UNIVERSIDAD DE GRANADA

Modelos de Computación

Temario

Ismael Sallami Moreno

Recursos Ingeniería Informática y Ade

Licencia

Modelos de Computación

Ismael Sallami Moreno

Índice general

I Teoría **7**

1 Introducción a la Computación **9**

1.1 Fundamentos y Orígenes **9**

1.2 Elementos Básicos: Lenguajes Formales **10**

1.3 Operaciones Esenciales **11**

1.4 Generación de Lenguajes **12**

Parte I

Teoría

Capítulo 1

Introducción a la Computación

1.1 Fundamentos y Orígenes

1.1.1 Preguntas Clave: Definición, problemas resolubles, y Modelos de Computación

La Ciencia de la Computación se fundamenta en la búsqueda de respuestas a preguntas trascendentales sobre los límites de la resolución automática de problemas. Entre las cuestiones centrales se encuentran:

- **¿Qué puede ser resuelto de forma automática?** Esta pregunta indaga sobre la existencia de algoritmos para problemas dados, es decir, procedimientos mecánicos sistemáticos que garantizan una terminación para cualquier entrada válida.
- **¿Qué puede ser resuelto de forma eficiente?** Más allá de la mera resolubilidad, este interrogante aborda la viabilidad práctica de los algoritmos en términos de recursos computacionales como tiempo y espacio.
- **¿Qué estructuras son comunes en la computación con símbolos y cómo pueden ser procesadas?** Esta línea de investigación se enfoca en el estudio de patrones en conjuntos de cadenas y en los mecanismos formales para su manipulación.

Para abordar estas preguntas, se desarrollan **Modelos de Computación**. Estos modelos son abstracciones matemáticas de un dispositivo de cómputo, que permiten analizar sus capacidades y limitaciones de manera formal. Un concepto clave en este ámbito es el de **autómata**, concebido como un sistema algebraico que procesa información. Un autómata, en su forma más general, se define por sus conjuntos de estados, señales de entrada y señales de salida, junto con las operaciones que gobiernan sus transiciones y respuestas. El estudio de estos modelos, como las Máquinas de Turing y sus simplificaciones (autómatas finitos, autómatas con pila), constituye la base de la Teoría de la Computabilidad y de Lenguajes Formales.

1.1.2 Breve Historia: Desde precursores hasta Autómatas y el Problema de la Parada

Los fundamentos de la computación se remontan a los trabajos de lógicos y matemáticos como Russell, Hilbert y Boole, quienes sentaron las bases del formalismo matemático moderno. En las décadas de 1930 y 1940, figuras como Alan Turing y Alonzo Church formalizaron la noción de çomputabilidad". Turing, en particular, propuso un modelo teórico, la **Máquina de Turing**, que se considera la definición de un dispositivo de cómputo universal.

En su búsqueda por resolver los 23 problemas propuestos por David Hilbert, la comunidad científica demostró que ciertos problemas son **indecidibles**, es decir, no admiten una solución algorítmica

general. El ejemplo canónico es el **Problema de la Parada (Halting Problem)**.

Definición 1.1 (Problema de la Parada). No existe un programa de ordenador universal, llamémoslo `Stops(P, x)`, que pueda tomar como entrada cualquier otro programa `P` y unos datos `x` y determinar, en un tiempo finito, si `P` terminará su ejecución con la entrada `x` o si entrará en un bucle infinito.

La imposibilidad de resolver este problema fue un resultado trascendental que estableció los límites fundamentales de lo que puede ser computado.

Paralelamente, el desarrollo de los primeros lenguajes de programación como FORTRAN, COBOL y LISP en los años 50 impulsó el estudio de la sintaxis. En esta década, Noam Chomsky, junto con Michael Rabin y Dana Scott, formalizó la teoría de los **Autómatas y Lenguajes Formales**, estableciendo una jerarquía de gramáticas y máquinas que reconocen distintas clases de patrones.

1.2 Elementos Básicos: Lenguajes Formales

1.2.1 Alfabetos

La teoría de lenguajes formales se construye a partir de conceptos básicos. El más fundamental es el de alfabeto.

Definición 1.2 (Alfabeto). Un **alfabeto** es un conjunto **finito** y no vacío de elementos denominados **símbolos** o **letras**.

Nota 1.1 . Se denotan los alfabetos con letras mayúsculas como Σ, A, B, etc., y sus símbolos con letras minúsculas como a, b, c o números.

