REPORTE DE PRÁCTICA 0

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

RESUMEN LENGUAJES FORMALES

ALUMNO: Ordaz Rangel David

DOCENTE: Dr. Eduardo Cornejo Velázquez

Resumen de Lenguajes Formales - Práctica 0

Un **lenguaje** está basado en un **alfabeto**, el cual es un conjunto finito de símbolos.

Definiciones Básicas

- **Alfabeto** (Σ): Conjunto finito de símbolos. Ejemplo: $\Sigma = \{0, 1\}$ (alfabeto de números binarios).
- **Palabra**: Secuencia finita y ordenada de símbolos pertenecientes a un alfabeto.
- **Cadena vacía** (λ): Palabra de longitud 0 sin ningún elemento.
- **Longitud** ($|x|$): Número de símbolos en una palabra x.
- **Apariciones de un símbolo en una palabra**: Se representa como $|x|_a$, indicando la cantidad de veces que un símbolo a aparece en la palabra.
- **Ordenación**: Según la menor cantidad de caracteres, si son la misma longitud, se ordena alfabéticamente
- **Combinación**: Ejemplo: $\Sigma^1 binarios = \{0, 1\}$
- **Clausura de Klenn**: Representa el infinito sobre un alfabeto, pues engloba todas las combinaciones posibles Ejemplo: $\Sigma^* = \{0, 1\}$

Operaciones

- **Concatenación**: Juntar una palabra detrás de la otra.
- **Potencia**: Es el resultado de concatenar una palabra consigo mismo n veces.
- **Prefijos**: Sílabas que añadimos delante de la palabra primitiva.
- **Sufijos**: Sílabas que añadimos detrás de la palabra primitiva.
- **Segmentos**: Sílabas que se encuentran en nuestra palabra.
- **Reverso**: Palabra leída de derecha a izquierda. *Ejemplo*: `Hola` = `aloH`.

Lenguaje

Definición: Subconjunto de Σ^*.

Lenguaje infinito: *Ejemplo*: $L = \{ax \mid x \in \Sigma^* \text{ español}\}$.

Lenguaje finito: *Ejemplo*: $L = \{x \in \Sigma^* \text{ español} \mid |x| < 5\}$.

Operaciones Booleanas de Conjuntos

- **Producto**: Concatenación de cada una de las palabras del primer lenguaje, con cada una de las palabras del segundo lenguaje.
- **Potencia**: L^n, producto de un lenguaje por sí mismo una cantidad de veces n.
- **Cierre**: Unión de todas las potencias de un lenguaje desde L^0.
- **Cociente**: Aquellas palabras obtenidas después de eliminar un símbolo v al lenguaje.
- **Homomorfismo**: Aplicación de un alfabeto hacia otro mediante reglas de asignación para la "traducción" de sus símbolos, puede funcionar como herramienta para encriptar.

Autómata

Una máquina abstracta que modela y describe procesos de cálculo, como un robot matemático que sigue instrucciones para procesar datos. Está formada por una serie de estados y transiciones entre ellos, indicando cómo reaccionar al recibir un input.
En pocas palabras, un autómata es una máquina de estados con una serie de instrucciones cuya función es procesar input y devolver una respuesta.
Se utilizan **grafos** para su implementación. Los nodos del grafo serán los estados del autómata y los arcos las transiciones entre estados.
Son importantes porque nos ayudan a comprender y resolver problemas de muchas áreas, pues existen distintos tipos:

- **Finito**: Los más sencillos, con número finito de estados y con procesamiento de entradas secuencial. Son usados para procesar lenguajes naturales y para reconocer patrones en textos y buscadores.
- **De Pila**: Procesan información usando una pila, lo que los hace más poderosos para el análisis sintáctico de lenguajes de programación y en la validación de expresiones matemáticas.
- **Máquina de Turing**: Tienen una cinta infinita y pueden resolver cualquier problema hecho con computación.

