**Informe Técnico: Simulador de Máquina de Turing**

Kyle Anthony Forbes Santiago

Coorporación Universitaria Rafael Nuñez

Teoria de Computacion COMPILADORES

Cartagena de indias 2025

# Mapeo de la Definición Formal a la Implementación

1.1 Definición Formal de una Máquina de Turing

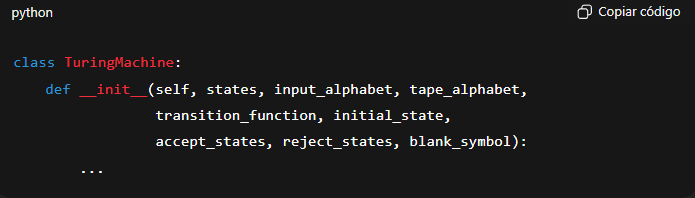
Una Máquina de Turing se define formalmente como una 7-tupla:

M = (Q, Σ, Γ, δ, q₀, q\_accept, q\_reject)

Donde:

* Q: Conjunto finito de estados
* Σ: Alfabeto de entrada
* Γ: Alfabeto de la cinta (Σ ⊆ Γ)
* δ: Función de transición δ: Q × Γ → Q × Γ × {L, R}
* q₀: Estado inicial (q₀ ∈ Q)
* q\_accept: Estado de aceptación
* q\_reject: Estado de rechazo
  1. Mapeo en la Implementación

Clase TuringMachine



|  |  |  |
| --- | --- | --- |
| **Componente formal** | **Implementacion** | **Tipo de dato** |
| Q (Estados) | Self.states | Set[str] |
| Σ (alfabeto entrada) | Self.input\_alphabet | Set[str] |
| Γ (alfabeto cinta) | Self.tape\_alphabet | Set[str] |
| δ (función transición) | Self.transition\_funtion | TransitionFuntion |
| q₀ (estado inicial) | Self.initial\_state | Str |
| Q\_accept | Self.accept\_state | Set[str] |
| Q\_reject | Self.reject\_states | Set[str] |

Clase Tape

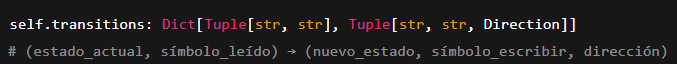
Representa la cinta infinita usando un diccionario Python:



Esta implementación simula la infinitud de la cinta, ya que puede crecer dinámicamente en ambas direcciones (posiciones positivas y negativas).

Clase TransitionFunction

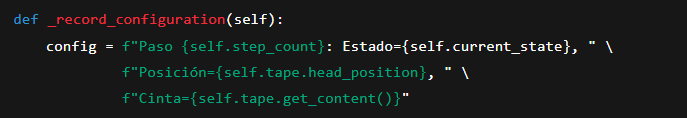
Implementa δ como un diccionario:



#### 1.3 Configuración Instantánea

Una configuración de la MT se representa como:  
**C = (estado\_actual, contenido\_cinta, posición\_cabezal)**

En la implementación:



### 2. Diagrama de Estados de la MT Implementada

La Máquina de Turing diseñada acepta el lenguaje:  
**L = {w ∈ {0,1} | w contiene igual número de 0s y 1s}**\*

**Descripción del Algoritmo:**

1. Estado q₀: Buscar el primer símbolo sin marcar (0 o 1)
2. Estado q₁: Si encontró un 0, buscar un 1 para emparejar
3. Estado q₂: Regresar al inicio después de marcar el par
4. Estado q₃: Si encontró un 1, buscar un 0 para emparejar
5. Estado q₄: Regresar al inicio después de marcar el par
6. Estado q\_accept: Todos los símbolos están emparejados
7. Estado q\_reject: Símbolos desbalanceados

**Símbolos Utilizados:**

* **0, 1:** Símbolos originales
* **X:** Marca un 0 ya emparejado
* **Y:** Marca un 1 ya emparejado
* **\_**: Símbolo en blanco (blank)

