ST0270 Lenguajes Formales y Compiladores
Introducción al curso

Andrés Sicard Ramírez

Universidad EAFIT

Semestre 2024-1

Pacto pedagógico

Como miembros de la Universidad EAFIT, nos comprometemos a actuar de manera íntegra siguiendo los más altos estándares éticos y morales.

- Respeto
- Tolerancia
- Honradez
- Compromiso

Pacto pedagógico

Página web del curso

http://www1.eafit.edu.co/asr/cursos/st2070-lenguajes-formales-y-compiladores

Pacto pedagógico

Página web del curso

`http://www1.eafit.edu.co/asr/cursos/st2070-lenguajes-formales-y-compiladores`

Programa de la materia

El programa de la materia está en EAFIT Interactiva.

Pacto pedagógico

Página web del curso

http://www1.eafit.edu.co/asr/cursos/st2070-lenguajes-formales-y-compiladores

Programa de la materia

El programa de la materia está en EAFIT Interactiva.

Conducto regular, fechas y porcentajes de las evaluaciones

La información está en la página web del curso.

Pacto pedagógico

Página web del curso

http://www1.eafit.edu.co/asr/cursos/st2070-lenguajes-formales-y-compiladores

Programa de la materia

El programa de la materia está en EAFIT Interactiva.

Conducto regular, fechas y porcentajes de las evaluaciones

La información está en la página web del curso.

Responsabilidad compartida

- Profesor
- Estudiantes

Pacto pedagógico

Orientaciones para el curso

- Se recomienda ocho horas de trabajo por semana (dos horas por cada hora de clase).
- Las clases son presenciales.
- La evaluación a la docencia es obligatoria.
- Se recomienda revisar periódicamente los canales de comunicación institucionales (EAFIT Interactiva, correo institucional, Microsoft Teams).
- Se pueden sacar las notas de clase (en papel o digitales) durante los exámenes parciales.
- El proyecto final no se puede realizar de manera individual y se debe realizar máximo entre dos estudiantes.

Acerca del curso

En el curso estableceremos relaciones entre los modelos de computación, los lenguajes formales, las gramáticas y los compiladores.

Acerca del curso

En el curso estableceremos relaciones entre los modelos de computación, los lenguajes formales, las gramáticas y los compiladores.

Programa del curso

- Teoría de los lenguajes formales: Definiciones y operaciones básicas (Kozen 2012, Lectura 2).
- Lenguajes regulares y autómatas finitos (Kozen 2012, Lecturas 3–6, 11).
- Lenguajes libres de contexto y autómatas a pila (Kozen 2012, Lecturas 19–21, 23, 24).
- Máquinas de Turing y clasificación de Chomsky (Kozen 2012, Lecturas 28, 29).
- Análisis sintáctico descendente (Aho et al. 2006, § 4.4.1–4.4.5).
- Análisis sintáctico ascendente (Aho et al. 2006, § 4.5.1–4.5.5, § 4.6.1–4.6.4, § 4.6.6).
- Semántica de traducción y análisis estático (Crespi Reghizzi et al. 2019, § 5.1–5.4, § 5.5–5.7).

Abstracción

Acerca de la abstracción

En relación con la abstracción, Kozen (2012, pág. 5) escribió:

> *When studying real-world phenomena, one becomes aware of recurring patterns and themes that appear in various guises. These guises may differ substantially on a superficial level but may bear enough resemblance to one another to suggest that there are common underlying principles at work. When this happens, it makes sense to try to construct an abstract model that captures these underlying principles in the simplest possible way, devoid of the unimportant details of each particular manifestation. This is the process of* ***abstraction****. Abstraction is the essence of scientific progress, because it focuses attention on the important principles, unencumbered by irrelevant details.*

Abstracción

Acerca de la abstracción

En relación con la abstracción, Aho et al. (2006, pág. 15) escribieron:

> *Compiler design is full of beautiful examples where complicated real-world problems are solved by abstracting the essence of the problem mathematically. These serve as excellent illustrations of how abstractions can be used to solve problems: take a problem, formulate a mathematical abstraction that captures the key characteristics, and solve it using mathematical techniques. The problem formulation must be grounded in a solid understanding of the characteristics of computer programs, and the solution must be validated and refined empirically.*

Modelos de computación

Modelos de computación en orden creciente de poder computacional

- Memoria finita
  - (i) Autómatas finitos
  - (ii) Expresiones regulares
- Memoria finita y una pila: Autómatas a pila
- Memoria linealmente acotada: Autómatas linealmente acotados
- Memoria no acotada (infinita)
  - (i) Máquinas de Turing
  - (ii) Cálculo lambda

Modelos de computación

Modelos de computación en orden creciente de poder computacional

- Memoria finita
  - (i) Autómatas finitos (McCulloch y Pitts 1943).
  - (ii) Expresiones regulares (Kleene 1956).
- Memoria finita y una pila: Autómatas a pila (Oettinger 1961).
- Memoria linealmente acotada: Autómatas linealmente acotados (Myhill 1960).
- Memoria no acotada (infinita)
  - (i) Máquinas de Turing (Turing 1936-1937).
  - (ii) Cálculo lambda (Church 1936).