Jerarquía de Chomsky

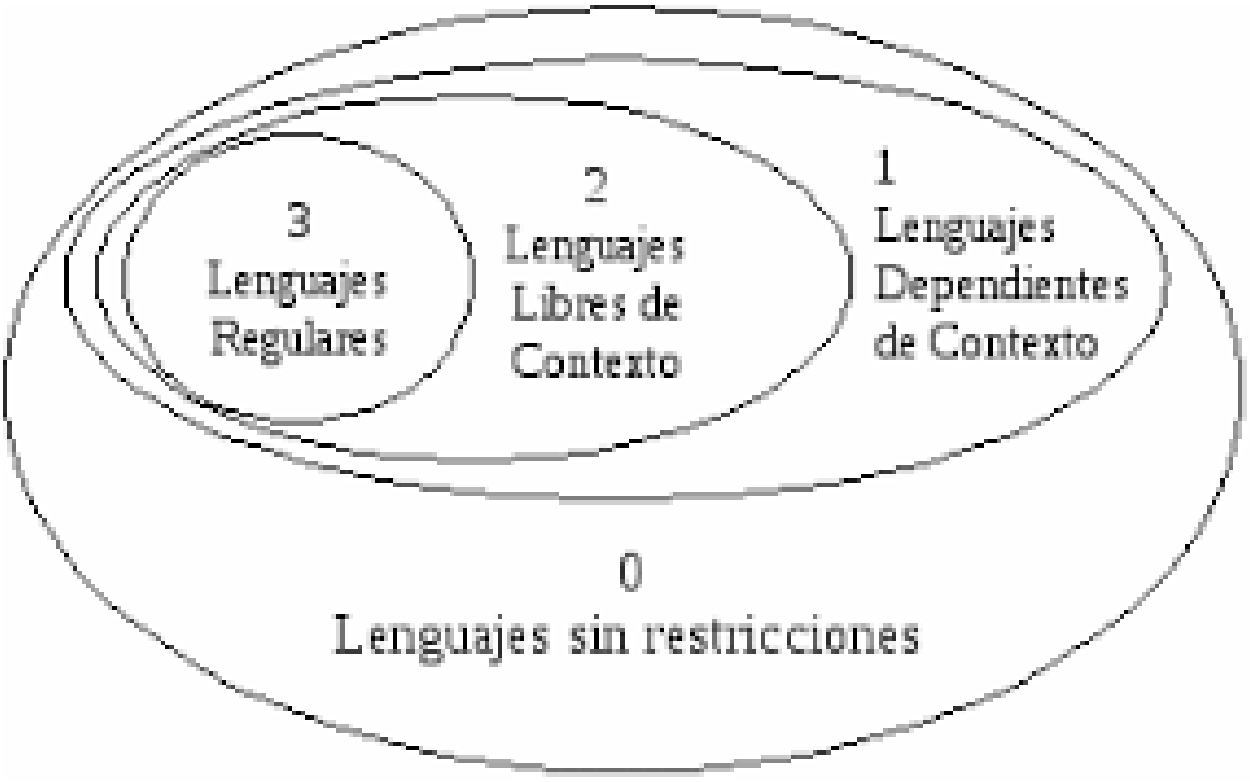


Figure 1: Modelo Relacional propuesto.

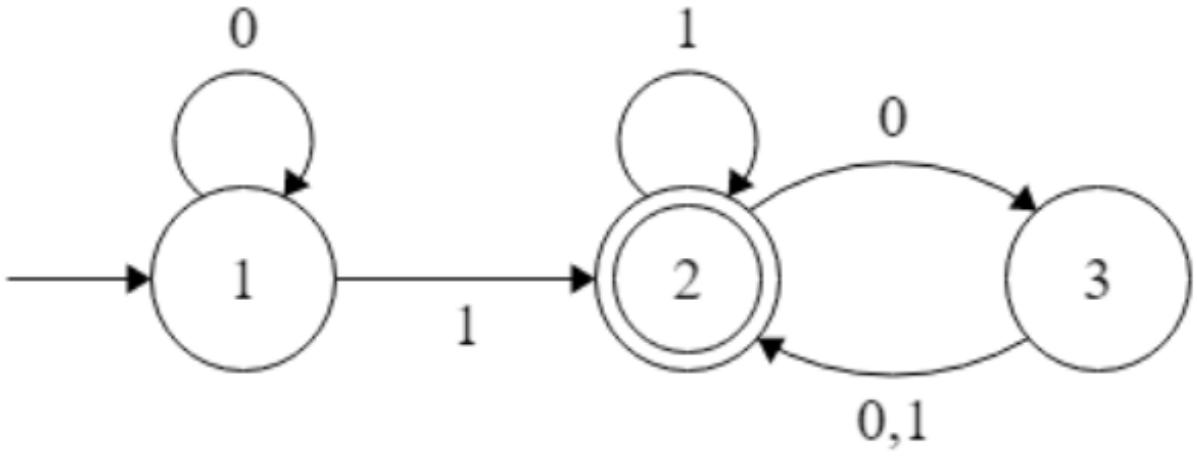
Autómata Finito Determinista (AFD)

