**Temas de Investigación**

1. **Teoría de autómatas**

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

La teoría de autómatas es **el estudio de las máquinas abstractas y los autómatas , así como de los problemas computacionales que se pueden resolver mediante su uso**. Es una teoría de la informática teórica con estrechas conexiones con la lógica matemática

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



Son conjunto de señales recibidas por el procesador de un ordenador, el que está compuesto por un número **finito** de compuertas(estados), cada una de las cuales pueden utilizar dos condiciones posibles, por lo general denotadas por 0 y 1(transiciones).

**2. Gramáticas formales**

* **Gramáticas regulares**: concepto y ejemplos de su uso en lenguajes de programación.

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) una **gramática regular** es una [gramática formal](https://es.wikipedia.org/wiki/Gram%C3%A1tica_formal) (V, Σ, R, S) que puede ser clasificada como regular izquierda o regular derecha. Las gramáticas regulares sólo pueden generar a los [lenguajes regulares](https://es.wikipedia.org/wiki/Lenguaje_regular) de manera similar a los autómatas finitos y las expresiones regulares.

Dos gramáticas regulares que generan el mismo lenguaje regular se denominan equivalentes. Toda gramática regular es una [gramática libre de contexto](https://es.wikipedia.org/wiki/Gram%C3%A1tica_libre_de_contexto).

Una **gramática regular derecha** es aquella cuyas reglas de producción P son de la siguiente forma:

1. *A* → *a*, donde *A* es un símbolo no-terminal en *N* y *a* uno terminal en Σ
2. *A* → *aB*, donde *A* y *B* pertenecen a *N* y *a* pertenece a Σ
3. *A* → ε, donde *A* pertenece a *N*.

Análogamente, en una **gramática regular izquierda**, las reglas son de la siguiente forma:

1. *A* → *a*, donde *A* es un símbolo no-terminal en *N* y *a* uno terminal en Σ
2. *A* → *Ba*, donde *A* y *B* pertenecen a *N* y *a* pertenece a Σ
3. *A* → ε, donde *A* pertenece a *N*.

Una definición equivalente evita la regla 1 (*A* → *a*) ya que es sustituible por:

*A* → *aL*

*L* → ε

en el caso de las gramáticas regulares derechas y por:

*A* → *La*

*L* → ε

en el caso de las izquierdas.

Algunos autores alternativamente no permiten el uso de la regla 3 suponiendo que la cadena vacía no pertenece al lenguaje.

Un ejemplo de una gramática regular G con N = {S, A}, Σ = {a, b, c}, P se define mediante las siguientes reglas:

S → aS

S → bA

A → ε

A → cA

donde S es el símbolo inicial. Esta gramática describe el mismo lenguaje expresado mediante la [expresión regular](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular) a\*bc\*.

Dada una gramática regular izquierda es posible convertirla, mediante un algoritmo en una derecha y viceversa.

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

**3. Análisis léxico**

* **Tokenización**: qué es y cómo se usa en el proceso de compilación.
* **Expresiones regulares**: su función en el análisis léxico.
* **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.