Observación

Algunos de los modelos anteriores fueron definidos antes de que los computadores fueran creados.

Gramáticas y lenguajes

Descripción

Una **gramática** es un meta-lenguaje para describir lenguajes.

Gramáticas y lenguajes

Descripción

Una **gramática** es un meta-lenguaje para describir lenguajes.

Acerca de los lenguajes formales

En relación a los lenguajes formales, Crespi Reghizzi et al. (2019, pág 5) escribieron:

> *For a language to be **formalized** (or **formal**), the form of its sentences (or syntax) and their meaning (or semantics) must be precisely and algorithmically defined.*

Gramáticas y lenguajes

Jerarquía de Chomsky (1959) para las gramáticas

Las gramáticas en la tabla están en orden creciente de acuerdo a la clase de lenguaje que generan.

Tipo de gramática	Clase de lenguaje generado
3	Regulares
2	Libres (independientes) de contexto
1	Dependientes del contexto
0	Recursivamente enumerables

Gramáticas y modelos de computación

Correspondencia entre las gramáticas y los modelos de computación

Un tipo de gramática genera una clase de lenguajes y un modelo de computación reconoce una clase de lenguajes.

Tipo de gramática	Clase de lenguaje	Modelo de computación
3	Regulares	Autómatas finitos
2	Libres (independientes) de contexto	Autómatas a pila
1	Dependientes del contexto	Autómatas linealmente acotados
0	Recursivamente enumerables	Máquinas de Turing

Gramáticas y modelos de computación

Correspondencia entre las gramáticas y los modelos de computación

Un tipo de gramática genera una clase de lenguajes y un modelo de computación reconoce una clase de lenguajes.

Tipo de gramática	Clase de lenguaje	Modelo de computación
3	Regulares	Autómatas finitos
2	Libres (independientes) de contexto	Autómatas a pila
1	Dependientes del contexto	Autómatas linealmente acotados
0	Recursivamente enumerables	Máquinas de Turing

Observación

La correspondencia indicada en la tabla es un resultado de la abstracción.

Compiladores

Descripción

Usualmente la **compilación** es la translación de un programa en un lenguaje de programación (lenguaje **fuente**) en un programa ejecutable en un lenguaje de máquina* (lenguaje **destino**).

*Lenguaje binario que es leído, interpretado y ejecutado por la CPU.

Compiladores

Descripción

Usualmente la **compilación** es la translación de un programa en un lenguaje de programación (lenguaje **fuente**) en un programa ejecutable en un lenguaje de máquina* (lenguaje **destino**).

Un **compilador** es un programa que realiza una compilación.

*Lenguaje binario que es leído, interpretado y ejecutado por la CPU.

Compiladores

Otros programas requeridos para generar un programa ejecutable

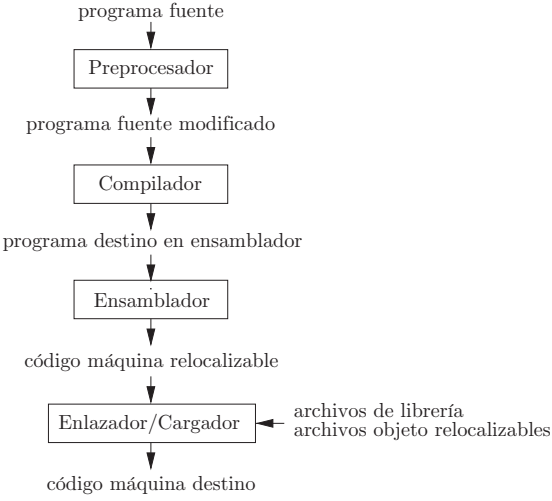

Figura 1.5 de (Aho et al. 2008)

Compiladores

Partes de un compilador

Las partes generales de un compilador son la parte de análisis (o *front-end*) y la parte de síntesis (o *back-end*) (Aho et al. 2008, pág 4):

(i) «La parte del **análisis** divide el programa fuente en componentes e impone una estructura gramatical sobre ellas. Después utiliza esta estructura para crear una representación intermedia del programa fuente... La parte del análisis también recolecta información sobre el programa fuente y la almacena en una estructura de datos llamada **tabla de símbolos**.»