En estos autómatas los inputs son cadenas o palabras. Cuando el AFD los recibe, los procesa secuencialmente y decide si la cadena es aceptada o no por el autómata.
Un AFD es una máquina abstracta formada por cinco elementos:

$$A = (Q, \Sigma, \Delta, q_0, F)$$

- Q: Conjunto de estados del autómata.
- Σ: Alfabeto, signos con los que puede trabajar.
- Δ: Función parcial de transición, recoge todas las posibles transiciones entre estados del autómata. En cada estado no tiene por qué haber una transición para cada símbolo, pero si esto ocurre, se dice que es un autómata completo.
- q_0: Estado inicial del autómata, representado por una flecha pequeña, por el cual comienza el procesamiento de cualquier cadena.
- F: Subconjunto de Q, que recoge todos los estados finales o aquellos estados en los que, si al terminar el procesamiento nos encontramos en ellos, la cadena será aceptada.

Es determinista porque para cada estado, existe una transición por símbolo como máximo.



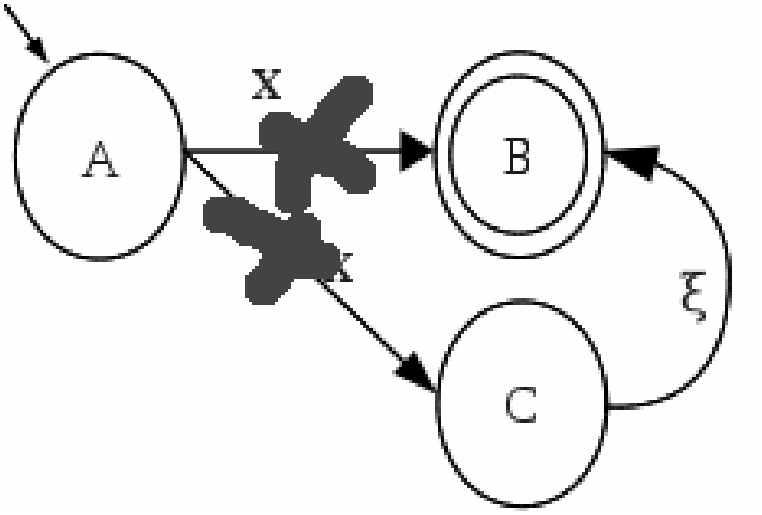
Grafo AFD

Autómata Finito No Determinista (AFND)

Estos autómatas es una tupla de 5 elementos con la forma:

$$A = (Q, \Sigma, \Delta, q_0, F)$$

Muy parecido a un AFD, a excepción de Δ pues al no ser determinista permite varias transiciones definidas para cada símbolo desde un estado, es decir, varios caminos a tomar al recibir un input, tal como se observa en el grafo de ejemplo:



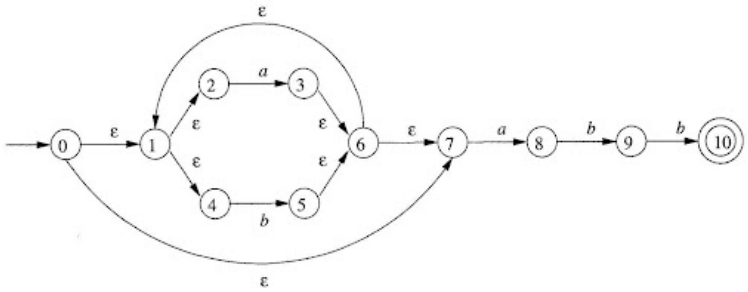
Grafo AFND

Podemos transformar un AFND en un AFD agrupando sus estados en subconjuntos y ajustando sus funciones de transición

