Máquinas de Turing

Juan Mendivelso

Universidad Nacional de Colombia
Facultad de Ciencias
Departamento de Matemáticas

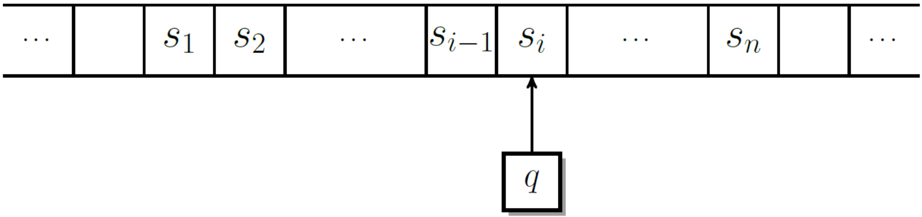
Presentación basada en las Notas de Clase del Profesor Rodrigo De Castro.

Outline

Outline

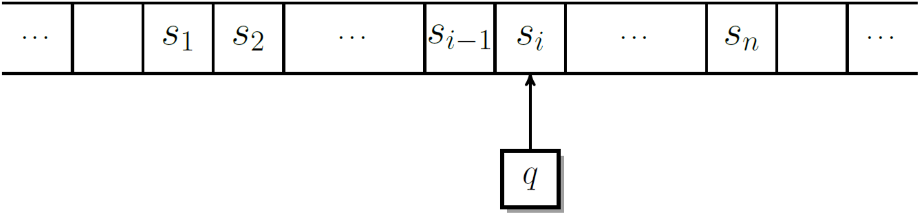
Máquinas de Turing

- Tiene una cinta infinita por los dos lados.
- Las celdas en blanco se representan como $\square$.
- Tiene un alfabeto de entrada Σ y un alfabeto de cinta Γ.
- $\square$ no pertenece a Γ.
- Se define $\Gamma_\square = \Gamma \cup \{\square\}$.



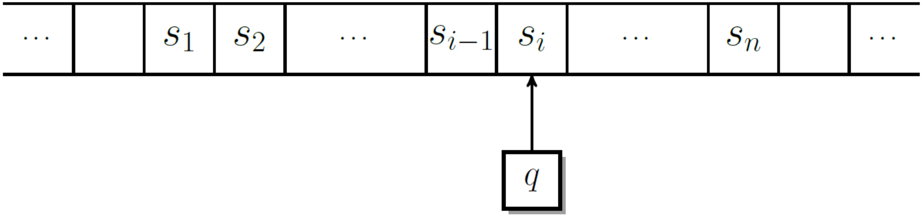
Máquinas de Turing

- Tiene un conjunto finito de estados Q.
- Hay estado inicial q_0 y estados de aceptación $F \neq \emptyset$.
- La unidad de control empieza en el estado q_0 y apuntando al primer caracter de una cadena escrita en la cinta.
- No es necesario procesar toda la cadena. Si se llega a un estado de aceptación, la cadena se acepta.



Máquinas de Turing

- Cuando la unidad está en determinado estado y determinado símbolo, este puede ser sobre-escrito o no. Luego, la unidad puede quedarse quieta, o puede desplazarse a la izquierda o a la derecha.
- Este modelo es determinista.
- No se hacen transiciones desde estados de aceptación.
- Como la unidad se puede quedar quieta, no es necesario definir transiciones λ.



Máquinas de Turing

Una **Máquina de Turing (MT)** $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ consta de seis componentes:

1. Q es el conjunto (finito) de estados internos de la unidad de control.
2. $q_0 \in Q$ es el estado inicial.
3. $\emptyset \neq F \subseteq Q$ es el conjunto de estados finales o estados de aceptación.
4. Σ es el alfabeto de entrada.
5. Γ es el alfabeto de cinta. $\Sigma \subseteq \Gamma$. $\Gamma - \Sigma$ son los símbolos auxiliares.
6. δ es la función de transición de la máquina:

$$\delta : Q \times \Gamma_\square \to Q \times \Gamma_\square \times \{\leftarrow, \to, -\},$$

donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.

Función de Transición

Dada la Máquina de Turing $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco, δ está definida así:

$$\begin{aligned} \delta : Q \times \Gamma_\square &\to Q \times \Gamma_\square \times \{\leftarrow, \rightarrow, -\} \\ (q, s) &\mapsto \delta(q, s) = (q', s', D). \end{aligned}$$

Esto significa que cuando la unidad de control está en el estado q y escanea el símbolo s, la unidad de control sobre-escribe s por s', cambia al estado q' y realiza un desplazamiento $D \in \{\leftarrow, \rightarrow, -\}$.

La unidad de control se detiene al entrar en un estado de aceptación, i.e. no hay transiciones $\delta(q, s)$ donde $q \in F$.

Función de Transición

Instrucción	Acción de la unidad de control	Grafo
$\delta(q,s) = (q', s', \rightarrow)$	Se desplaza a la derecha	$q \xrightarrow{s\|s' \rightarrow} q'$
$\delta(q,s) = (q', s', \leftarrow)$	Se desplaza a la izquierda	$q \xrightarrow{s\|s' \leftarrow} q'$
$\delta(q,s) = (q', s', -)$	Permanece estacionaria	$q \xrightarrow{s\|s' -} q'$

Función de Transición

Instrucción	Acción de la unidad de control	Grafo
$\delta(q, s) = (q', \square, D)$	Borra el símbolo s y la unidad de control realiza el desplazamiento D, que puede ser $\rightarrow$ o $\leftarrow$ o $-$.	$q \xrightarrow{s \mid \square\, D} q'$
$\delta(q, \square) = (q', s, D)$	Escribe el símbolo s en la casilla vacía escaneada, y la unidad de control realiza el desplazamiento D, que puede ser $\rightarrow$ o $\leftarrow$ o $-$.	$q \xrightarrow{\square \mid s\, D} q'$

Descripción o Configuración Instantánea

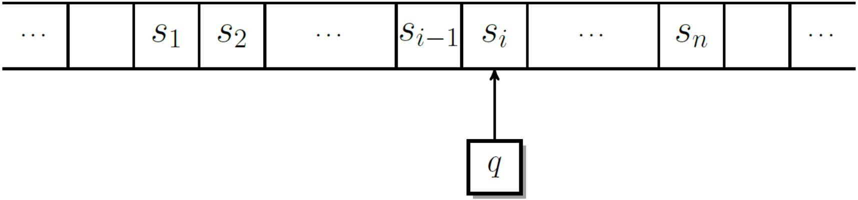
- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una Máquina de Turing donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.
- Una **descripción o configuración instantánea** es una secuencia de la forma

$$s_1 s_2 \cdots s_{i-1} q s_i s_{i+1}, s_{i+2} \cdots s_n,$$

  donde $s_1, s_2, \ldots, s_n \in \Gamma_\square$ y $q \in Q$.
- Esta descripción indica que
  - la unidad de control de M está en el estado q escaneando el símbolo s_i.
  - $s_1, s_2, \ldots, s_{i-1}$ son los símbolos escritos en las celdas anteriores.
  - $s_{i+1}, s_{i+2}, \ldots, s_n$ son los símbolos escritos en las celdas siguientes.
  - todas las celdas a la izquierda de s_1 y a la derecha de s_n contienen el símbolo $\square$.

Descripción o Configuración Instantánea

Como la cadena de entrada es finita, después de haber ejecutado un número finito de instrucciones, solo hay una porción finita de la cinta que no está totalmente en blanco.



$$aabq_2bABa$$
$$q_5ababca$$
$$ab\square\square aabq_0BbA$$
$$\square aX\square Ycq_3Bb\square$$

Configuración Inicial y de Aceptación

- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una Máquina de Turing donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.
- Una configuración instantánea es una secuencia de la forma uqv donde $u, v \in \Gamma_\square^*$ y $q \in Q$.
- La **configuración inicial** es $q_0 u$ donde $u \in \Sigma^*$ es la cadena de entrada.
- Una configuración instantánea vpw, donde $v, w \in \Gamma_\square^*$ y $p \in Q$ es una **configuración de aceptación** si $p \in F$. v y w son irrelevantes.

