad en las gramáticas. Eliminación:
    a) Mediante reglas de precedencia.
    b) Mediante transformación.
6. Clasificación de los métodos de análisis sintáctico.

Tema 4. Análisis Sintáctico Descendente. (8 horas)

1. Análisis descendente: el autómata predice/concuerda.
2. Problemas en el análisis descendente:
    a) Recursividad a izquierdas y su eliminación.
    b) Indeterminismo en las alternativas. Factorización.
3. Gramáticas LL(1):
    a) Los conjuntos de PRIMEROS y SIGUIENTES.
    b) La condición LL(1).
4. Analizador sintáctico predictivo recursivo.
5. Analizador sintáctico predictivo dirigido por tabla.
6. Recuperación de errores en los analizadores descendentes:
    a) Los conjuntos de predicción y sincronización.
    b) Recuperación en modo de pánico.
7. Limitaciones de los métodos descendentes.

Tema 5. Análisis Sintáctico Ascendente. (9 horas)

1. Análisis ascendente: el autómata *desplaza/reduce.*
2. Gramáticas LR:

a) Tipos de conflictos: *desplaza/reduce* y *reduce/reduce*.
b) Prefijo viable y mango de una forma secuencial derecha.
c) Elemento LR(0).
d) Autómata finito de elementos LR(0).

3. Análisis LR(0):
   a) Algoritmo de análisis LR(0).
   b) Tablas de análisis sintáctico LR(0).
   c) Limitaciones del análisis LR(0).
4. Análisis SLR(1):
   a) Algoritmo de análisis SLR(1).
   b) Resolución de conflictos por precedencia de operadores.
   c) Limitaciones del análisis SLR(1).
5. Análisis LR(1) y LALR(1):
   a) Elemento LR(1).
   b) Autómata finito de elementos LR(1)
   c) Algoritmo de análisis LR(1).
   d) Agrupación de estados: análisis LALR(1).
6. Generadores de analizadores sintácticos LALR(1): *Yacc*
7. Recuperación de errores en los analizadores ascendentes
8. Métodos ascendentes versus métodos descendentes.

Tema 6. Análisis Semántico.(11 horas)

1. La fase de análisis semántico.
2. Especificación semántica de un lenguaje:
   a) Concepto de atributo, tipos de atributos y tipos de enlace.
   b) Gramáticas de atributos.
   c) Ecuaciones de atributos.
   d) Árbol de análisis sintáctico anotado.
3. Métodos para la evaluación de atributos:
   a) Basados en grafos de dependencias.
   b) Basados en reglas: atributos sintetizados y heredados.
   c) Cálculo de atributos durante el proceso de análisis sintáctico.
4. La tabla de símbolos:
   a) Organización y mantenimiento de la tabla de símbolos.
   b) Reglas de ámbito de referencia.
5. Comprobación de tipos:
   a) Representación de expresiones de tipos mediante árboles.
   b) Equivalencia estructural y por nombre.
6. Inferencia de tipos.
7. Errores semánticos.
8. Esquemas de traducción dirigidos por la sintaxis.

Tema 7. Entorno de ejecución.(6 horas)

1. Aspectos del lenguaje fuente que determinan el entorno de ejecución.
2. Organización de la memoria durante la ejecución.
3. Registros de activación.
4. Tipos de entornos de ejecución:
    a) Entornos estáticos.
    b) Entornos basados en pila.
    c) Entornos basados en un montículo.
5. Técnicas para la asignación dinámica de la memoria:
    a) Asignación explícita de bloques de tamaño fijo y variable.
    b) Desasignación: las referencias suspendidas y la recolección de basura.
6. La gestión del ámbito de los identificadores:
    a) Acceso a nombres no locales.
    b) El anidamiento de las funciones.
7. El paso de parámetros.

Tema 8. Generación de Código Intermedio y Optimización. (12 horas)

1. Introducción.
2. Tipos de representaciones intermedias:
    a) Código de 3-direcciones.
    b) Código-P.
3. Traducción dirigida por la sintaxis a código intermedio:
    a) Código intermedio como un atributo sintetizado.
4. Código intermedio para referenciar estructuras de datos.
5. Generación de código para expresiones y sentencias de control:
    a) Proposiciones de asignación.
    b) Expresiones aritméticas.
    c) Expresiones booleanas.
    d) Sentencias de control.
    e) Funciones.
6. Optimización de código:
    a) Bloques básicos y optimización local.
    b) Eliminación de subexpresiones comunes.
    c) Eliminación de código muerto.
    d) Transformaciones aritméticas.
    e) Empaquetamiento de variables temporales.
    f) Mejoras en lazos.