(ii) «La parte de la **síntesis** construye el programa destino deseado a partir de la representación intermedia y de la información en la tabla de símbolos.»

Compiladores

Fases de un compilador

Véase figura en la siguiente diapositiva.

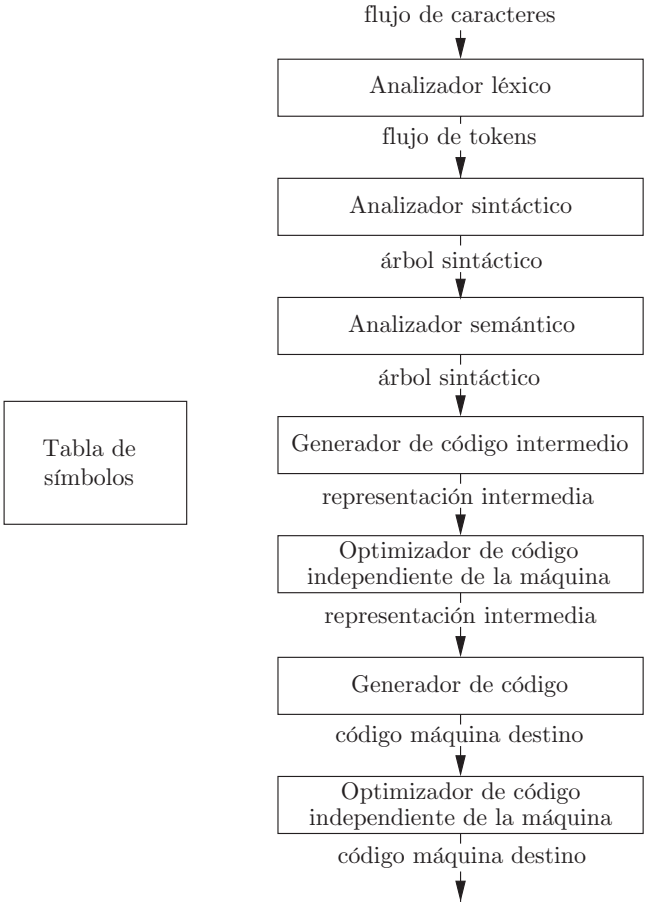

Figura 1.6 de (Aho et al. 2008).

Referencias


Aho, Alfred V., Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman [1986] (2006). *Compilers: Principles, Techniques, & Tools*. 2.ª ed. Addison-Wesley (vid. págs. 8, 9, 11).


— [1986] (2008). *Compiladores: Principos, Técnicas y Herramientas*. Trad. por Alfonso Vidal Romero Elizondo. 2.ª ed. Pearson Educación (vid. págs. 21, 22, 24).


Church, Alonzo (1936). «An Unsolvable Problem of Elementary Number Theory». En: *American Journal of Mathematics* 58.2, págs. 345-363. DOI: 10.2307/2371045 (vid. págs. 12, 13).


Crespi Reghizzi, Stefano, Luca Breveglieri y Angelo Morzenti [2009] (2019). *Formal Languages and Compilation*. 3.ª ed. Texts in Computer Science. Springer. DOI: 10.1007/978-3-030-04879-2 (vid. págs. 8, 9, 14, 15).


Kleene, Stephen C. (1956). «Representation of Events in Nerve Nets and Finite Automata». En: *Automata Studies*. Ed. por C. E. Shannon y J. McCarthy. Vol. 34. Annals of Mathematics Studies. Princenton University Press, págs. 3-42 (vid. págs. 12, 13).


Kozen, Dexter C. [1997] (2012). *Automata and Computability*. Third printing. Undergraduate Texts in Computer Science. Springer. DOI: 10.1007/978-1-4612-1844-9 (vid. págs. 8-10).

Referencias

- McCulloch, W. S. y W. Pitts (1943). «A Logical Calculus of The Ideas immanent in Nervous Activity». En: *Bulletin of Mathematical Biophysics* 5.4, págs. 115-133. DOI: 10.1007/BF02478259 (vid. págs. 12, 13).
- Oettinger, Anthony G. (1961). «Automatic Syntactic Analysis and The Pushdown Store». En: *Proc. Symp. App. Math.* Vol. 12, págs. 104-129 (vid. págs. 12, 13).
- Turing, Alan M. (1936-1937). «On Computable Numbers, with an Application to the Entscheidungsproblem». En: *Proceeding of the London Mathematical Society* s2-42, págs. 230-265. DOI: 10.1112/plms/s2-42.1.230 (vid. págs. 12, 13).