Teoría de la Computación para Ingeniería de Sistemas: un enfoque práctico

Prof. Hilda Contreras

6 de febrero de 2018

Índice general

Capítulo 1

Introducción

La teoría de la Computación es un poco más antigua que las computadoras electrónicas. Uno de sus pioneros, Alan Turing, pudo anticipar el poder de las computadoras a través de un modelo conceptual en 1936. Otras disciplinas como la matemática, filosofía, lingística, biología e ingeniería eléctrica intervienen para completar sus teorías. Las teorías de bases son dos: Teoría de Autómatas y Teoría de los Lenguajes Formales. En general, la Teoría de la Computación facilita la comprensión de muchas áreas de la ciencia de la computación (como los compiladores), además:

1. Se utiliza en el diseño y construcción de aplicaciones importantes de software y hardware.
2. Ayuda a comprender que esperar del software.
3. Permite deducir si es posible resolver un problema (determinar los límites de la computación).

Además, la comprensión de estas teorías representa en la práctica un conjunto de herramientas muy útiles como alternativas simples y eficientes para resolver problemas.

1.1. Marco histórico de la teoría de la computación

Esta teoría surge a partir de importantes antecedentes del área de las matemáticas, la cual estuvo en una profunda cirsis a finales del siglo XIX y primera mitad del XX. La lógica y las matemáticas trabajaron juntas en la formalización de la ciencia de la computación, los siguientes hechos son los mas resaltantes:

- Entre 1910 y 1923, Bertrand Russell elabora el "Principia Matemática"presentando la relación entre la lógica y la matemática pura.
- En 1928, el matemático alemán David Hillbert, presenta a sus colegas como reto demostrar tres proposiciones de gran generalidad sobre su ciencia:
  - La matemáticas son completas.
  - Las matemáticas son consistentes.
  - Las matemáticas son decidibles.

- En 1928, el matemático Ackermann presenta también el problema de la decisión en un estudio sobre la lógica de primer orden.

- En 1930, el matemático checo Kurt Gdel prueba que las matemáticas no pueden ser completas y consistentes al mismo tiempo (teorema de la Incompletitud).

- En 1936, el matemático americano Alonzo Church responde negativamente en un artículo al tercer problema propuesto por Hillbert, el problema decisorio (Teorema de Church y λ-cálculo).

- En ese mismo año 1936, el matemático inglés Alan Turing da una respuesta también negativa a esa tercera cuestión, sus resultados son más consistentes que los obtenidos por Church. La demostración de Turing se basa en principios completamente básicos y elementales. Turing enuncia el problema de decisión de la siguiente forma: "Buscar un algoritmo para determinar si una conclusión particular puede derivarse de ciertas premisas con el uso de reglas de prueba". Da una noción precisa del concepto de algoritmo, como aquello que pueda ser realizado por una máquina abstracta, la Máquina de Turing. De este modo, Alan Turing con su máquina universal capaz de realizar el trabajo de cualquier otra máquina, mediante la lectura de su descripción en una cinta, delinea el diseño de un computardor.

- En 1969, S. Cook extiende el estudio de Tuning. Cook separa aquellos problemas que pueden ser solucionados de aquellos que en principio pueden ser solucionados pero que en la práctica toman demasiados recursos (Complejidad computacional).

- En 1937, Claude Shannon presenta su tesis de licenciatura en el MIT, estableciendo el paralelismo entre la lógica de Boole y los circuitos de transmisión.

- En 1943, McCulloch-Pitts desarrolla unas máquinas simples, en cuanto su funcionamiento, que fueron conocidas como autómatas finitos de actividad nerviosa, para modelar el funcionamiento de una neurona biológica.

- En 1956, Moore publica el primer estudio riguroso sobre autómatas. Con anterioridad, debido al desarrollo de los primeros computadores, se habían estudiado diversos métodos para la síntesis de circuitos secuenciales (Huffman en 1954 y Mealy en 1955). En los años 60 es donde se realizan la mayor parte de trabajos sobre la teoría de los autómatas finitos.