Ejemplo 1.1 . Algunos alfabetos comunes son:

- El alfabeto binario: $\Sigma = \{0, 1\}$.
- El alfabeto de letras minúsculas: $\Sigma = \{a, b, c, \ldots, z\}$.
- Un alfabeto de vectores: $B = \{\langle 0, 0\rangle, \langle 0, 1\rangle, \langle 1, 0\rangle, \langle 1, 1\rangle\}$.

1.2.2 Palabras

Definición 1.3 (Palabra). Una **palabra** (también llamada **cadena** o **string**) sobre un alfabeto A es una sucesión finita de símbolos de A. La **longitud** de una palabra u, denotada por $|u|$, es el número de símbolos que la componen.

Nota 1.2 . Las palabras se denotan con letras minúsculas como u, v, w, x, y, z. El conjunto de todas las palabras posibles sobre un alfabeto A se denota como A^*.

Definición 1.4 (Palabra Vacía). La **palabra vacía**, denotada por ϵ (o a veces λ), es la única palabra de longitud cero, $|\epsilon| = 0$. Es un elemento de A^* para cualquier alfabeto A.

Nota 1.3 . El conjunto de todas las palabras no vacías sobre un alfabeto A se denota como A^+. Por lo tanto, $A^+ = A^* \setminus \{\epsilon\}$.

El conjunto de palabras A^* con la operación de concatenación forma una estructura algebraica conocida como monoide, donde la palabra vacía ϵ es el elemento neutro.

1.2.3 Lenguajes

Definición 1.5 (Lenguaje). Un **lenguaje** L sobre un alfabeto A es cualquier subconjunto de A^*, es decir, $L \subseteq A^*$. Las palabras que pertenecen al lenguaje se denominan a menudo çadenas del lenguaje.º "frases".

Ejemplo 1.2 . Dado el alfabeto $A = \{a, b\}$:

- $L_1 = \{a, b, \epsilon\}$ es un lenguaje finito con tres palabras.
- $L_2 = \{a^i b^i \mid i \geq 0\}$ es un lenguaje infinito que contiene palabras con igual número de a's seguidas de b's.
- $L_3 = \{uu^{-1} \mid u \in \{a, b\}^*\}$ es el lenguaje de los palíndromos de longitud par.
- $L_4 = \{a^{n^2} \mid n \geq 1\}$ es el lenguaje de cadenas de a's cuya longitud es un cuadrado perfecto.

Una propiedad fundamental del conjunto de todos los lenguajes sobre un alfabeto no vacío es su **no numerabilidad**. Mientras que el conjunto de todas las palabras A^* es numerable, el conjunto de todos sus subconjuntos (es decir, el conjunto de todos los lenguajes, $\mathcal{P}(A^*)$) es no numerable. Esto tiene una profunda implicación: existen más lenguajes que programas para describirlos o reconocerlos, lo que demuestra que debe haber problemas (lenguajes) que no son computacionalmente resolubles.

1.3 Operaciones Esenciales

1.3.1 Sobre Palabras

Existen varias operaciones fundamentales que se pueden aplicar a las palabras.

Definición 1.6 (Concatenación). Dados $u = a_1 \dots a_n$ y $v = b_1 \dots b_m$ dos palabras sobre un alfabeto A, su **concatenación**, denotada uv, es la palabra $a_1 \dots a_n b_1 \dots b_m$.

La concatenación es asociativa y tiene a ϵ como elemento neutro ($u\epsilon = \epsilon u = u$).

Definición 1.7 (Iteración). La **iteración n-ésima** de una palabra u, denotada u^n, es la concatenación de u consigo misma n veces. Se define formalmente como $u^0 = \epsilon$ y $u^{i+1} = u^i u$ para $i \geq 0$.

Definición 1.8 (Palabra Inversa). La **palabra inversa** de $u = a_1 \dots a_n$, denotada u^{-1}, es la palabra $a_n \dots a_1$.

Nota 1.4 . La operación de inversión no es un homomorfismo. Por ejemplo, si $f(u) = u^{-1}$, entonces $f(uv) = (uv)^{-1} = v^{-1}u^{-1}$, que en general es distinto de $f(u)f(v) = u^{-1}v^{-1}$.

1.3.2 Sobre Lenguajes

Al ser los lenguajes conjuntos de palabras, heredan las operaciones de conjuntos como la **unión** ($L_1 \cup L_2$), **intersección** ($L_1 \cap L_2$) y **complementario** ($\bar{L} = A^* \setminus L$). Adicionalmente, se definen operaciones específicas.

Definición 1.9 (Concatenación de Lenguajes). La **concatenación** de dos lenguajes L_1 y L_2 es el lenguaje $L_1L_2 = \{uv \mid u \in L_1, v \in L_2\}$.