Autómata con Transiciones Epsilon (AFλ)

Lambda (λ) representa la cadena vacía, es una palabra que simboliza la ausencia de símbolos. Además, tiene la característica de que es posible aplicarle cualquier operación con cadenas de caracteres, como concatenación, potencia, etc.
Por lo tanto, un **Autómata Finito con Transiciones Vacías** es capaz de cambiar de estado sin procesar ningún símbolo de entrada, pues admite en su definición el uso de los valores lambda:

$$\Delta = Q \times (\Sigma \cup \{\lambda\}) \to 2^Q \tag{1}$$



Grafo AF Lambda

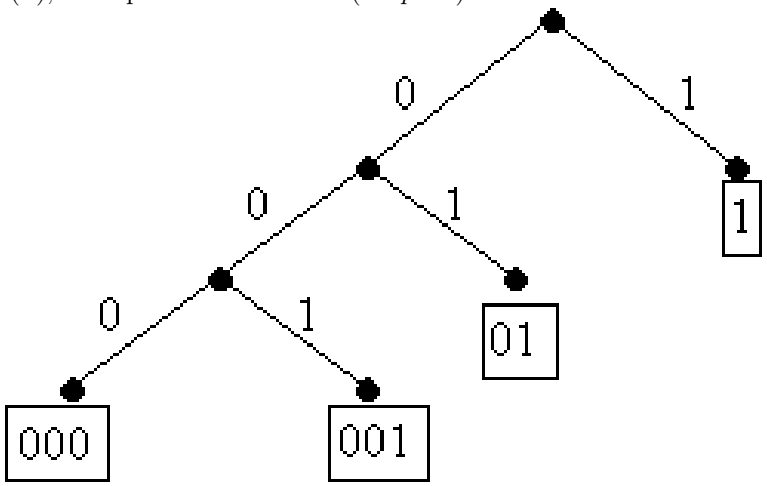
Pattern Matching

Es una técnica muy importante dentro del mundo de la computación, pues se centra en encontrar coincidencias de un patrón específico entre una estructura de datos más grande como una secuencia de texto o una base de datos. También se pueden trabajar patrones más complejos como expresiones regulares para hacer búsquedas más flexibles y complejas, usado en Sistemas Operativos o incluso en el campo de la Biología para la identificación de secuencias genéticas.

Naive Algorithm

Construir un Árbol Aceptor de Prefijos, el cual es un autómata que contiene los prefijos de un conjunto C, este tiene un bucle en el conjunto vacío para reiniciar la ejecución por cada carácter y que estos se procesen individualmente, así encontrar constantemente el número de palabras que coincidan.

No es muy eficiente, en el mejor de los casos, si el patrón aparece al inicio del texto, su complejidad será lineal $O(n)$, en el peor de los casos $O(n * p * x)$



Árbol Aceptor de Prefijos

Autómata Diccionario

A partir de un conjunto C de palabras y un alfabeto σ, es posible definir un Autómata Diccionario, se diferencia de los demás autómatas en que sus estados finales F estarán formados por los estados identificados con una palabra que tenga al menos un sufijo C.

También cambia δ pues ahora tendremos a $h(u)$ que indica al sufijo más largo de u que pertenece a C, esto quiere decir que la función de transición no es parcial, si no que es completo, pues para cada estado existirá una transición definida para cada símbolo del alfabeto, esto para no detenerse en un estado, con esto identificaremos que patrones contienen a otros y podremos volver a direccionar cada estado, este algoritmo siempre tendrá una complejidad lineal $O(n)$.