Bibliografía:

Aho, A.V., Sethi, R., Ullman, J.D. (1990) *Compiladores: principios, técnicas y herramientas* (Addison-Wesley Iberoamericana, Madrid, ISBN: 0-201-62903-8, 1990), traducción castellana de *Compilers: Principles, Techniques and Tools* (Addison-Wesley, Reading, Massachusetts, 1986). *Copias*: 3, castellano, 2 en inglés *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 AHO

> El famoso libro del dragón de A. Aho y J. Ullman. Este texto ha marcado la docencia sobre compiladores desde su primera edición a mitad de los 80. Esta versión revisada y actualizada aborda de una forma detallada el diseño e implementación de compiladores, los conceptos teóricos y técnicas sobre los que se basa la traducción desde un punto de vista formal y práctico. El texto comienza con una introducción de las ideas que subyacen en el proceso de la compilación y posteriormente ilustra estas ideas construyendo un compilador de una pasada. El resto del libro amplía los conceptos presentados en los dos primeros capítulos y trata temas avanzados como el análisis sintáctico, la traducción dirigida por la sintaxis, la verificación de tipos, la organización del entorno de ejecución y la generación y optimización de código. Incluye una amplia variedad de problemas y ejercicios con un nivel gradual de dificultad. Todos los contenidos de la materia reciben un tratamiento completo y apropiado. La exposición de la materia en el aula sigue en muchos casos sus algoritmos y notaciones fielmente. Si bien, a veces la poco acertada traducción a castellano de algunos de los términos, el excesivo detalle y formalismo en que se presentan los contenidos hace que su lectura no sea clara e incluso a veces en algunas secciones un tanto aburrida.

Louden, K.C. (1997) *Compiler Construction: Principles and Practice* (PWS Publishing Company, Boston, Massachusetts, ISBN: 0-534-93972-4, 1997). *Copias*: 2, inglés *Lugar*: Biblioteca Campus *Signatura*: 681.3.06 LOU

> Este libro combina de forma proporcionada y muy amena un estudio detallado de los aspectos teóricos y prácticos subyacentes en el diseño moderno de compiladores, junto con ejemplos prácticos muy bien escogidos. Tal y como dicen sus autores, para entender los aspectos prácticos de un compilador, uno necesita tener un buen conocimiento de la teoría, y realmente para apreciar la teoría, es importante verla en acción en un entorno real o casí real. Con su libro estos autores han conseguido proporcionar el apropiado balance entre teoría y práctica, con suficiente detalle, incluso a nivel de implementación, para dar al lector una completa idea de las técnicas sin abrumarlo. Además, incluye gran número de ejercicios sobre problemas específicos de los lenguajes de programación y una descripción completa, junto con el código C, de un pequeño pero completo compilador para un sencillo lenguaje, generado con las técnicas presentadas en cada capítulo. Este libro recoge de una forma completa y con suficiente nivel de detalle, pero sin extenderse tanto como el texto de Aho y Ullman, todos los contenidos del temario. Todo el libro está muy bien escrito y se lee con mucha facilidad. Son especialmente claros y concisos los capítulos en los que se describen la implementación de de analizadores LR, la generación de código intermedio y la compilación de funciones y el tratamiento de tipos. Es un libro muy valioso para completar o contrastar apuntes.

Vivancos, E., Moreno, L., Gisbert, V y Benedí, J.M. (2000) *Compiladores I: una introducción a la fase de análisis* (Servicio de Publicaciones de la Universidad Politécnica de Valencia, ISBN: 84-7721-915-X, 2000). *Copias*: 3 (pedidas) *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 VIV

> Este libro proporciona una adecuada y proporcionada guía de inicio a las técnicas de diseño e implementación de traductores, pero que tan sólo se centra en la fase de análisis: análisis léxico, sintáctico y semántico. La forma de exposición es clara y concisa. Es especialmente válido por el tratamiento a los métodos de análisis sintáctico LL y LR, por la extensa y diversa serie de ejercicios resueltos y propuestos (con soluciones incluidas) y por un apéndice donde se describen con cierto detalle las herramientas automáticas de generadores de analizadores (*Lex* y *Yacc*). Su principal limitación es que apenas trata la verificación de tipos, los esquemas de traducción dirigidos por la sintaxis, la organización y gestión de la tabla de símbolos, y no se abordan los temas relacionados con la fase de síntesis (la generación y optimización de código) ni el entorno de ejecución. A pesar de ello, es un libro a tener en cuenta.