- En 1956, N. Chomsky comienza el estudio formal de las gramáticas (como generadoras de lenguajes). Formaliza matemáticamente los conceptos de gramática generativa o reglas para la construcción las frases de un lenguaje. Enuncia la teoría sobre el origen y la naturaleza de los lenguajes naturales. Estas herramientas fueron empleadas para la formalización de los lenguajes de computación: Teoría de los Lenguaje Formales. A finales de los años 50 se relacionan los autómatas, las gramáticas y los lenguajes (Jerarquía de Chomsky).

1.2. Conceptos básicos: Alfabeto, Palabra o Cadena, Lenguaje y Problema

La Teoría de la Computación puede entenderse como un paradigma o un enfoque para resolver problemas. Este paradigma tiene los siguientes conceptos básicos sobre los cuales se fundamenta: Alfabeto, Palabra o Cadena, Lenguaje y Problema. Para todas estas definiciones debe tenerse presente primero el concepto de un *Conjunto*, como la estructura que las soporta.

Conjunto

Colección (no importa el orden) de objetos (elementos del mismo tipo) sin repetición. Existen 2 (dos) tipos de representación de los conjuntos:

- Por Extensión, se enumeran todos sus elementos, por ejemplo: {0, 1}
- Por Comprensión, se define de manera formal las características de sus elementos, por ejemplo: { i | Para todo j entero i = 2j }

El conjunto vacío se representa así: ϕ. La pertenencia de un elemento a un conjunto: A = {0, 1, 2} y 1 ϵ A o también se escribe 1 en A. [1]
El subconjunto: A $\subseteq$ B significa que todo elemento de A esta en B. Si A $\subseteq$ B y A $\neq$ B entonces escribimos A $\subsetneq$ B (subconjunto propio).
Operadores sobre conjuntos:

- Unión: A $\bigcup$ B = {x | x en A o x en B}
- Intersección: A $\bigcap$ B = {x | x en A y x en B}
- Diferencia: A - B = {x | x en A y x no está en B}
- Conjunto Producto ó Producto Cartesiano: A $\times$ B {(x,z) | x en A y z en B}. Si A tiene n elementos y B tiene m elementos entonces A $\times$ B tiene n $\times$ m elementos y A^* tiene 2^n elementos.
- Conjunto de Potencias 2^A ó A^*, es el conjunto de todos los subconjuntos de A. También A* es denominado Clausura de Kleene de A. Las definiciones formales son: A* = $\bigcup_{i=0}^{|A|} A^i$ y A$^+$ = $\bigcup_{i=1}^{|A|} A^i$ (clausura positiva que no incluye al conjunto vacío). Donde A^i representa todos los subconjuntos de A con i elementos.

Ejemplo:
Dados los conjuntos A = {1,2} y B = {2,3}, se realizan las siguientes operaciones sobre conjuntos:

- A $\bigcup$ B = {1, 2, 3}
- A $\bigcap$ B = {2}
- A - B = {1}
- A $\times$ B = {(1,2),(1,3),(2,2),(2,3)}
- $2^A = A^*$ = {ϕ,{1},{2},{1,2}}

[1] Nota: ϵ es un símbolo que puede ser utilizado por algunos autores para denotar a la cadena vacía.

En la Teoría de la Computación se visualizan las entradas, salidas y procesos de un dominio de aplicación como un tipo de problema particular: *resolver la pertenencia de un elemento a un conjunto*. El problema, los elementos y el tipo de conjunto que se van a utilizar se definen en base a los siguientes conceptos básicos:

1. **Alfabeto**: Denotado con Σ (sigma). Conjunto finito de símbolos no vacíos. Por ejemplo: Alfabeto binario Σ = {0,1}. Símbolo: es una entidad abstracta que no se define, por ejemplo números, letras, etc.

2. **Cadena** (Palabra): secuencia finita de símbolos pertenecientes a un alfabeto (yuxtapuestos = uno detrás del otro). Por ejemplo: Cadena binaria "0111001". Sobre las cadenas se definen:

   a) Cadena Vacía λ (lambda): contiene cero (0) símbolos, puede construirse a partir de cualquier alfabeto.

   b) Longitud de una Cadena: número de posiciones de la cadena ocupadas por símbolos del alfabeto. Denotado con el pipe |, por ejemplo para el alfabeto binario, las longuitudes de las siguientes cadenas son |01101|= 5 y $|\lambda|$=0

   c) Concatenación de Cadenas : x y y son cadenas entonces xy denota la concatenación. Por ejemplo: x = 011 y = 1110, la concatenación de xy = "0111110"

   d) Potencias de un Alfabeto: Conjunto de todas las cadenas formadas a partir de un alfabeto de cierta longitud.