Lenguaje por la Derecha

$$Rq = x \in \Sigma^* : \delta(q, x) \in F \tag{2}$$

Dado un Autómata Finito Determinista el *Lenguaje por la derecha* de un estado q es el conjunto de palabras en la clausura de Klenn(Σ^*) tal que la función de transiciones para dichas palabras desde el estado q nos lleva a un estado final, en otras palabras es el conjunto de cadenas que al ser procesadas desde un estado, nos hacen alcanzar un estado final.
El lenguaje por la derecha es útil para definir el comportamiento de los estados del autómata, si entre dos estados (p y q) tienene el mismo lenguaje por la derecha, significará que se comportan igual, en el caso contrario, si todos los estados del autómata tienen lenguajes por la derecha distintos, entonces diremos que el autómata es **reducido**.

Clases de Equivalencia

Una **relación** R sobre A es de equivalencia si cumple con las siguientes 3 caracteristicas:

1. **Reflexiva:** Cada estado tiene el mismo comportamiento que si mismo $\forall x \in A(xRx)$
2. **Simétrica:** Si el estado x tiene el mismo comportamiento que y entonces y igual tiene el mismo comportamiento que x $\forall x \in A(xRy => yRx)$
3. **Transitiva:**Si el estado x tiene el mismo comportamiento que y, e y tiene el mismo comportamiento que z, entonces x tiene el mismo comportamiento que z $\forall x, y, zA(xRy \wedge yRz => xRz)$

Por lo tanto **clases de equivalencia** son un concepto fundamental de los autómatas, en estas dos estados se consideran equivalentes si y solo si no se pueden distinguir mediante ninguna cadena de entrada.

Definición Formal

Dados dos estados q_i y q_j de un autómata finito, se dice que son equivalentes si para toda cadena $w \in \Sigma^*$ se cumple que:

$$\delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \in F \tag{3}$$

Es decir, ambos estados llevan a la misma aceptación o rechazo para cualquier entrada.

Conjunto Cociente

Es el conjunto cuyos elementos son las diferentes clases de equivalencia de R y se representa por A/R

Lenguaje Regular

Un **lenguaje regular** es aquel que puede ser reconocido por un autómata finito determinista (AFD) o un autómata finito no determinista (AFND). También puede ser descrito mediante una expresión regular.

Definición Formal

Un lenguaje L es regular si existe un autómata finito $A = (Q, \Sigma, \delta, q_0, F)$ tal que:

$$L = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\} \tag{4}$$

Ejemplo

El lenguaje $L = \{a^n b^m \mid n, m \geq 0\}$ es regular porque se puede describir con la expresión regular:

$$L = a^* b^* \tag{5}$$

Lenguaje No Regular

Un **lenguaje no regular** es aquel que no puede ser reconocido por un autómata finito. Se puede demostrar que un lenguaje no es regular utilizando el *lema del bombeo*.

Lema del Bombeo

Si un lenguaje L es regular, entonces existe una constante $p > 0$ (llamada *longitud de bombeo*) tal que cualquier cadena $w \in L$ con $|w| \geq p$ se puede dividir en tres partes:

$$w = xyz \tag{6}$$

con las siguientes condiciones:

- $|xy| \leq p$
- $|y| > 0$
- Para todo $k \geq 0$, la cadena $xy^k z \in L$

Ejemplo de Lenguaje No Regular

El lenguaje $L = \{a^n b^n \mid n \geq 0\}$ no es regular porque no puede ser reconocido por un autómata finito y falla en el lema del bombeo.

Minimización de Estados de un Autómata

La minimización de un autómata busca reducir la cantidad de estados manteniendo el mismo lenguaje reconocido. Esto se logra mediante el algoritmo de minimización de estados.

Algoritmo de Minimización

1. Eliminar estados inaccesibles.
2. Construir una tabla de equivalencias para distinguir estados no equivalentes.
3. Agrupar los estados equivalentes y construir un nuevo autómata.

Ejemplo de Minimización

Consideremos el siguiente autómata con estados $Q = \{q_0, q_1, q_2, q_3, q_4\}$. Después del proceso de minimización, podemos reducirlo a un autómata con menos estados, manteniendo la misma capacidad de reconocimiento de cadenas.

Referencias Bibliográficas

References

[1] Pérez, E. M., Acevedo, J. M., & Silva, C. F. (2009). Autómatas programables y sistemas de automatización. Marcombo.