Iñesta, J.M., García, P. y Gracia, I. (1998) *Técnicas básicas para el diseño de compiladores* (Servicio de Publicaciones de la Universitat Jaume I, ISBN: 84-8021-259-4, 1998). *Copias*: 1 *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 IÑE

Este es un texto que a modo de apuntes recoge de forma clara y concisa una introducción a las técnicas de diseño de compiladores en cuanto a la fase de análisis se refiere. Con un estilo directo se abordan las ideas subyacentes en la especificación y reconocimiento de la estructura léxica y sintáctica de un lenguaje. Es un libro útil por su fácil lectura y la capacidad de esquematizar y sintetizar en apenas 90 páginas los conceptos y técnicas relevantes. Si bien debe ser complementado con alguno de los textos básicos expuestos anteriormente, ya que no se tratan los temas relacionados con la fase de síntesis de código, ni algunos aspectos semánticos claves como la verificación de tipos. Cabe resaltar la colección de ejercicios resueltos y propuestos por su extensión, variedad y originalidad.

Bennet, J.B.(1996) *Introduction to Compiling Techniques: a first course using ansi C, Lex and Yacc* (McGraw-Hill Publishing Company, England, ISBN: 007709221X, 1996). *Copias*: 2 en inglés *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 BEN

Este texto aborda los conceptos relacionados con la construcción de compiladores especialmente desde un punto de vista práctico, con un estilo directo y simple. Destaca por los detalles a nivel de implementación que le hacen ser especialmente útil para la parte de laboratorio. En él se tratan no sólo los temas relacionados con el análisis léxico, sintáctico y semántico, sino también con la implementación de técnicas de recuperación de errores, la comprobación de tipos, la generación de código y las técnicas de optimización local, todo ello ilustrado sobre la implementación de un pequeño traductor de un sencillo lenguaje de alto nivel a lenguaje ensamblador para una máquina abstracta con un conjunto reducido de instrucciones, registros y de direccionamiento. Dedica un capítulo especial a describir las herramientas *Lex* y *Yacc* para la generación de analizadores.

Teufel, B., Schmidt, S., Teufel, T. (1995) *Compiladores: conceptos fundamentales* (Addison-Wesley Iberoamericana, Wilmington, Delaware, ISBN: 0-201-65365-6, 1995). *Copias*: 1 *Lugar*: Bibiloteca Departamento

Libro muy recomendable como una introducción bastante concisa, pues en 179 páginas en castellano se tocan todos los aspectos necesarios para comprender de qué manera operan los sistemas de traducción. Por contra, es pobre en ejemplos y el código en metalenguaje que se ofrece no siempre es afortunado. No se trata con el detalle necesario la generación de código y se pasan por alto la recuperación de errores, la optimización y el entorno de ejecución.

Pratt, T.W. (1998) *Lenguajes de Programación. Diseño e Implementación* (Prentice-Hall Hispanoamericana, ISBN: 0-13-678012-1, 1998). *Copias*: 1 *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 PRA

Tercera edición del libro *Programming Languages. Design and Implementation* de 1984. Es un libro sobre lenguajes de programación en el que algunos conceptos propios de la traducción se abordan en 60 páginas en el capítulo tercero. Es útil como referencia a los aspectos de los lenguajes de programación que van a dictar la complejidad del traductor: tipos de datos y de declaraciones, estructuración, control de secuencia y de subprogramas, recursividad, herencia, encapsulamiento, paso de parámetros, etc. Además de repasar los distintos paradigmas. Especialmente válido para el tema dedicado al entorno de ejecución donde se trata la organización y gestión de la memoria.