Paso Computacional

- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una Máquina de Turing donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.
- Sean $u_1 q u_2$ y $v_1 p v_2$ dos configuraciones instantáneas donde $u_1, u_2, v_1, v_2 \in \Gamma_\square^*$ y $q, p \in Q$.
- El paso de $u_1 q u_2$ a $v_1 p v_2$ se denomina **paso computacional** si se puede realizar por medio de una transición definida por δ. Se denota $u_1 q u_2 \vdash v_1 p v_2$.
- $u_1 q u_2 \vdash^k v_1 p v_2$ denota que se puede pasar de $u_1 q u_2$ a $v_1 p v_2$ en k pasos computacionales.
- $u_1 q u_2 \vdash^* v_1 p v_2$ que se puede en 0, 1 ó más pasos computacionales.
- $u_1 q u_2 \vdash^+ v_1 p v_2$ que se puede en 1 ó más pasos computacionales.

Lenguaje Aceptado por una Máquina de Turing

- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una Máquina de Turing donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.
- El **lenguaje aceptado por** M es:

$$L(M) := \{u \in \Sigma^* : q_0 u \vdash^* vpw \text{ donde } p \in F \text{ y } v, w \in \Gamma_\square^*\}.$$

- Es decir, una cadena u es aceptada por M si el procesamiento inicia en la configuración inicial y termina en una configuración de aceptación.

Finalización del Procesamiento

- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una Máquina de Turing donde $\Gamma_\square = \Gamma \cup \{\square\}$ y $\square$ es el caracter en blanco.
- Recordar que δ está definida de manera que no hay transiciones $\delta(p, s)$ donde $p \in F$ y $s \in \Gamma_\square$.
- Por lo anterior, si $u \in L(M)$, M se detiene al procesar u.
- Si $u \notin L(M)$ pueden darse dos casos:
  1. **Cómputo Abortado:** M se detiene en un estado que no es de aceptación. Para un $q \in (Q - F)$ y $s \in \Gamma_\square$, el procesamiento de $\delta(q, s)$ no está definido.
  2. **Ciclo Infinito:** M no se detiene nunca. Esto se denota como $q_0 u \vdash^* \infty$.

Familias de Lenguajes Aceptados

Las Máquinas de Turing originan las siguientes clases de Lenguajes:

1. L es un **Lenguaje Turing-Aceptable** si existe una MT M tal que $L(M) = L$.
2. L es un **Lenguaje Turing-Decidible** si existe una MT M tal que $L(M) = L$ y M se detiene con todas las cadenas de entrada.

- Un lenguaje Turing-decidible es Turing-aceptable pero lo recíproco no es cierto en general.
- El fenómeno de las máquinas que no se detienen no se puede eliminar de la Teoría de la Computación.

Lenguajes Regulares

- Los lenguajes regulares son Turing-decidibles.
- Los AFD se pueden simular con Máquinas de Turing.
- Sin embargo, hay que hacer una adaptación porque en los AFD sí se permiten transiciones desde los estados de aceptación.
- Entonces, la simulación de un AFD M por medio de una MT M' consiste en añadir en M':
  - un único estado de aceptación q_a.
  - transiciones desde los estados de aceptación originales hasta q_a para $\square$.

Teorema

Todo autómata finito se puede simular con una MT estándar. En consecuencia, todo lenguaje regular es Turing-decidible.

Lenguajes Regulares

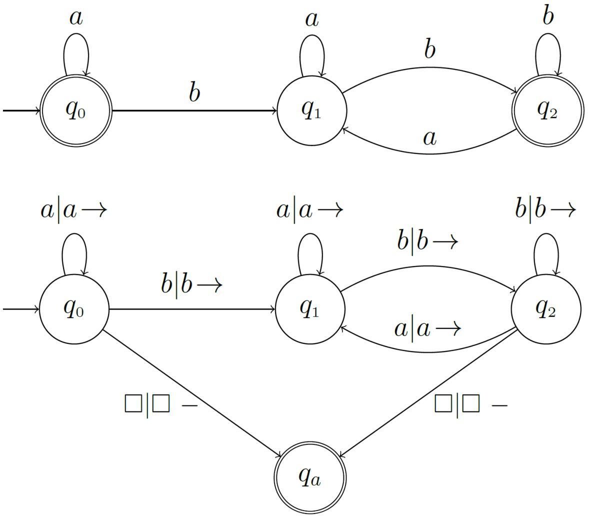
Demostración

- Todo autómata finito se puede converter a un AFD $M = (\Sigma, Q, q_0, F, \delta)$ equivalente.
- Se puede construir una MT $M' = (Q', q_0', F', \Sigma, \Gamma, \delta')$ donde

$$\begin{aligned} Q' &= Q \cup \{q_a\}, \\ \Gamma &= \Sigma, \\ F' &= \{q_a\}, \\ \delta'(q, s) &= (\delta(q, s), s, \rightarrow), (\forall q, s \in \Sigma), \\ \delta'(q, \square) &= (q_a, \square, -), (\forall q \in F). \end{aligned}$$

- M' se detiene con cualquier entrada $u \in \Sigma^*$ y $L(M) = L = (M')$. Por tanto, $L(M)$ es Turing-decidible.

Ejemplo

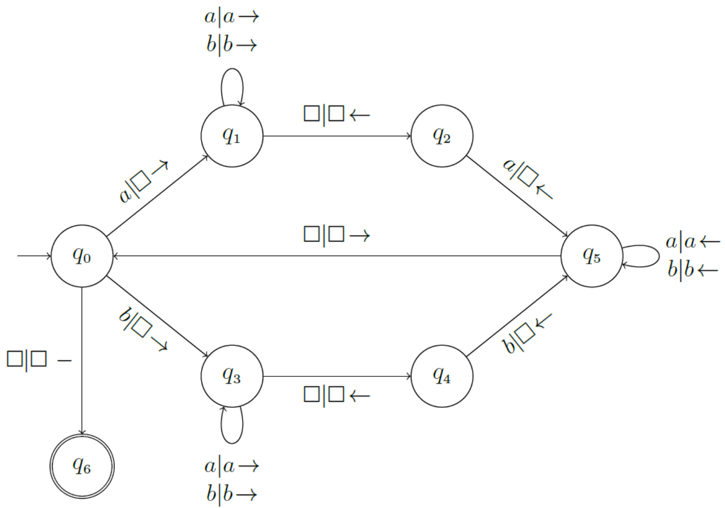


Ejemplo

Ejercicio

Dado $\Sigma = \{a, b\}$, diseñar una MT que acepte $L = \{ww^R : w \in \Sigma^*\}$.

Ejemplo

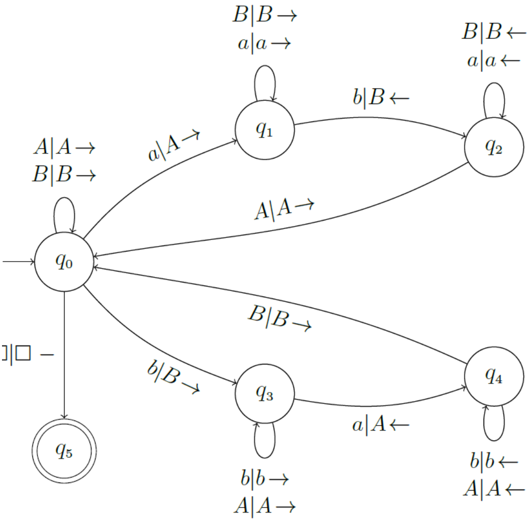

$a|a\rightarrow$
$b|b\rightarrow$
$\Box|\Box\leftarrow$
q_1
q_2
$a|\Box\rightarrow$
$a|\Box\leftarrow$
$\Box|\Box\rightarrow$
q_0
q_5
$a|a\leftarrow$
$b|b\leftarrow$
$b|\Box\rightarrow$
$b|\Box\leftarrow$
$\Box|\Box$ –
q_3
q_4
$\Box|\Box\leftarrow$
q_6
$a|a\rightarrow$
$b|b\rightarrow$

Ejemplo

Ejercicio

Dado $\Sigma = \{a, b\}$, diseñar una MT que acepte
$L = \{u \in \Sigma^* : \#_a(u) = \#_b(u)\}$.