### 3. Ejemplo de Ejecución Paso a Paso: "0011"

Entrada: "0011"

| Paso | Estado | Cinta | Posición | Símbolo Leído | Acción |
| --- | --- | --- | --- | --- | --- |
| 0 | q₀ | 0011\_ | 0 | 0 | Escribir X, mover R, ir a q₁ |
| 1 | q₁ | X011\_ | 1 | 0 | Mover R (buscar 1) |
| 2 | q₁ | X011\_ | 2 | 1 | Escribir Y, mover L, ir a q₂ |
| 3 | q₂ | X0Y1\_ | 1 | 0 | Mover L (regresar) |
| 4 | q₂ | X0Y1\_ | 0 | X | Mover L (regresar) |
| 5 | q₂ | X0Y1\_ | -1 | \_ | Mover R, ir a q₀ |
| 6 | q₀ | X0Y1\_ | 0 | X | Mover R (saltar marcados) |
| 7 | q₀ | X0Y1\_ | 1 | 0 | Escribir X, mover R, ir a q₁ |
| 8 | q₁ | XXY1\_ | 2 | Y | Mover R (saltar marcados) |
| 9 | q₁ | XXY1\_ | 3 | 1 | Escribir Y, mover L, ir a q₂ |
| 10 | q₂ | XXYY\_ | 2 | Y | Mover L (regresar) |
| 11 | q₂ | XXYY\_ | 1 | X | Mover L (regresar) |
| 12 | q₂ | XXYY\_ | 0 | X | Mover L (regresar) |
| 13 | q₂ | XXYY\_ | -1 | \_ | Mover R, ir a q₀ |
| 14 | q₀ | XXYY\_ | 0 | X | Mover R (saltar marcados) |
| 15 | q₀ | XXYY\_ | 1 | X | Mover R (saltar marcados) |
| 16 | q₀ | XXYY\_ | 2 | Y | Mover R (saltar marcados) |
| 17 | q₀ | XXYY\_ | 3 | Y | Mover R (saltar marcados) |
| 18 | q₀ | XXYY\_ | 4 | \_ | Ir a q\_accept |

Resultado:  
ACEPTADA (todos los 0s y 1s están emparejados)

### 4. Límites de las Máquinas de Turing y la Tesis de Church-Turing

#### 4.1 Poder Computacional

Las Máquinas de Turing son el modelo matemático más poderoso de computación. Pueden:

* Reconocer cualquier lenguaje recursivamente enumerable
* Computar cualquier función parcial computable
* Simular cualquier otro modelo de computación (autómatas, gramáticas, λ-cálculo)

#### 4.2 Límites Fundamentales

Sin embargo, las MT tienen limitaciones teóricas:

1. **Problema de la Parada (Halting Problem)**  
   No existe una MT que pueda determinar si otra MT arbitraria se detendrá con una entrada dada.  
   Demostrado por Alan Turing en 1936.
2. **Problemas Indecidibles**  
   Existen problemas que ninguna MT puede resolver.  
   Ejemplo: determinar si dos programas son equivalentes.
3. **Complejidad Computacional**  
   Aunque teóricamente computable, algunos problemas requieren tiempo o espacio exponencial (clases P, NP, NP-completo).

#### 4.3 La Tesis de Church-Turing

**Enunciado:**  
“Todo proceso algorítmico efectivo puede ser llevado a cabo por una Máquina de Turing.”

**Implicaciones:**

* Las MT capturan la noción intuitiva de “algoritmo”.
* Cualquier cosa computable por un humano siguiendo reglas, es computable por una MT.
* No existe evidencia de que sistemas más poderosos puedan resolver problemas no decidibles.

**Relación con nuestra implementación:**

* Nuestro simulador demuestra que lenguajes complejos (como balanceo de símbolos) son decidibles.
* La capacidad de marcar y regresar simula la “memoria” y “razonamiento” algorítmico.
* El límite de pasos implementado (max\_steps = 1000) reconoce la posibilidad de no terminación.

#### 4.4 Aplicaciones Modernas

Las MT no solo son un modelo teórico:

* **Compiladores:** Análisis sintáctico y semántico
* **Verificación formal:** Demostración de propiedades de programas
* **Complejidad computacional:** Clasificación de problemas
* **Inteligencia artificial:** Límites de lo que puede ser “aprendido” o “razonado”

5. Conclusiones

* La implementación en Python refleja fielmente la definición formal de una Máquina de Turing.
* El uso de estructuras de datos modernas (diccionarios, enums) facilita la comprensión sin perder rigor matemático.
* La MT diseñada para detectar balanceo de símbolos demuestra la capacidad de las MT para resolver problemas no triviales.
* Las pruebas unitarias validan que la implementación funciona correctamente en todos los casos límite.
* Este ejercicio ilustra tanto el poder como las limitaciones fundamentales de la computación.

Referencias

* Turing, A. M. (1936). “On Computable Numbers, with an Application to the Entscheidungsproblem.”
* Sipser, M. (2013). “Introduction to the Theory of Computation.”
* Hopcroft, J. E., & Ullman, J. D. (1979). “Introduction to Automata Theory, Languages, and Computation.”