[2] Glushkov, V. M. (1961). The abstract theory of automata. Russian Mathematical Surveys, 16(5), 1.

[3] Perrin, D. (1990). Finite automata. In Formal Models and Semantics (pp. 1-57). Elsevier.

[4] Codemath. (2024). Autómatas y Lenguajes Formales DESDE CERO. YouTube. https://www.youtube.com/playlist?list=PLyRNgg3I27WiOZqDamrZon3QDPZMlZ5P4

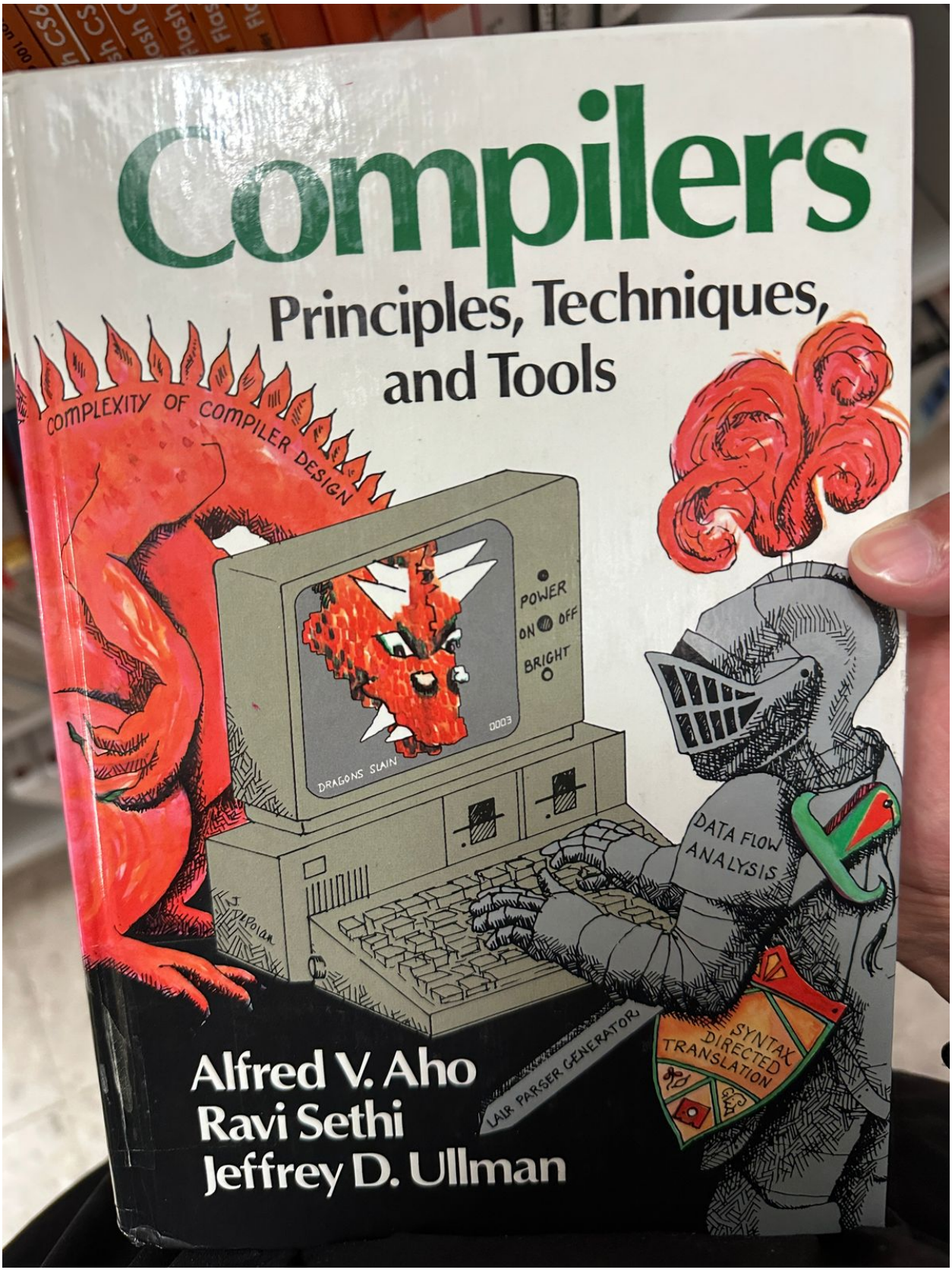


Figure 2: Libro1.