   - Σ^k Cadenas de longitud k formadas por símbolos del alfabeto Σ. Para el alfabeto binario Σ = {0,1}, Σ^2 = {00, 11, 10, 01}
   - Σ^* Conjunto de todas las cadenas formadas por símbolos del alfabeto Σ. Se define por $\Sigma^* = \Sigma^0 \bigcup \Sigma^i$. Donde $\Sigma^0 = \{\lambda\}$ y $\Sigma^i = \bigcup_{i=1}^{\infty} \Sigma^i$. Además se denota el $\Sigma^+ = \Sigma^* - \{\lambda\}$

   e) Sufijo, Prefijo: cualquier cadena al final o comienzo respectivamente de la cadena. Por ejemplo para la cadena w=abca, formada de $\Sigma = \{a, b, c\}$, se tiene:

   - Los prefijos de w= λ, a, ab, abc, abca
   - Los sufijos de w= λ, a, ca, bca, abca

   f) Prefijo (Sufijo) propio, es cualquier Prefijo (Sufijo) que no sea la misma cadena.

3. **Lenguaje**: Conjunto de cadenas de símbolos del mismo alfabeto. El lenguaje formado por toda las cadenas posibles de un alfabeto se denota por Σ^* (Sigma asterisco).

   - ϕ es el lenguaje vacío, es un lenguaje para cualquier alfabeto.
   - $\{\lambda\}$ es un lenguaje formado por la cadena vacía. Es un lenguaje para cualquier alfabeto Σ.

   Por Ejemplo: $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{00, 1\}$ [2]

   - Σ_1^* = {λ, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, ... }
   - Σ_2^* = {λ, 00, 1, 001, 100, 0000, 11, 00001, 00100, 10000, 1001, 0011, 111, 000000,... }
   - Las cadenas formadas con Σ_2 al aplicarle la cardinalidad: |001| = 2 y |1100| = 3

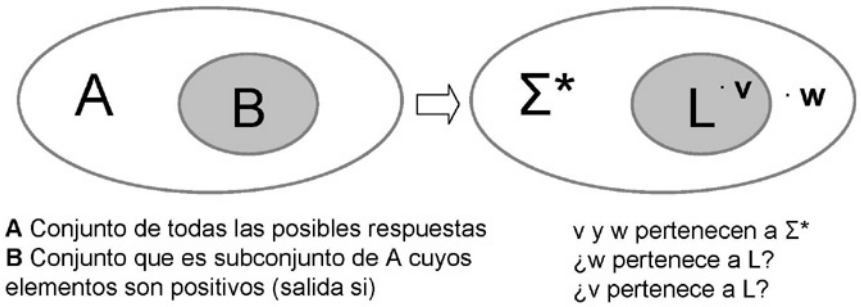

Figura 1.1: Definición de Problema en la Teoría de la Computación

4. **Problema**: consiste en determinar la pertenencia de una cadena sobre un lenguaje. Este tipo de problema es llamado un *Problema de Decisión*: función con salida binaria (si, no) aplicada sobre conjuntos. Por ejemplo ver figura 1.1: se define un conjunto A con todas las posibles respuestas y un conjunto B que es subconjunto de A cuyos elementos son positivos (salida si, que pertenecn a B). Expresado en términos de la teoría de la computación, si Σ es un alfabeto y L es un lenguaje de Σ^*, entonces el problema sobre L es el siguiente:

Dada una cadena w que pertenece a Σ^*, decidir si ¿w pertenece o no a L?