Ejemplo

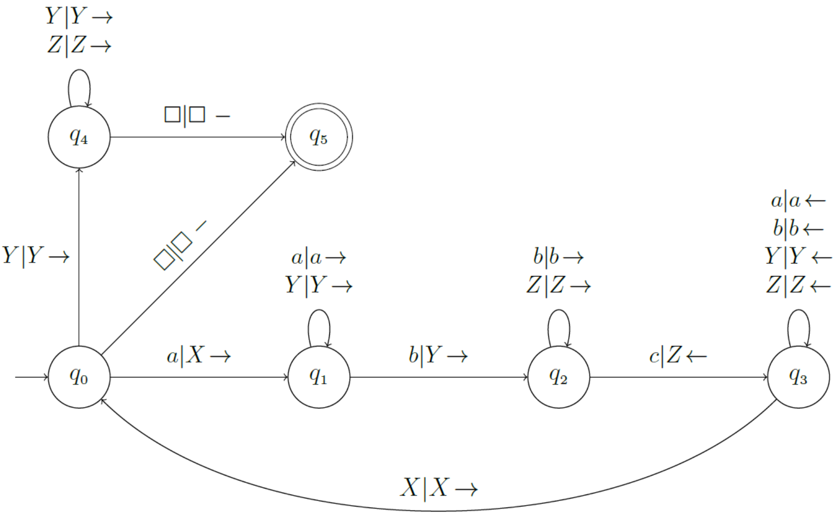

$B|B\rightarrow$
$a|a\rightarrow$
$B|B\leftarrow$
$a|a\leftarrow$
$b|B\leftarrow$
q_1
q_2
$A|A\rightarrow$
$B|B\rightarrow$
$a|A\rightarrow$
$A|A\rightarrow$
q_0
$B|B\rightarrow$
$\Box|\Box -$
$b|B\rightarrow$
q_3
q_4
$a|A\leftarrow$
q_5
$b|b\rightarrow$
$A|A\rightarrow$
$b|b\leftarrow$
$A|A\leftarrow$

Ejemplo

Ejercicio

Dado $\Sigma = \{a, b, c\}$, diseñar una MT que acepte $L = \{a^n b^n c^n : n \geq 0\}$.

Ejemplo

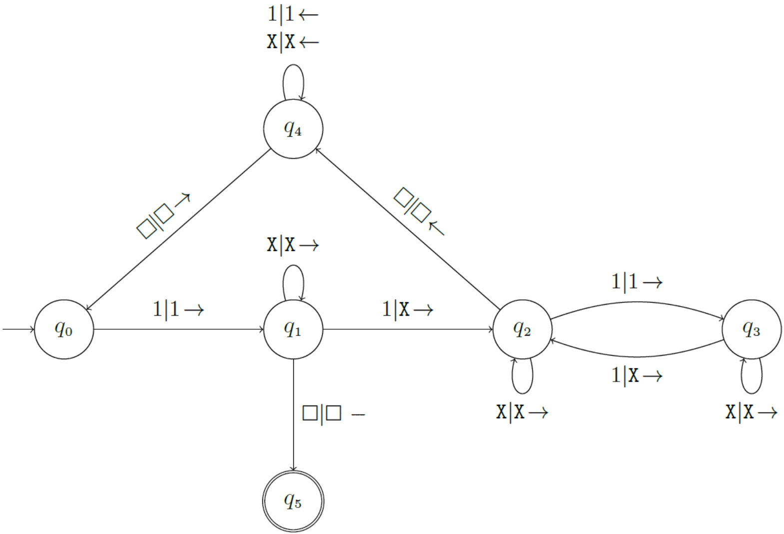

$Y|Y\rightarrow$
$Z|Z\rightarrow$
$\square|\square\ -$
q_4
q_5
$Y|Y\rightarrow$
$\square|\square\ -$
$a|a\rightarrow$
$Y|Y\rightarrow$
$b|b\rightarrow$
$Z|Z\rightarrow$
$a|a\leftarrow$
$b|b\leftarrow$
$Y|Y\leftarrow$
$Z|Z\leftarrow$
$a|X\rightarrow$
$b|Y\rightarrow$
$c|Z\leftarrow$
q_0
q_1
q_2
q_3
$X|X\rightarrow$

Ejemplo

Ejercicio

Dado $\Sigma = \{1\}$, diseñar una MT que acepte
$L = \{1^{2^n} : n \geq 0\} = \{1, 1^2, 1^4, 1^8, 1^{16}, 1^{32}, \ldots\}$.

Ejemplo

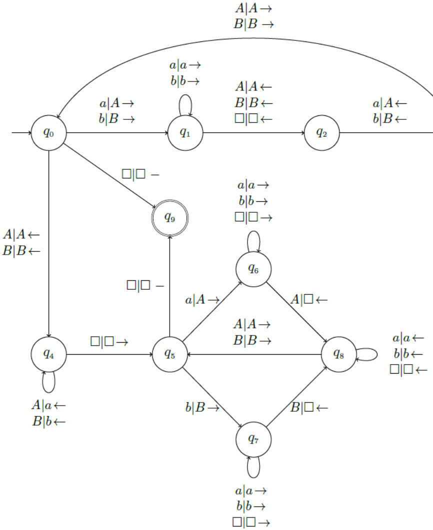


Ejemplo

Ejercicio

Dado $\Sigma = \{a, b\}$, diseñar una MT que acepte $L = \{ww : w \in \Sigma^*\}$.

Ejemplo



$A|A \rightarrow$
$B|B \rightarrow$
$a|a \rightarrow$
$b|b \rightarrow$
$a|A \rightarrow$
$b|B \rightarrow$
$A|A \leftarrow$
$B|B \leftarrow$
$\square|\square \leftarrow$
$a|A \leftarrow$
$b|B \leftarrow$
q_0
q_1
q_2
q_3
$a|a \leftarrow$
$b|b \leftarrow$
$\square|\square -$
q_9
$a|a \rightarrow$
$b|b \rightarrow$
$\square|\square \rightarrow$
$A|A \leftarrow$
$B|B \leftarrow$
q_6
$\square|\square -$
$a|A \rightarrow$
$A|\square \leftarrow$
$A|A \rightarrow$
$B|B \rightarrow$
q_4
$\square|\square \rightarrow$
q_5
q_8
$a|a \leftarrow$
$b|b \leftarrow$
$\square|\square \leftarrow$
$A|a \leftarrow$
$B|b \leftarrow$
$b|B \rightarrow$
$B|\square \leftarrow$
q_7
$a|a \rightarrow$
$b|b \rightarrow$
$\square|\square \rightarrow$

Ejercicios

Ejercicio

Dado $\Sigma = \{a, b, c\}$, diseñar una MT que acepten los siguientes lenguajes:

1. $L = \{a^n b^{n+1} c^{n+2} : n \geq 0\}$.
2. $L = \{a^n b^{2n} c^n : n \geq 0\}$.
3. $L = \{a^k b^m c^n : 0 \leq k \leq m \leq n\}$.
4. $L = \{a^k b^m c^n : 0 \leq k < m < n\}$.
5. $L = \{a^k b^m c^n : k \geq m \geq n \geq 0\}$.

Outline

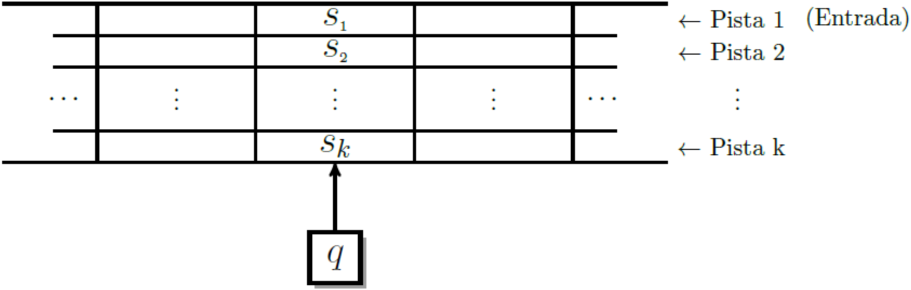
MT con Estado Único de Aceptación

- Sea $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ una MT que puede tener varios estados de aceptación donde $\Gamma_{\square} = \Gamma \cup \{\square\}$. Esta se puede simular a través de una máquina M' con único estado de aceptación.
- Esto se puede hacer porque en la definición del Modelo estándar se exige que la unidad de control se detenga siempre que llega a un estado de aceptación.
- **Simulación:**
  1. Remover todos los estados de aceptación.
  2. Agregar un único estado de aceptación q_a.
  3. Cada transición $\delta(q, s) = (p, s', D)$, donde $q, p \in Q$, $p \in F$, $s, s' \in \Gamma_{\square}$ y $D = \{\leftarrow, \rightarrow, -\}$, se reemplaza por $\delta(q, s) = (q_a, s', D)$.
  4. La máquina resultante es $M' = ((Q - F) \cup \{q_a\}, q_0, \{q_a\}, \Sigma, \Gamma, \delta)$.