Esta operación es asociativa y tiene como elemento neutro el lenguaje $\{\epsilon\}$.

Definición 1.10 (Clausura de Kleene y Clausura Positiva). Dado un lenguaje L, su **clausura de Kleene** (o estrella), denotada L^*, es la unión de todas sus potencias:

$$L^* = \bigcup_{i \geq 0} L^i = L^0 \cup L^1 \cup L^2 \cup \dots$$

La **clausura positiva**, denotada L^+, se define de manera similar pero excluyendo la potencia cero:

$$L^+ = \bigcup_{i \geq 1} L^i = L^1 \cup L^2 \cup \dots$$

donde $L^0 = \{\epsilon\}$ y $L^{i+1} = LL^i$.

Se cumple que $L^+ = L^*$ si y solo si $\epsilon \in L$. Si $\epsilon \notin L$, entonces $L^* = L^+ \cup \{\epsilon\}$. Si L es un lenguaje no vacío, L^* siempre es infinito.

1.4 Generación de Lenguajes

1.4.1 Gramáticas: Definición y Lenguaje Generado

Los lenguajes, especialmente los infinitos, requieren un mecanismo finito para su descripción. Las gramáticas generativas, introducidas por Noam Chomsky, son uno de dichos mecanismos.

Definición 1.11 (Gramática Generativa). Una **gramática generativa** es una tupla $G = (V, T, P, S)$, donde:

- V es un alfabeto finito de **variables** o símbolos no terminales.
- T es un alfabeto finito de **símbolos terminales**, disjunto de V.
- P es un conjunto finito de **reglas de producción** de la forma $\alpha \to \beta$, donde $\alpha, \beta \in (V \cup T)^*$ y α contiene al menos una variable.
- $S \in V$ es el **símbolo inicial** o axioma.

Una palabra β se **deriva** de α en un paso ($\alpha \Rightarrow \beta$) si existe una regla $\gamma \to \phi \in P$ y α puede escribirse como $\alpha_1\gamma\alpha_2$, de tal forma que $\beta = \alpha_1\phi\alpha_2$. La relación de derivación en múltiples pasos se denota por $\Rightarrow^*$.

Definición 1.12 (Lenguaje Generado). El **lenguaje generado** por una gramática G, denotado $L(G)$, es el conjunto de todas las palabras compuestas exclusivamente por símbolos terminales que pueden derivarse a partir del símbolo inicial S:

$$L(G) = \{u \in T^* \mid S \Rightarrow^* u\}$$

.

Ejemplo 1.3 . Sea la gramática $G = (\{S\}, \{a, b\}, \{S \to aSb, S \to \epsilon\}, S)$. Esta gramática genera el lenguaje $L(G) = \{a^i b^i \mid i \geq 0\}$. Por ejemplo, para generar $aabb$: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aab\epsilon b = aabb$.

1.4.2 Jerarquía de Chomsky

Noam Chomsky clasificó las gramáticas en cuatro tipos, estableciendo una jerarquía basada en las restricciones impuestas sobre sus reglas de producción.

Definición 1.13 (Jerarquía de Chomsky). La jerarquía de Chomsky se define de la siguiente manera:

- **Tipo 0 (Sin restricciones):** Corresponde a cualquier gramática generativa. Estas gramáticas generan los lenguajes **recursivamente enumerables**, que son aquellos reconocibles por una Máquina de Turing.
- **Tipo 1 (Dependientes del contexto):** Todas las producciones son de la forma $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, con $A \in V$, $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$ y $\beta \neq \epsilon$. Se permite la regla $S \to \epsilon$ si S no aparece en la parte derecha de ninguna regla. Generan los lenguajes **dependientes del contexto**. Un ejemplo es el lenguaje $\{a^n b^n c^n \mid n \geq 1\}$.
- **Tipo 2 (Independientes del contexto):** Todas las producciones son de la forma $A \to \alpha$, con $A \in V$ y $\alpha \in (V \cup T)^*$. Generan los lenguajes **independientes del contexto**.
- **Tipo 3 (Regulares):** Todas las producciones son de la forma $A \to uB$ o $A \to u$, donde $A, B \in V$ y $u \in T^*$. Generan los **lenguajes regulares**.

Esta jerarquía establece una relación de inclusión estricta entre las familias de lenguajes generados, denotadas por $\mathcal{L}_i$ para el tipo i:

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

. Por ejemplo, todo lenguaje regular es también independiente del contexto, pero no a la inversa.

Bibliografía

[1] Ismael Sallami Moreno, **Estudiante del Doble Grado en Ingeniería Informática + ADE**, Universidad de Granada, 2025.