1.3. Jerarquía de Chomsky

Noan Chomsky en los años 50, fue el creador de una jerarquía de lenguajes según las gramáticas que los generan. Esta gramática está organizada en base al poder generativo formal de la gramática y donde se destaca 4 tipos de gramáticas (denominados tipo 0, tipo 1, tipo 2 y tipo 3), cada una definida por la clase de reglas que contiene. El cuadro 1.1, muestra una jerarquía implicativa, de modo que los lenguajes definidos por gramáticas del tipo-i incluyen los lenguajes del tipo-(i+1). Dicho de otra forma: Tipo 3 $\subset$ Tipo 2 $\subset$ Tipo 1 $\subset$ Tipo 0. Por tanto las gramáticas de tipo 0 son las más poderosas y las de tipo 3 las más restringidas.
La Teoría de Autómatas y la Teoría de las Gramáticas formales tienen una estructura similar aunque traten de aspectos diferentes - procedimientos y gramáticas, respectivamente-. Las gramáticas generan un tipo de lenguaje y los autómatas reconocen un tipo de lenguaje, es decir el punto de conexión son los lenguajes. Esta es la razón por la cual se utiliza la jerarquía de Chomsky en la solución de problemas: un problema consiste en determinar si una cadena pertenece a un lenguaje, entonces si se conoce el tipo de lenguaje se puede determinar con que máquina reconocerlo o con cual tipo de gramática generarlo.

[2]Los símbolos del alfabeto pueden ser de diferentes tamaños

Cuadro 1.1: Jerarquía de Chomsky

Tipo	Lenguaje	Máquina	Gramática G = (V,T,P,S)
0	Recursivamente enumerable	Máquina de Turing	Gramática sin restricciones $\alpha \rightarrow \beta$ (α, β en (V $\bigcup$ T)*, α contiene una variable)
1	Dependiente del Contexto	Autómata linealmente acotado	Gramática sensible al contexto $\alpha \rightarrow \beta$ (α, β en (V $\bigcup$ T)*, α contiene una variable, $\|\beta\| \geq \|\alpha\|$)
2	Independiente del Contexto	Autómata de Pila	Gramática libre de contexto A $\rightarrow \alpha$ (A en V y α en (V $\bigcup$ T)*)
3	Lenguaje Regular	Autómata finito	Gramática Regular A $\rightarrow$ aB, A $\rightarrow$ a, (A,B en V y a en T)

1.4. Los problemas en la Teoría de la Computación

Teoría de la Computación trata de una teoría o formalismo general, la idea es aplicarla sobre todos los problemas que se pueden calcular en cualquier máquina. La teoría usa modelos de máquinas abstractas, pues no tendría sentido usar las máquinas reales que cambian constantemente en el tiempo y que además son diferentes y complejas. Entonces ¿qué sucede con los problemas?, ¿los problemas también son abstractos?.
Lo que se pretende calcular, es decir el *Problema* o *Cálculo*, puede ser de cualquier tipo, naturaleza y condición. Los que han programado han podido resolver, de muchas formas, con un lenguaje de programación múltiples tipos de problemas. Además hay muchos lenguajes de programación e incluso algunos específicos para ciertas características de los problemas. Se sabe que los problemas no son iguales, entonces, ¿cómo esta Teoría puede generalizar a todos los problemas para aplicar sus reglas y sus modelos abstractos?. Bien, es lo primero que hay que entender, lo que hace la Teoría de la Computación es formalizar un solo tipo de problema, por tanto se debe convertir, transformar o reducir *cualquier* problema a lo que se llaman un "problema de decisión".
Un problema de decisión es un problema que puede dar una respuesta positiva o negativa (si o no). Un problema de decisión, en términos matemáticos, es equivalente a decidir si un elemento pertenece o no a un conjunto dado. El conjunto equivalente al problema esta formado por los elementos para el cual la respuesta es positiva (si). La figura 1.2 muestra el esquema general de los problemas.
El Entscheidungsproblem [1] (problema de decisión en alemán) se define como "dada una

Figura 1.2: Problema de decisión en Teoría de la Computación

frase del cálculo de predicados de primer orden, decidir si ella es un teorema", es el problema de

decisión que motivó a Turing y a Church a dar origen a la Teoría de la Computación en 1936. Un ejemplo típico de este tipo de problema es la siguiente pregunta: ¿Es primo un número entero dado?, y una instancia de este problema sería: ¿Es 17 un número primo?. Se trata de entender al 17 como un elemento del conjunto de todos los números enteros y se quiere saber si pertenece al conjunto de los números primos.
Lo anterior lleva a la pregunta de si realmente es posible reducir o redefinir *todos* los problemas a la forma de un problema de decisión. Para lo cual le sugiero platearse ejemplos e investigar esta posibilidad, pues es en base a esta generalización de problemas sobre la cual funciona la Teoría de la Computación. Esto puede parecer una desventaja, como lo menciona Hopcroft [2]:

> ... Un aspecto potencialmente poco satisfactorio de esta definición de problema es que normalmente no se piensa en los problemas como cuestiones de decisión (¿es o no verdadero lo siguiente?) sino como solicitudes para calcular o transformar algunos valores de entrada (encontrar la mejor forma para realizar esta tarea) ...