Outline

MT con Cinta Dividida en Pistas

- La cinta está dividida en k pistas.
- La cadena de entrada se pone en la **pista de entrada**.
- El resto de pistas están en blanco. Se usan para poner marcas.



MT con Cinta Dividida en Pistas

- Sea $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, una MT con k pistas.
- En un paso computacional, la unidad de control cambia simultáneamente el contenido de las k pistas de la casilla escaneada y luego uno de los desplazamientos $\leftarrow$, $\rightarrow$, o $-$.
- La función de transición es:

$$\delta : Q \times (\Gamma_\square)^k \to Q \times (\Gamma_\square)^k \times \{\to, \leftarrow, -\}$$
$$(q, (s_1, s_2, \ldots, s_k)) \mapsto \delta(q, (s_1, s_2, \ldots, s_k)) = (q', (s'_1, s'_2, \ldots, s'_k), D).$$

MT con Cinta Dividida en Pistas

Simulación:

- Sea $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, una MT con k pistas.
- M se puede simular con una MT estándar M' en la que el alfabeto de cinta está formado por el conjunto de k-tuplas $(s_1, s_2, \ldots, s_k)$ donde $s_i \in \Gamma_\square$, para $1 \leq i \leq k$.
- Es decir, el alfabeto de cinta es $(\Gamma_\square)^k$.
- Entonces, $M' = (Q, q_0, F, \Sigma, (\Gamma_\square)^k, \delta)$.

Ejercicio

Ejercicio

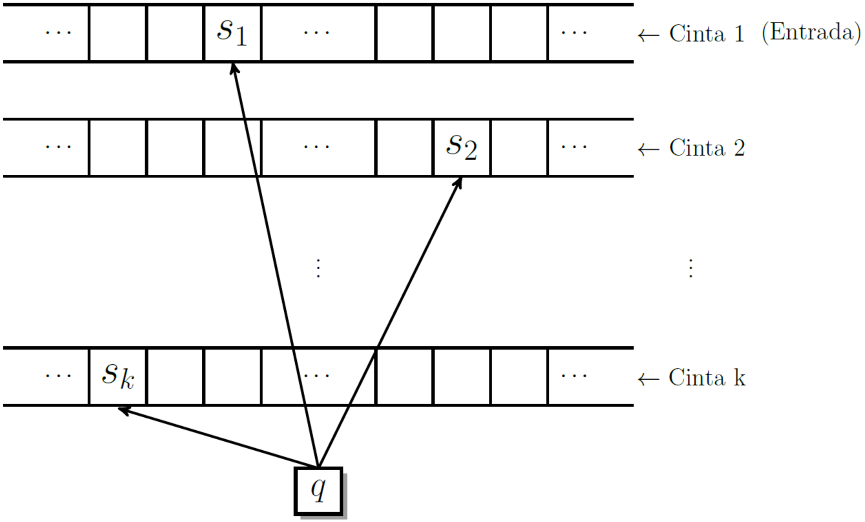
Dibujar una MT dividida en dos pistas que, dado $\Sigma = \{a, b\}$, acepte $L = \{a^n b^n c^n : n \geq 0\}$.

Outline

MT con Múltiples Cintas

- En el modelo multi-cinta hay k cintas diferentes, con $k \geq 2$ es finito, donde cada cinta está dividida en celdas.
- La unidad de control tiene k visores que leen una celda de cada cinta.
- La cadena de entrada se pone en la **cinta de entrada**, i.e. la primera.
- Las otras cintas están inicialmente en blanco.
- En un paso computacional, la unidad de control cambia el contenido en la casilla escaneada de cada cinta y luego realiza un desplazamiento $\rightarrow$, $\leftarrow$ o $-$ (en cada cinta).
- La operación en cada cinta está definida de manera independiente.

MT con Múltiples Cintas

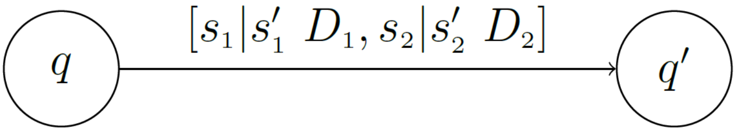


Función de Transición

- Sea $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, una MT con k cintas.
- La función de transición δ está definida de la siguiente manera:

$$\begin{aligned}\delta : Q \times (\Gamma_\square)^k &\to Q \times (\Gamma_\square \times \{\to, \leftarrow, -\})^k \\ (q, (s_1, s_2, \ldots, s_k)) &\mapsto \delta(q, (s_1, s_2, \ldots, s_k)) \\ &= (q', (s_1', D_1), (s_2', D_2), \ldots, (s_k', D_k)).\end{aligned}$$

- Se puede representar por medio de grafos:



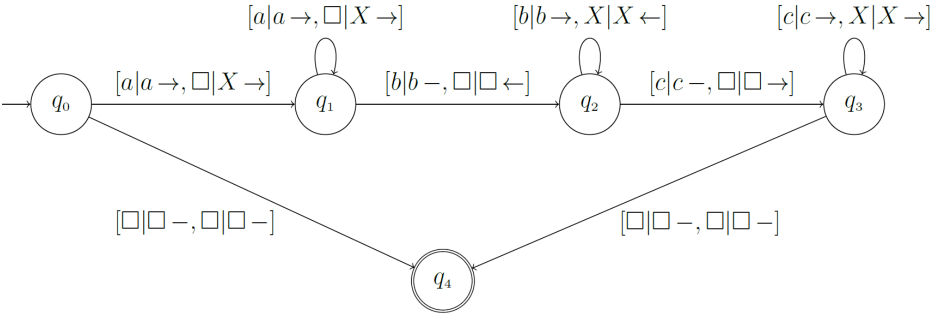
Simulación

- Sea $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, una MT con k cintas.
- M se puede simular con una MT M' dividida en pistas.
- Cada cinta se representa con dos pistas de la máquina simuladora: una para el contenido y otra que solo tiene el símbolo especial X que indica en qué posición está el visor de la cinta.
- Además, se tiene una pista adicional con los símbolos especiales Y y Y' para indicar las posiciones de más a la izquierda y de más a la derecha de la unidad de control en la máquina original.
- Para simular un solo paso computacional, la nueva máquina requiere hacer múltiples recorridos de izquierda a derecha, mientras va actualizando los X, Y e Y'.

Ejemplo

Ejercicio

Diseñar una MT con dos cintas que acepte $L = \{a^n b^n c^n : n \geq 0\}$.



Ejemplo

Ejercicio

Diseñar una MT con dos cintas que acepte $L = \{a^n b^n c^n : n \geq 0\}$.