Kelley, D. (1995) *Teoría de Autómatas y Lenguajes Formales* (Prentice Hall International, UK, ISBN: 0-13-518705-2, 1995). *Copias*: 2 *Lugar*: Bibiloteca Campus *Signatura*: 681.3.06 KEL

Libro adecuado para corregir la deficiencia en los aspectos relacionados con los conceptos formales sobre lenguajes y gramáticas que asumen muchos de los libros sobre construcción de compiladores y que son necesarios para el desarrollo de la asignatura. En concreto, la construcción de expresiones regulares a partir de una determinada especificación y de los autómatas finitos deterministas para su reconocimiento, la especificación de autómatas mediante diagramas y tablas de estado, la conversión de autómatas no-deterministas a deterministas y la minimización del número de estados, las gramáticas independientes del contexto y los autómatas a pila para su reconocimiento, la notación Backus-Naur-Form y su versión extendida, los diagramas sintácticos para la definición de una gramática, las gramáticas limpias y bien formadas, los tipos de derivaciones de una sentencia, así como el problema de la ambigüedad en las gramáticas.

Referencias avanzadas

Muchnick, S. (1997) *Advanced Compiler Design and Implementation* (Morgan Kaufmann Publishers, San Francisco California, ISBN: 1-55860-320-41, 1997). *Copias*: 1 *Lugar*: Bibiloteca Departamento *Signatura*: 2209

Este texto es de carácter claramente avanzado. En más de sus 700 páginas se abordan en profundidad los conceptos, técnicas y tendencias actuales en la construcción de compiladores, dedicando especial atención a la fase de síntesis de código. En él se examina en detalle la implementación y gestión de la tabla de símbolos locales y globales; el diseño e implementación del entorno de ejecución con los registros de activación, la construcción y mantenimiento de la pila de ejecución de activaciones, los métodos de paso de parámetros, la estructura de los procedimientos y los algoritmos para la recolección automática de basura y la compactación de la memoria; la generación automática de código dirigida por la sintaxis y mediante correspondencia de árboles; las técnicas de optimización global mediante métodos de análisis de flujo de control y de datos, así como las optimizaciones dependientes de la máquina. También se discuten en detalle varios compiladores comerciales para diferentes lenguajes de alto nivel examinado su entorno de ejecución, el rango de las arquitecturas objeto, los aspectos de diseño, las representaciones de código intermedio y las optimizaciones. En concreto de la familia de procesadores de *Intel* y la arquitectura *Sparc* de *Sun*.

Grune, D. y Jacobs, C. (1990) *Parsing Techniques: a practical guide* (Ellis Horwood Limited, England, ISBN: 0-13-651431-6, 1990). *Copias*: 1 *Lugar*: Bibiloteca Departamento *Signatura*: 0919

Texto especialmente dedicado a las técnicas de análisis sintáctico tanto descendentes como ascendentes, direccionales y no direccionales. Aunque con un carácter un tanto formal es fácil de leer y referencia clave para aspectos avanzados en este tema. Trata especialmente detallada cada uno de los tipos de análisis, los tipos de gramáticas LL y LR, sus reconocedores y los problemas de ambigüedad. Incluye un capítulo completo a las técnicas de recuperación de errores. Los ejemplos son escasos y no se incluyen ejercicios resueltos ni propuestos.

Evaluación:

Para evaluar la teoría se realizarán dos exámenes. Un primer parcial (opcional) en febrero cuya materia será eliminatoria. Un segundo examen en junio donde se evaluará sólo la segunda parte de la asignatura si se aprobó el primer parcial o toda la asignatura si se suspendió el primer parcial. La nota final será la media de ambos parciales. Se realizarán 10 sesiones prácticas de 3 horas y se realizará un examen. El 60 % de la nota vendrá dada por los trabajos que se irán realizando en cada sesión de prácticas y el 40 % restante será la nota obtenida en un examen de laboratorio de carácter obligatorio. Para realizar cualquier promedio es necesario obtener un mínimo de 4 en cualquier parte. En cuanto a septiembre, si se aprobaron las prácticas de laboratorio en junio se guardará la nota. Respecto a la teoría, no se guardará la nota del primer parcial.

Tutorías:

Las tutorías serán los martes de 11:30 a 2:30 en mi despacho (Despacho 4302, Dpto. de Informática, Bloque D, Facultad de Física, tercer piso). Mi dirección de correo electrónico Elena.Diaz@uv.es.

Páguina WEB:

`http://informatica.uv.es/iiguia/2000/PL/`