■ Ruta de Aprendizaje de Teoría de Autómatas y Cómputo Formal (6 meses)

Módulo 1 (Semanas 1–4): Fundamentos matemáticos

Semana 1: Lógica proposicional

- Tablas de verdad y su construcción.
- Conectivos lógicos ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$).
- Equivalencias lógicas y leyes de De Morgan.
- Ejercicio: implementar en Rust un verificador de tautologías.

Semana 2: Lógica de predicados

- Cuantificadores ($\forall$, $\exists$) y negación de cuantificadores.
- Traducción entre enunciados naturales $\leftrightarrow$ fórmulas lógicas.
- Alcance de variables en fórmulas.
- Ejercicio: construir un verificador de validez para fórmulas simples.

Semana 3: Conjuntos y funciones

- Operaciones con conjuntos (unión, intersección, diferencia).
- Producto cartesiano y relaciones binarias.
- Funciones inyectivas, suprayectivas y biyectivas.
- Ejercicio: programa en Rust que verifique estas propiedades en conjuntos finitos.

Semana 4: Inducción y recurrencias

- Principio de inducción matemática.
- Prueba de propiedades en sucesiones numéricas.
- Recurrencias básicas (Fibonacci, factorial).
- Ejercicio: comprobación automática de inducción para casos simples.

Módulo 2 (Semanas 5–8): Lenguajes y expresiones regulares

Semana 5: Lenguajes formales

- Definición: alfabeto, palabra, lenguaje como conjunto.
- Operaciones con cadenas.
- Ejercicio: generar todas las cadenas de $\Sigma=\{a,b\}$ de longitud ≤ 4 en Rust.

Semana 6: Operaciones con lenguajes

- Unión, concatenación, clausura de Kleene, complemento.
- Lenguajes finitos vs infinitos.
- Ejercicio: implementar estas operaciones en Rust con conjuntos.

Semana 7: Expresiones regulares

- Definición formal de expresiones regulares.
- Equivalencia regex $\leftrightarrow$ lenguajes regulares.
- Construcción de NFA a partir de regex (algoritmo de Thompson).
- Ejercicio: implementa generador de NFA desde regex simples.

Semana 8: Limitaciones de regex

- Lenguajes no regulares (ej. {a^n b^n | n ≥ 0}).
- Uso del lema de bombeo.
- Ejercicio: demostrar con el lema que {a^n b^n} no es regular.

Módulo 3 (Semanas 9–12): Autómatas finitos

Semana 9: DFA

- Definición formal $(Q, \Sigma, \delta, q0, F)$.
- Diagramas de estados.
- Ejercicio: DFA que valide cadenas terminadas en 01.

Semana 10: NFA

- Definición formal de NFA.
- Equivalencia con DFA.
- Ejercicio: programa un simulador de NFA en Rust.

Semana 11: Conversión NFA$\rightarrow$DFA

- Algoritmo de construcción de subconjuntos.
- Eliminación de ε-transiciones.

- Ejercicio: implementa el algoritmo en Rust.

Semana 12: Minimización de DFA

- Algoritmo de particiones de estados.
- Construcción de DFA mínimo equivalente.
- Ejercicio: implementa minimizador de DFA.

Módulo 4 (Semanas 13–16): Autómatas con pila y gramáticas

Semana 13: PDA

- Definición formal de autómata con pila.
- Ejemplo: balanceo de paréntesis.
- Ejercicio: implementar simulador de PDA.

Semana 14: CFG

- Gramáticas libres de contexto.
- Notación BNF y EBNF.
- Ejercicio: definir gramática para expresiones aritméticas.

Semana 15: Árboles de derivación

- Construcción de árboles de derivación.
- Ejercicio: generar árbol automáticamente en Rust.

Semana 16: Ambigüedad gramatical

- Gramáticas ambiguas.
- Caso clásico del if-else.
- Ejercicio: detectar ambigüedad en gramática.

Módulo 5 (Semanas 17–20): Computabilidad y máquinas de Turing

Semana 17: Introducción TM

- Definición formal de TM.

- Ejemplo: suma en unario.
- Ejercicio: implementar TM básica en Rust.

Semana 18: Variantes de TM

- Multi-cinta y no determinista.
- Equivalencia de potencias de cómputo.
- Ejercicio: comparar TM estándar vs multi-cinta.

Semana 19: Lenguajes recursivos y RE

- Lenguajes decidibles e indecidibles.
- Ejercicio: clasificar ejemplos de lenguajes.

Semana 20: Problema de la parada

- Demostración de indecidibilidad.
- Ejercicio: simular TM y mostrar no terminación.

Módulo 6 (Semanas 21–24): Complejidad y aplicaciones

Semana 21: Complejidad computacional

- Clases P y NP.
- Ejemplo: SAT.
- Ejercicio: resolver SAT pequeño por fuerza bruta.

Semana 22: Reducciones

- Concepto de reducción entre problemas.
- Ejemplo: 3-SAT $\rightarrow$ CLIQUE.
- Ejercicio: demostrar reducción con ejemplos.

Semana 23: Lexer práctico

- Construcción de lexer basado en DFA.
- Reconocimiento de tokens.
- Ejercicio: lexer para lenguaje mini en Rust.

Semana 24: Parser JSON

- Construcción de parser con PDA o recursivo.
- Ejercicio: validar { "key": [1,2,3] }.

■ Recomendaciones finales

- Lectura paralela: 1 capítulo de *Sipser* por semana.
- Práctica: todo concepto debe tener implementación en Rust.
- Refuerzo: usar diagramas de estados y árboles en papel antes de programar.

■ Al final de los 6 meses habrás alcanzado:

- Dominar DFA, NFA, PDA, CFG y TM.
- Implementar simuladores en Rust.
- Construir un lexer y parser propio para JSON (base de compiladores).