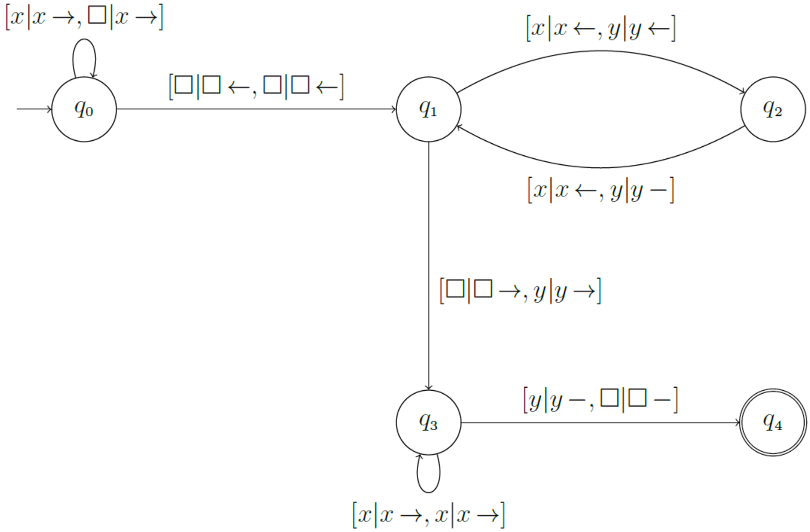
$$
\begin{aligned}
\delta(q_0, (a, \square)) &= (q_1, (a, \rightarrow), (X, \rightarrow)),\\
\delta(q_1, (a, \square)) &= (q_1, (a, \rightarrow), (X, \rightarrow)),\\
\delta(q_1, (b, \square)) &= (q_2, (b, -), (\square, \leftarrow)),\\
\delta(q_2, (b, X)) &= (q_2, (b, \rightarrow), (X, \leftarrow)),\\
\delta(q_2, (c, \square)) &= (q_3, (c, -), (\square, \rightarrow)),\\
\delta(q_3, (c, X)) &= (q_3, (c, \rightarrow), (X, \rightarrow)),\\
\delta(q_3, (\square, \square)) &= (q_4, (\square, -), (\square, -)),\\
\delta(q_0, (\square, \square)) &= (q_4, (\square, -), (\square, -)).
\end{aligned}
$$

Ejemplo

Ejercicio

Diseñar una MT que acepte $L = \{ww : w \in \Sigma^+\}$, donde $\Sigma = \{a, b\}$.

Ejemplo


$[x|x \rightarrow, \square|x \rightarrow]$
q_0
$[\square|\square \leftarrow, \square|\square \leftarrow]$
q_1
$[x|x \leftarrow, y|y \leftarrow]$
q_2
$[x|x \leftarrow, y|y -]$
$[\square|\square \rightarrow, y|y \rightarrow]$
q_3
$[y|y -, \square|\square -]$
q_4
$[x|x \rightarrow, x|x \rightarrow]$

Ejercicio

Ejercicio

Dado $\Sigma = \{a, b, c\}$, diseñar MT con múltiples cintas que acepten los siguientes lenguajes:

1. $L = \{a^n b^{2n} c^n : n \geq 0\}$.
2. $L = \{a^k b^m c^n : 0 \leq k \leq m \leq n\}$.
3. $L = \{a^k b^m c^n : k \geq m \geq n \geq 0\}$.

Outline

Máquina de Turing No Determinista (MTN)

- Tiene los mismos componentes del Modelo Estándar con un único estado de aceptación.
- La diferencia es que se permite que, en un paso computacional, la unidad de control escoja aleatoriamente entre varias transiciones.
- La función de transición de una MTN $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, es:

$$\begin{aligned}\delta : Q \times \Gamma_\square &\to \mathcal{P}(Q \times \Gamma_\square \times \{\rightarrow, \leftarrow, -\})\\ (q, s) &\mapsto \delta(q, s)\\ &= \{(p_1, s_1, D_1), (p_2, s_2, D_2), \ldots (p_k, s_k, D_k)\}.\end{aligned}$$

- Una cadena es aceptada si existe al menos un procesamiento en el que sea aceptada, i.e. una *computación de aceptación*.

Máquina de Turing No Determinista (MTN)

Teorema

Todo lenguaje aceptado por una MTN $M = (Q, q_0.F, \Sigma, \Gamma, \delta)$, donde $\Gamma_\square = \Gamma \cup \{\square\}$, puede ser aceptado por una MT estándar M'.

Demostración

- Si una cadena $u \in \Sigma^*$ es aceptada por M, M' la acepta después de realizar una búsqueda exhaustiva entre todas las computaciones posibles.
- Primero se examinan las computaciones de 1 paso, luego las de 2 y así sucesivamente hasta encontrar una configuración de aceptación.

Máquina de Turing No Determinista (MTN)

Demostración

- Sea n el tamaño máximo de un $\delta(q, s)$ para $q \in Q$ y $s \in \Gamma_\square$.
- Para las $\delta(q, s)$ de tamaño menor a n se pueden repetir algunas opciones. Entonces estas opciones se pueden numerar.
- Sea $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ un alfabeto disyunto a $\Gamma_\square$. Estos símbolos pueden indexar las opciones de los $\delta(q, s)$.
- Entonces, una cadena $x = x_1 x_2 \cdots x_m$, donde $x \in \Phi^*$, representa un cómputo de m pasos. El valor de x_j representa qué opción se escoge en el paso j.

$$\delta(q, s) = \{\underbrace{(p_1, b_1, D_1)}_{\varphi_1}, \underbrace{(p_2, b_2, D_2)}_{\varphi_2}, \ldots, \underbrace{(p_n, b_n, D_n)}_{\varphi_n}\} \quad \leftarrow \text{índice}$$

Máquina de Turing No Determinista (MTN)

Demostración

La simulación se puede realizar con 3 cintas:

1. Cadena de entrada $u \in \Sigma^*$.
2. Cadena de cómputo simulado.
3. Simulación de dicho cómputo en la cadena de entrada.

Se realizan 3 etapas:

A. Copiar la cadena u en la Cinta 1 y escribir $x = \varphi_1$ en la Cinta 2.

B. Si $x = x_1 x_2 \cdots x_k$, donde $x \in \Phi^*$, es la cadena escrita en la Cinta 2, simular el cómputo de los k pasos correspondientes en u en la Cinta 3. Si no se acepta, pasar a C.

C. M' reemplaza la cadena x por la cadena que sigue en orden lexicográfico y se retorna a B.

Máquina de Turing No Determinista (MTN)

Demostración

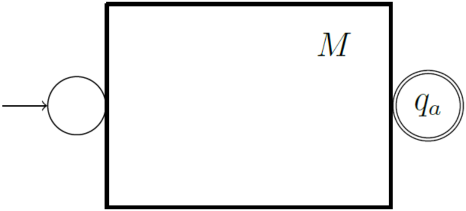
- Si $y = y_1 y_2 \cdots y_k$, donde $y \in \Phi^*$, es el cómputo de aceptación de u en M, dicho cómputo se explora cuando y esté en la Cinta 2. Por esto, $L(M) \subseteq L(M')$.
- Además, $L(M') \subseteq L(M)$ porque M' solo puede aceptar cadenas que son aceptadas por M.
- Entonces $L(M) = L(M')$.
- Si $u \notin L(M)$, M' entra en un bucle infinito.

Ejemplo

Ejercicio

Para un lenguaje L sobre $\Sigma = \{a, b\}$, se define P_L como el lenguaje de todos los prefijos de cadenas de L: $P_L = \{x \in \Sigma^* : (\exists y \in \Sigma^*)[xy \in L]\}$.

Sea M un autómata que acepta L, representado así:



Ejemplo

Ejercicio

Para un lenguaje L sobre $\Sigma = \{a, b\}$, se define P_L como el lenguaje de todos los prefijos de cadenas de L: $P_L = \{x \in \Sigma^* : (\exists y \in \Sigma^*)[xy \in L]\}$.

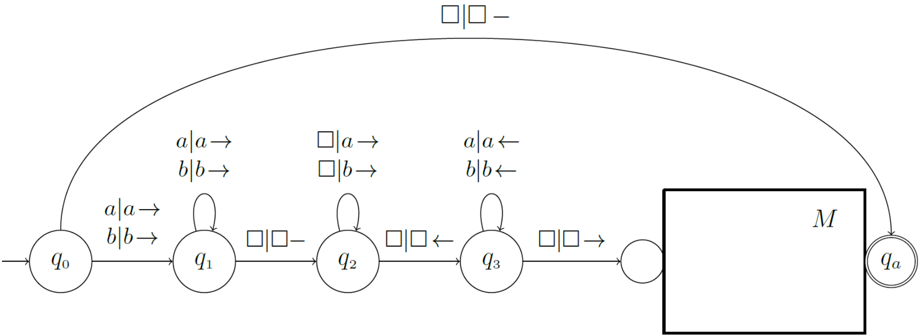
Idea

1. Leer la cadena de entrada x e ir al final.
2. De manera no determinista, escribir aleatoriamente una cadena y perteneciente a Σ^*.
3. Correr la MT M con la entrada xy. Si M acepta xy, se acepta x.