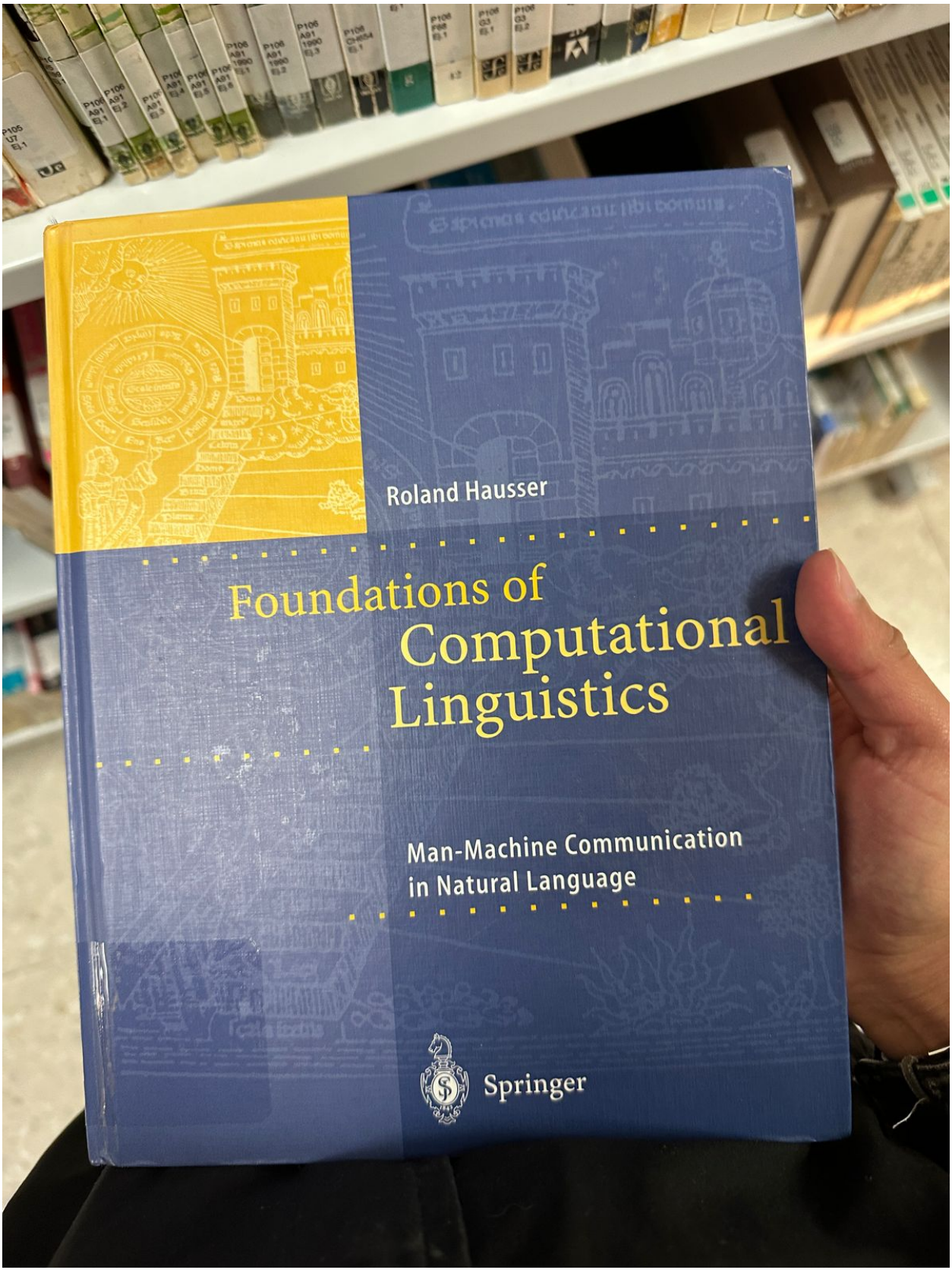


Figure 3: Libro1.