Sin embargo, según la computabilidad y complejidad las soluciones al problema de decisión y al problema original se diferencian a lo sumo por un factor lineal y se puede decir que vale la pena esta generalización de los problemas a cambio del formalismo que necesita la Ciencia de la Computación.
Esta teoría de la computación entonces va a definir los problemas en función de un lenguaje. Justamente, el cuadro 1.1 muestra la jerarquía de Chomsky en donde se relacionan los tipos de lenguajes (problemas) con el tipo de máquina (modelo) que reconoce cadenas que pertenecen a cada tipo y a la gramática (modelo) que genera las cadenas que pertenecen al tipo de lenguaje. Dicha jerarquía estable una relación inclusiva entre los lenguajes, asi, el lenguaje Tipo 3 $\subseteq$ Tipo 2 $\subseteq$ Tipo 1 $\subseteq$ Tipo 0. Esto quiere decir que los lenguajes Tipo 0 son el tipo de lenguaje que abarca a todos los que se pueden resolver a través de una Máquina de Turing (la más poderosa).

1.5. Preguntas y respuestas, ejercicios resueltos y propuestos

Preguntas y respuestas

1. ¿Qué es una teoría? [3] Dentro del proceso de producción científica: Una Teoría es un conjunto de medios de representación conceptual y simbólica (explicativo de los hechos), que tiene además un conjunto de reglas de inferencia que permiten la previsión de los datos de hecho.

2. ¿Qué es una máquina abstracta? Es una máquina que no existe físicamente, pero que ejecuta instrucciones sobre una entrada y produce una salida, habitualmente se han empleado en el estudio de la computabilidad. Ejemplos: Autómatas Finitos, Autómatas a Pila, Autómatas linealmente acotados y Máquina de Turing. La computadora que utilizamos es compleja, cambia con el tiempo, por tanto una teoría basada en especificaciones de hardware no sería muy útil, por tanto los Modelos de Computación están basados en máquinas abstractas. (matemática).

3. ¿Qué es el poder de una máquina abstracta?, ¿qué significa que una máquina sea más poderosa que otra?. En el contexto de la Teoría de la Computación, una máquina es

poderosa en la medida que soluciona problemas sin importar como lo hace (no importa la eficiencia en tiempo y recursos, sino su efectividad en lograr resolver el problema). En los términos del paradigma, si una máquina reconce lenguajes más complejos será más pododera, según la Jerarquía de Chomsky la máquina que reconoce el lenguaje tipo 0 es la más poderosa.

Ejercicios resueltos

1. Diferencia entre el operador clausura de Kleene aplicado sobre un conjunto y sobre un alfabeto. Por ejemplo: El conjunto A = {a,b} cuál es el A* y para el alfabeto B = {a,b} cuál es B*.
   La diferencia se debe a que el operador estrella o clausura de Kleene es polimórfico. Aunque A y B son el mismo conjunto, se interpretan de forma diferente para el operador. En el caso de A es un conjunto de elementos y A* es el conjunto de todos sus subconjuntos, A es finito y A* también, A* = {ϕ, {a}, {b}, {a, b}}. En el caso de B, es interpretado como un alfabeto, es decir un conjunto finito de símbolos. Aplicando el mismo operador a B nos dá un conjunto infinito, B* = {λ, a, b, aa, ab, ba, bb, aaa, aab, aba, ... } (se está formando el conjunto de todas las cadenas, incluyendo la cadena vacía, que se puede formar con el alfabeto B).

