**Temas de Investigación**

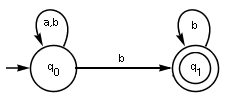
**1. Teoría de autómatas**

* **Definición y usos de los autómatas finitos deterministas (AFD)**

Un **autómata finito determinista** (abreviado **AFD**) es un **autómata finito** que además es un sistema **determinista**; es decir, para cada estado en que se encuentre el **autómata**, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

* **Ejemplos y aplicaciones** en informática y ciencias de la computación

La **informática**,[1](https://es.wikipedia.org/wiki/Inform%C3%A1tica#cite_note-1)​ también llamada **computación**,[2](https://es.wikipedia.org/wiki/Inform%C3%A1tica#cite_note-2)​ es el área de la [ciencia](https://es.wikipedia.org/wiki/Ciencia) que se encarga de estudiar la administración de métodos, técnicas y procesos con el fin de almacenar, procesar y transmitir [información](https://es.wikipedia.org/wiki/Informaci%C3%B3n) y [datos](https://es.wikipedia.org/wiki/Dato) en formato digital

Por ejemplo, **los termostatos, los pilotos automáticos de aeronaves, los sistemas de guía de misiles, las redes telefónicas y los controles de ciertos tipos de ascensores automáticos** son todos tipos de autómatas.

**2. Gramáticas formales**

* **Gramáticas regulares**: concepto y ejemplos de su uso en lenguajes de programación.
* Una gramática ("G") desde el punto de vista de la teoría de autómatas es un **conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un lenguaje específico** L. Dos gramáticas que describan el mismo lenguaje se llaman gramáticas equivalente



* **Gramáticas recursivas**: explicación y diferencias con las gramáticas regulares

Una gramática ("G") desde el punto de vista de la teoría de autómatas es un **conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un lenguaje específico**

Las gramáticas se utilizan para describir lenguajes. Existe una primera clasificación para las gramáticas, y esta estará en función a los tipos de lenguajes que genere, esto es ya sean lenguajes naturales o lenguajes formales.

Un Lenguaje Formal, por el contrario surge a partir de su gramática, y por lo tanto no presenta excepciones en su definición. Esto es así, porque  los Lenguajes Formales son los que se utilizan para que se comuniquen los hombres con las máquinas. De esta manera a partir de las gramáticas formales es como surgen los Lenguajes de Programación

**3. Análisis léxico**

* **Tokenización**: qué es y cómo se usa en el proceso de compilación.

La tokenización **divide el texto en partes más pequeñas para facilitar el análisis mecánico, ayudando a las máquinas a comprender el lenguaje humano**.

La tokenización es el **proceso de sustituir los datos sensibles por símbolos de identificación únicos que conservan toda la información esencial de los datos sin comprometer su seguridad**.

* **Expresiones regulares**: su función en el análisis léxico.

Se corresponden con las gramáticas de tipo 3 de la jerarquía de Chomsky. Las expresiones

regulares son una forma de especificar patrones, entendiendo por patrón la forma de describir

cadenas de caracteres. Es la forma de definir los tokens o componentes léxicos y, como

veremos, cada patrón concuerda con una serie de cadenas

* **Reconocimiento de palabras clave**: cómo se identifican en los lenguajes de programación.
* **Identificación de tokens**: clasificación y ejemplos prácticos.

**4. Análisis sintáctico**

* **Tipos de análisis sintáctico**:
  + Análisis sintáctico descendente.
  + Análisis sintáctico ascendente.
* **Árboles de análisis sintáctico**: qué son y cómo ayudan en la estructura de los programas.

**5. Algoritmos de análisis de cadenas**

* **Algoritmo de Boyer-Moore**: concepto, funcionamiento y su aplicación en la búsqueda de cadenas dentro de textos o programas.

**6. Compiladores e intérpretes**

* **Estructura de un compilador**: definición y componentes.
* **Fases del compilador**:
  + Análisis léxico.
  + Análisis sintáctico.
  + Generación de código.
  + Optimización de código.
* **Lenguajes de programación utilizados en la creación de compiladores**: como **Lex y Yacc**.

**7. Aplicaciones prácticas de autómatas**

* **Reconocimiento de patrones** en diferentes áreas.
* **Procesamiento de lenguaje natural** y su uso en inteligencia artificial.
* **Análisis de texto y minería de datos**: su importancia en el análisis de información.

**8. Lenguaje de programación C y C++**

* **Librerías fundamentales**: **STDIO y CONIO,** un resumen de las funciones de cada una.
* **Lista de 10 librerías más comunes en C/C++**, además de las ya mencionadas.
* **Manejadores de formatos para diferentes tipos de datos**, con ejemplos de uso en código.