Muchos algoritmos no deterministas tienen esta estructura:
NO-DETERMINISTIMO = CONJETURA + VERIFICACIÓN

Ejemplo

Entonces, la siguiente MTN acepta P_L:



MTN con Varias Cintas

El modelo multicinta también se puede construir de manera no determinista: se definen varias opciones para cada transición.

Teorema

Todo lenguaje aceptado por una MTN multicinta puede ser aceptado por una MT estándar.

Ejercicio

Ejercicio

Sea L un lenguaje sobre $\Sigma = \{a, b\}$ y M una MT, modelo estándar, tal que $L(M) = L$. Construir MTNs que acepten los siguientes lenguajes:

1. El lenguaje S_L de todos los sufijos de L.
2. El lenguaje C_L de todas lass subcadenas de L.

Outline

Funciones Parciales y Totales

Función Parcial y Total

- Sean Σ y Γ dos alfabetos.
- Una función $f : \Sigma^* \to \Gamma^*$, cuyo dominio $dom(f) \subset \Sigma^*$, es una **función parcialmente definida** o **función parcial**.
- Una función $f : \Sigma^*\Gamma^*$, cuyo dominio $dom(f) = \Sigma^*$, es una **función totalmente definida** o **función total**.

Funciones Turing-Computables

Como las MT tienen la capacidad de transformar entradas en salidas, se pueden utilizar como mecanismo para calcular funciones.

Función Turing-Computable

- Una función parcial o total $f : \Sigma^* \to \Gamma^*$, cuyo dominio $dom(f) \subseteq \Sigma^*$, es **Turing-computable** o **Turing-calculable** si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$, tal que para toda cadena $u \in \Sigma^*$,

$$u \in dom(f) \Rightarrow q_0 u \vdash^* q_a v, \text{ donde } v = f(u).$$

- Se dice que M **calcula** o **computa** la función f.

Funciones con Varios Argumentos

La noción de función Turing-computable se puede extender a funciones de varios argumentos.

Función Turing-Computable con Varios Argumentos

- Sea $(\Sigma^*)^k = \Sigma^* \times \Sigma^* \times \cdots \Sigma^*$ (k veces).
- Una función $f : (\Sigma^*)^k \to \Gamma^*$ parcial o total, con dominio $dom(f) \subseteq \Sigma^*$, es **Turing-computable** si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$ tal que, para toda k-tupla $(u_1, u_2, \ldots, u_k) \in (\Sigma^*)^k$ se tiene:

$$(u_1, u_2, \ldots, u_k) \in dom(f) \Rightarrow q_0 u_1 \Box u_2 \Box \cdots \Box u_k \vdash^* q_a v,$$

donde $v = f(u_1, u_2, \ldots, u_k)$.

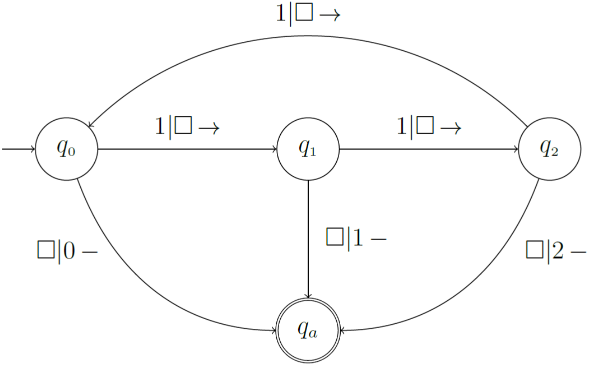
Funciones Numéricas

- Para funciones numéricas, se puede tomar $\Sigma = \{1\}$ y el sistema de numeración unitario: si n es natural, puede ser representado como 1^n y la cadena vacía representa el 0.
- $\mathbb{N} = \{1^n : n \geq 0\} = \{\lambda, 1, 1^2, 1^3, \ldots\}$.
- Una función numérica $f : \mathbb{N} \to \mathbb{N}$ es Turing-computable si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$ tal que $q_0 1^m \vdash^* q_a 1^n$ siempre que $f(m) = n$, con $m, n \in \mathbb{N}$.

Ejemplo

Ejercicio

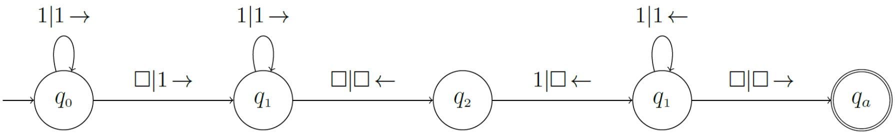
Diseñar una MT M que calcule $f(n) = n \mod 3$, para cualquier número natural $n \geq 0$, escrito en el sistema de numeración unitario.



Ejemplo

Ejercicio

Diseñar una MT M que calcule la función parcial suma $f(m, n) = m + n$, donde $m, n \geq 1$, escritos en el sistema de numeración unitario.



Orden Lexicográfico

Orden Lexicográfico

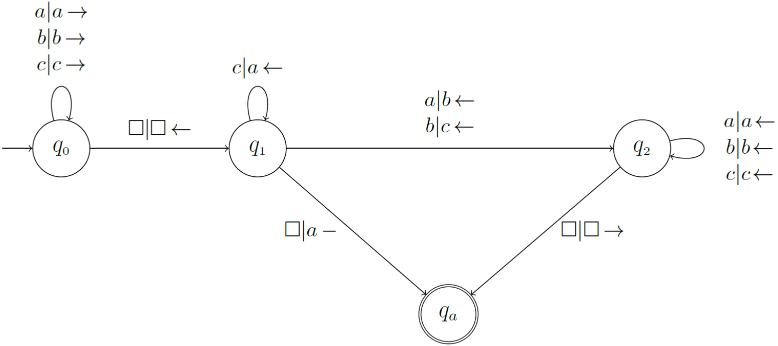
- Sea $\Sigma = \{s_1, s_2, \ldots, s_k\}$ un alfabeto dado en el cual los símbolos tienen un orden preestablecido $s_1 < s_2 < \cdots < s_k$.
- En el conjunto Σ^*, se define el **orden lexicográfico**, denotado por $<$, de la siguiente manera.
- Sean $u, v \in \Sigma^*$,
  - $u = a_1 a_2 \cdots a_m$ donde $a_i \in \Sigma$ para $1 \leq i \leq m$.
  - $v = b_1 b_2 \cdots b_n$ donde $b_i \in \Sigma$ para $1 \leq i \leq n$.
- Se tiene que $u < v$ si:
  1. $|u| < |v|$, o
  2. $|u| = |v|$ y para algún índice i, $1 \leq i \leq m$, se cumple que

$$a_1 = b_1, a_2 = b_2, \ldots, a_{i-1} = b_{i-1}, a_i < b_i.$$

Ejemplo

Ejercicio

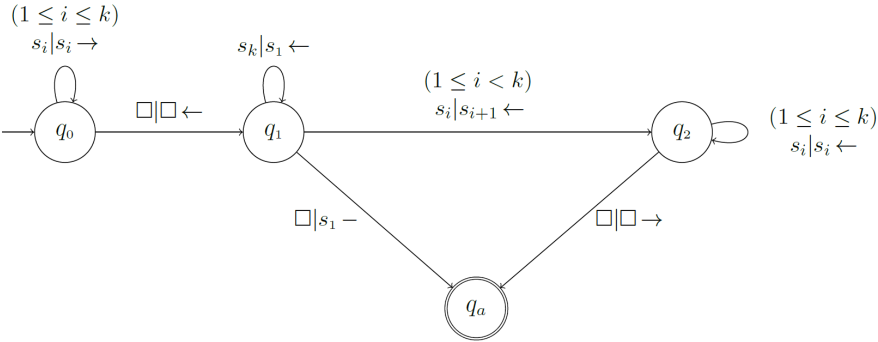
Sea $\Sigma = \{a, b, c\}$, donde $a < b < c$: Diseñar una MT M que calcule la función $f : \Sigma^* \to \Sigma^*$, definida por: $f(u) =$ cadena que sigue a u en el orden lexicográfico.