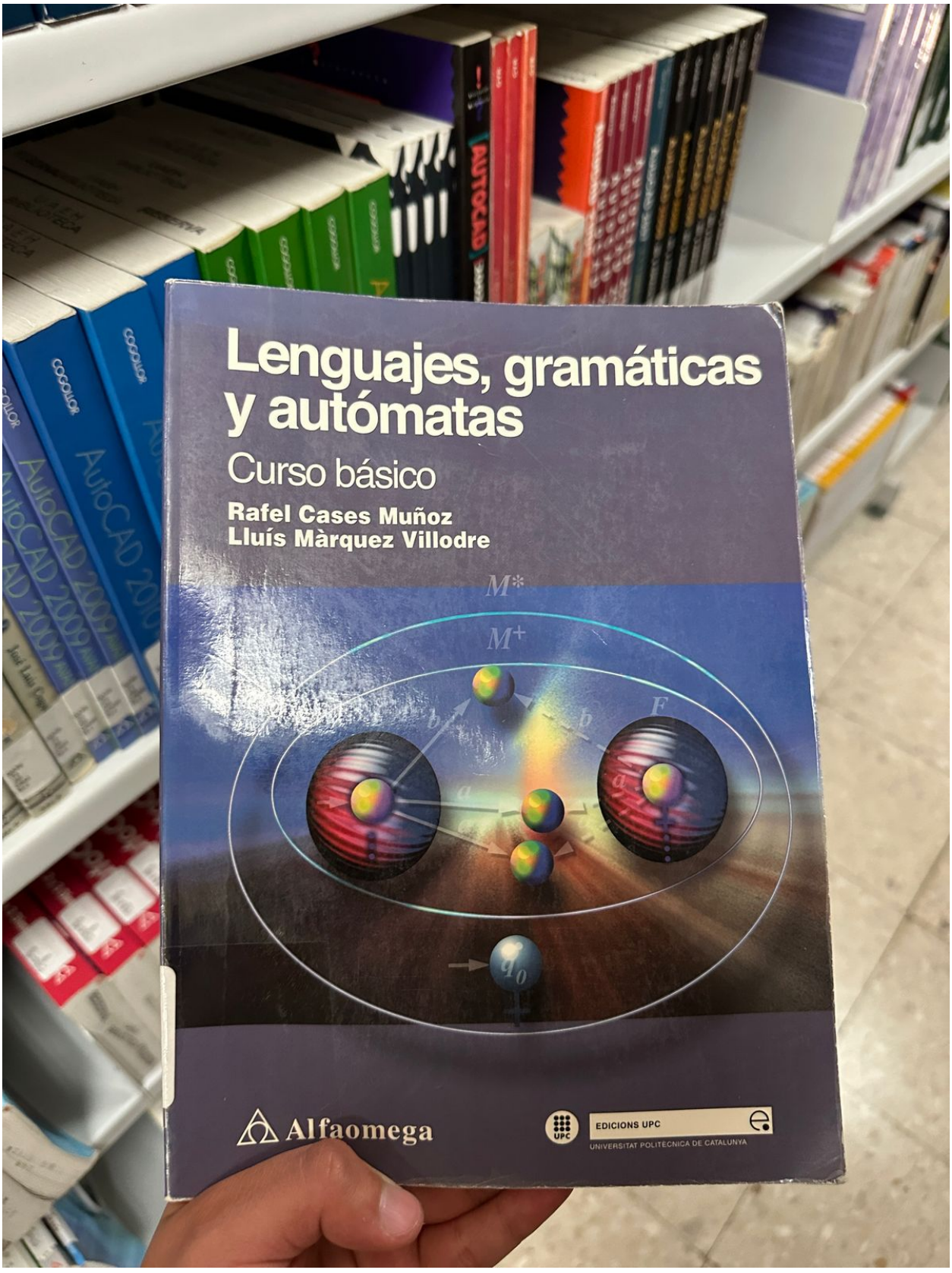


Figure 4: Libro1.