2. Considere el proceso de compilación de un programa escrito en un lenguaje de programación. Identifique el alfabeto, cadena, lenguaje y problema según la Teoría de la Computación.
   El problema de la compilación de un programa escrito en algún lenguaje de programación puede ser visto a través del paradigam de lenguajes formales que contiene la Teoría de la Computación. El compilador no es más que un sistema que es capaz de ver si una entrada (el código fuente) esta bien escrito y sigue las reglas del lenguaje de programación que está reconociendo. Es decir, el compilador es una máquina reconocedora de textos (códigos fuente) que pertenecen al conjunto de todos los códigos escritos en un lenguaje de programación. De esta forma se definen:

   - El alfabeto Σ: es el conjunto de símbolos que puede escribirse en un teclado de computador con el cual puede escribirse texto (caracteres alfanuméricos, caracteres especiales, espacio, etc.).
   - Cadena de entrada w: La entrada del compilador visto como una máquina reconocedora, no es más que cualquier texto que podamos escribir con el teclado del computador (es decir con Σ). Cada cadena w se puede generar concatenando elementos de Σ, de esta maneras tenemos infinitas cadenas (textos)y entonces Σ^* es infinito.
   - El lenguaje L: es el subconjunto de Σ^* formado por las cadenas (texto) que siguen las reglas de un lenguaje de programación específico. El compilador reconoce un lenguaje L infinito pues las reglas de un lenguaje de programación (aunque son finitas) permiten escribir infinitos programas que formarían parte de L.
   - El problema P: El compilador resuelve un problema de decisión al determinar si un texto de entrada compila o no. Decir que un texto compila significa que pertenece al lenguaje L de todas los posibles e infinitos códigos que pueden escribirse en el lenguaje de programación. Decir que un texto o cadena de entrada en Σ^* no

compila es decir que no pertenece a L, es decir que no sigue las reglas del lenguaje de programación.

Comentario: Evidentemente, el compilador es un sistema que además de resolver el problema de decisión aporta como valor agregado un listado de errores en el caso de no poder compilar y el código objeto en el caso de compilar. El problema P (compilación), es un problema de pertenencia de un conjunto: ¿pertenece el programa w al conjunto infinito de programas L escritos en un lenguaje de programación particular?. Si L fuese finito resultaría sencillo de resolver, pero como afortunadamente (para quienes programan) L es infinito (se pueden escribir infinitos programas utilizando un lenguaje de programación) se debe buscar el modelo finito de L que permita reconocer un programa bien escrito en un lenguaje de programación dado. Este modelo finito es justamente lo que la Teoría de la Computación permite hacer.

Ejercicios propuestos

1. ¿Qué es un lenguaje formal?, Diferencia con los lenguajes naturales
2. Contestar verdadero (V) o falso (F):
   a) Σ^* es un lenguaje formado por el alfabeto.
   b) La clausura de Kleene Σ^* sobre el alfabeto Σ puede ser en algunos alfabetos un conjunto finito.
   c) Dado el alfabeto Σ = {a,b} y Δ= {0,1}, la cadena w = a1b0 pertenece al alfabeto $\Sigma \bigcup \Delta$
   d) Según la Jerarquía de Chomsky un lenguaje esta asociado un nivel de clasificación de problemas asociado a una máquina
   e) Un algoritmo es el conjunto de instrucciones que puede calcular
   f) No existe un lenguaje infinito L en el alfabeto Σ = {a,b} para el cual L sea diferente a L* (L $\neq$ L*)
3. Sea A, B y C lenguajes de un alfabeto Σ. En cada una de los ítems siguientes se indican la expresión de 2 lenguajes. Señale si son siempre iguales y justifique (indique contraejemplos en caso de ser apropiado)
   a) A (B $\bigcup$ C) y AB $\bigcup$ AC
   b) A*B* y (AB)*
4. Enumere al menos 5 lenguajes (que no sean lenguajes de programación o idiomas) que haya utilizado. Para cada uno de ellos identifique el alfabeto y de ejemplo de una cadena del lenguaje, describa el lenguaje.

Bibliografía

[1] Alonzo Church: A note on the Entscheidungsproblem. Journal of Symbolic Logic. 1 (1936). pp 40 - 41.

[2] Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. Introducción a la Teoría de Autómatas y Lenguajes Formales. PrenticeHall, 2002.

[3] Chacín, M. y Padrón, J. (1994) Qué es teoría. Investigación y Docencia. Caracas: Publicaciones del Decanato de Postgrado, USR.

[4] Seymour Lipschutz (1970) Teoría de Conjuntos y Temas afines. macGraw-Hill