Ejemplo

Ejercicio

Sea $\Sigma = \{s_1, s_2, \ldots, s_k\}$, donde $s_1 < s_2 < \cdots < s_k$: Diseñar una MT M que calcule la función $f : \Sigma^* \to \Sigma^*$, definida por: $f(u) =$ cadena que sigue a u en el orden lexicográfico.



Ejercicios

Ejercicio

Diseñar MT que calculen las siguientes funciones numéricas, utilizando para $\mathbb{N}$ el sistema unitario:

1. $f : \mathbb{N} \to \mathbb{N}$, donde $f(n) = 2n$.
2. $f : \mathbb{N} \to \mathbb{N}$, donde $f(n) = 2^n$.
3. $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, donde $f(m, n) = \text{máx}(m, n)$.

Outline

Outline

Tesis de Church-Turing

Tesis de Church-Turing

Todo algoritmo puede ser descrito por medio de una MT.

- La MT antecedió en varias décadas la implementación física de los computadores actuales.
- Sin embargo, es un modelo muy útil para representar lo computable.
- No se un enunciado matemático susceptible a demostración.
- Sin embargo, sí se podría refutar mostrando un procedimiento que todos acepten como algoritmo y no pueda ser descrito por medio de una MT.
- Después de décadas de investigación, no se ha podido encontrar uno.

Tesis de Church-Turing

Hechos que contribuyen a apoyar la tesis:

1. La adición de recursos computacionales (múltiples pistas o cintas, no determinismo, etc.) no incrementa el poder computacional de las MT. Esto demuestra que el modelo es robusto y representa el límite de lo que un dispositivo de computación secuencial puede hacer.
2. Los modelos computacionales para describir la noción de algoritmo han resultado ser equivalentes a la MT:
    - Funciones parciales recursivas (Modelo de Godel y Kleene, 1936).
    - Cálculo-λ (Church, 1936).
    - Sistemas de deducción canónica (Post, 1943).
    - Lógica combinatoria (Schonfinkel,1924; Curry, 1929).
    - Algoritmos de Markov (Markov, 1951).
    - Gramáticas irrestrictas (Chomsky, 1956).
    - Máquinas de registro (Shepherdson-Sturgis, 1963).

Outline

Codificación Binaria

- Toda MT se puede codificar como una secuencia binaria finita.
- A continuación se presenta un esquema de codificación:
  1. Estados disponibles: q_1, q_2,..., donde q_1 es el estado inicial y q_2 es el único estado de aceptación.
  2. Símbolos de cinta disponibles: s_1, s_2, ..., s_m, s_{m+1}, s_{m+2}, ... donde
     - $s_1 = \square$,
     - $\Sigma = \{s_2, s_3, \ldots, s_m\}$ es el alfabeto de entrada común de todas las MT, y
     - s_k, $k > m$ son los símbolos auxiliares de una MT dada.
- Cada máquina posee un número finito de estados y de símbolos de cinta, pero hay una secuencia infinita de símbolos de ambos tipos disponibles para todas las MT.

Codificación Binaria

- Una instrucción determinada I en una MT, $\delta(q_i, s_j) = (q_k, s_\ell, D_t)$, se codifica por medio de una cadena binaria:

$$01^i01^j01^k01^\ell01^t0,$$

donde $t = 1$ representa $\rightarrow$, $t = 1$ representa $\leftarrow$ y $t = 2$ representa 3.

- La codificación de la instrucción I se denota como $\langle I \rangle$.
- Una MT M está definida completamente por su función de transición:

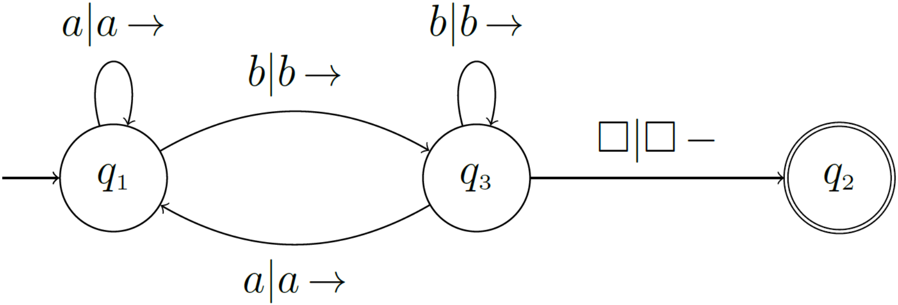
$$\langle M \rangle = \langle I_1 \rangle \langle I_2 \rangle \cdots \langle I_r \rangle 00.$$

- Las instrucciones pueden ir en cualquier orden y se pueden intercambiar el nombre de estados (excepto q_1 y q_2).
- Por tanto, una MT tiene infinitas codificaciones posibles.

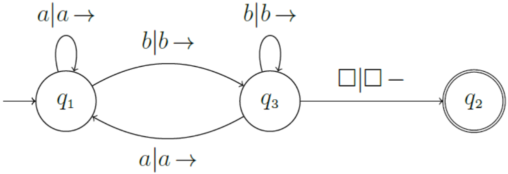
Ejemplo

Ejercicio

Sea $\Sigma = \{a, b\}$ y M la siguiente MT. Encuentre una codificación de M:



Ejemplo



$s_1 = \square$
$s_2 = a$
$s_3 = b$

$I_1 : \ \delta(q_1, s_2) = (q_1, s_2, \rightarrow)$.	Codificación de I_1:	$\underline{0101101011010}$.
$I_2 : \ \delta(q_1, s_3) = (q_3, s_3, \rightarrow)$.	Codificación de I_2:	$\underline{01011101110111010}$.
$I_3 : \ \delta(q_3, s_2) = (q_1, s_2, \rightarrow)$.	Codificación de I_3:	$\underline{011101101011010}$.
$I_4 : \ \delta(q_3, s_3) = (q_3, s_3, \rightarrow)$.	Codificación de I_4:	$\underline{0111011101110111010}$.
$I_5 : \ \delta(q_3, s_1) = (q_2, s_1, -)$.	Codificación de I_5:	$\underline{0111010110101110}$.

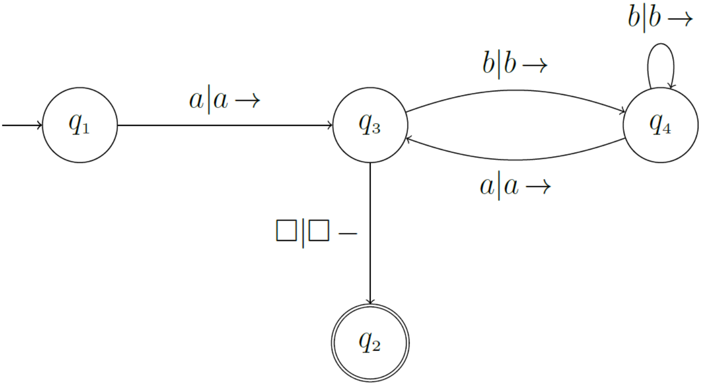
$$\langle M \rangle = \langle I_1 \rangle \langle I_2 \rangle \langle I_3 \rangle \langle I_4 \rangle \langle I_5 \rangle 00.$$

$$\begin{aligned}\langle M \rangle &= \underline{0101101011010}\,\underline{01011101110111010}\,\underline{011101101011010}\,\underline{0111011101110111010}\,\underline{0111010110101110}00 \\ &= \underline{0101^2 0101^2 010}\,\underline{01 01^3 01^3 01^3 010}\,\underline{01^3 01^2 0101^2 010}\,\underline{01^3 01^3 01^3 01^3 010}\,\underline{01^3 0101^2 0101^3 0}00\end{aligned}$$

Ejercicio

Ejercicio

Sea $\Sigma = \{a, b\}$ y M la siguiente MT. Encuentre el lenguaje aceptado por M y una de sus codificaciones.



Código Válido

- No toda secuencia binaria puede representar una MT.
- Una cadena binaria que codifique una MT se denomina **código válido** o **codificación válida**.
- Se puede diseñar un algoritmo que determine si una cadena binaria finita codifica o no una MT.

Algoritmo para Verificar Validez de un Código

Una cadena $u \in \{0,1\}^*$ es un código válido de una MT si:

1. $u \in (01^+01^+01^+01^+01^+0)^+00$. Recordar que $\delta(q_i, s_j) = (q_k, s_\ell, D_t)$ se representa como $01^i01^j01^k01^\ell01^t0$.
2. En cada bloque (instrucción) $01^i01^j01^k01^\ell01^t0$, $t \in \{1,2,3\}$.
3. No hay dos bloques que comiencen con la misma subcadena 01^i01^j0.
4. No hay ningún bloque que comience con 01^20.

Decodificación & Lenguajes

- Una MT puede tener infinitas codificaciones, pero un código válido se puede decodificar y representa una única MT.
- Si una cadena binaria no es un código válido de una MT, se considera que es una MT que no acepta ninguna cadena, i.e. $\emptyset$.
- Todas las cadenas binarias representan MTs.
- Dado que $\{0,1\}^*$ se puede ordenar lexicográficamente, se puede establecer el concepto de la i-ésima MT denotada por M_i.
- Cada MT aparece varias veces.
- Las MT que aceptan $\emptyset$ aparecen infinitas veces.

Outline

Lenguajes Turing-aceptables

La enumeración M_1, M_2, M_3, ... de todas las MT sobre el alfabeto Σ induce una enumeración de todos los lenguajes Turing-aceptables: $L_1 = L(M_1)$, $L_2 = L(M_2)$, ...

Teorema

Dado el alfabeto Σ, el conjunto de todas las MT con alfabeto Σ es infinito enumerable. También lo es el conjunto de todos los lenguajes Turing-aceptables.

Lenguajes que no son Turing-aceptables

- Utilizando argumentos de cardinalidad, es posible concluir que existen lenguajes que no son Turing-aceptables.
- Dado un alfabeto Σ, $\mathcal{P}(\Sigma^*)$ es el conjunto de todos los lenguajes sobre Σ.
- Por el Teorema de Cantor, $\mathcal{P}(\Sigma^*)$ no es enumerable, pero según el teorema anterior, el conjunto de todos los lenguajes Turing-aceptables sí lo es.
- Por tanto, existen lenguajes que no pueden ser aceptados por ninguna máquina de Turing.

Ejemplo de un Lenguaje que no es Turing-aceptables

- Dado un alfabeto de entrada $\Sigma = \{s_2, s_3, \ldots, s_m\}$, las cadenas de Σ^* se pueden ordenar lexicográficamente estableciendo un orden. Por ejemplo, $s_2 < s_3 < \cdots < s_m$.
- De esta manera, se obtiene una enumeración w_1, w_2, ... de todas las cadenas de Σ^*.
- La interacción entre estas dos enumeraciones diferentes w_1, w_2, ... y M_1, M_2, ... produce un lenguaje que no es Turing-aceptable.
- En particular, este lenguaje es **lenguaje de la diagonalización**

$$L_d = \{w_i \in \Sigma^* : w_i \text{ no es aceptada por } M_i\}.$$

Ejemplo de un Lenguaje que no es Turing-aceptables

Teorema

Dado el alfabeto Σ, el lenguaje
$L_d = \{w_i \in \Sigma^* : w_i$ no es aceptada por $M_i\}$ no es Turing-aceptable.

Demostración (por contradicción)

- Si L_d fuera Turing-aceptable, existiría una MT M_k tal que $L_d = L(M_k)$.
- Entonces, se tendría:

$$w_k \in L_d \Rightarrow w_k \text{ no es aceptada por } M_k \Rightarrow w_k \notin L(M_k) = L_d.$$
$$w_k \notin L_d \Rightarrow w_k \text{ es aceptada por } M_k \Rightarrow w_k \in L(M_k) = L_d.$$

- Entonces, $w_k \in L \iff w_k \notin L_d$, lo cual es una contradicción.

Outline

Codificación de una Cadena

- Se adopta la codificación binaria mostrada anteriormente para el alfabeto de entrada $\Sigma = \{s_2, s_3 \dots s_m\}$.
- Ahora se complementa con la codificación de las cadenas de Σ^*.
- El símbolo s_i se codifica como 1^i.
- Una cadena $s_{i_1} s_{i_2} \cdots s_{i_k}$ se codifica como $1^{i_1}01^{i_2}0 \cdots 01^{i_k}0$.
- Una **codificación válida de una cadena** sigue esta estructura . Además todo i_j debe cumplir con que $2 \leq i_j \leq m$.
- La codificación de cadenas sí es única.
- La codificación de $w \in \Sigma^*$ se denota como $\langle w \rangle$.
- **Ejemplo:** Dado $\Sigma = \{a, b, c, d\}$, donde $s_2 = a$, $s_3 = b$, $s_4 = c$, $s_5 = d$, $\langle dbabc \rangle = 1^5 01^3 01^2 01^3 01^4 0$.

MT Universal

- Una MT universal $M_{\mathcal{U}}$ simula el funcionamiento de todas las MT sobre el alfabeto de entrada Σ.
- $M_{\mathcal{U}}$ está diseñada para procesar cadenas binarias de la forma $\langle M\rangle\langle w\rangle$, siendo M una MT determinada y $w \in \Sigma^*$.
- $M_{\mathcal{U}}$ es una MT con tres cintas cuyo alfabeto de cinta es $\{0, 1\}$ y el símbolo externo $\square$.
- Las entradas son cadenas binarias que se escriben en la primera cinta la cual es de solo lectura. Las otras están en blanco.

Funcionamiento de la MT Universal

1. Copia la entrada en la Cinta 2 y verifica que sea de la forma $\langle M \rangle \langle w \rangle$. Si es válida, se borra el contenido de la Cinta 2; sino $M_{\mathcal{U}}$ se detiene sin aceptar.
2. Si la entrada es válida, se copia el código de w en la Cinta 2. En esta se simula el procesamiento con M de w.
3. La cadena 1 que representa el estado inicial q_1, se coloca en la Cinta 3. En la Cinta 3 se guarda el estado actual. La unidad de control pasa a escanear el primer símbolo de cada una de las cintas.
4. $M_{\mathcal{U}}$ utiliza la información de las Cintas 2 y 3 para buscar en la Cinta 1 la transición que sea aplicable. Luego, simula en la Cinta 2 lo que haría M y cambia el estado en la Cinta 3. Después de realizar la transición, la unidad regresa al primer símbolo en las Cintas 1 y 3.
5. Se repite el paso anterior hasta que no haya una transición aplicable o se llegue al estado de aceptación.

Lenguaje Universal

- $M_{\mathcal{U}}$ acepta la entrada $\langle M\rangle\langle w\rangle$ si y solamente si M acepta w.
- El lenguaje aceptado por la MT Universal, denominado **lenguaje universal**, es:

$$L_{\mathcal{U}} = \{\langle M\rangle\langle w\rangle : M \text{ acepta } w\}.$$

- Este lenguaje es Turing-aceptable pero no es Turing-decidible.

Lenguaje Aceptado por la MT Universal

Teorema

El lenguaje universal $L_{\mathcal{U}} = \{\langle M \rangle \langle w \rangle : M \text{ acepta } w\}$ no es Turing-decidible.

- Existen lenguajes que pueden ser aceptados por MT específicas pero que en cualquier MT que los acepte habrá cómputos que nunca terminan.
- Los cómputos interminables, o bucles infinitos, son inherenes a la computación y no se pueden eliminar de la Teoría de la Computación.

Bibliografía

1. Rodrigo De Castro Korgi. **Notas de Clase de Introducción a la Teoría de la Computación**. 2023. Contenidos e imágenes de estas notas fueron incluidas en esta presentación.
2. Harry R. Lewis, Christos H. Papadimitriou. **Elements of the Theory of Computation**. Prentice Hall. 1998.
3. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. **Introduction to Automata Theory, Languages and Computation, Third Edition**. Pearson. 2006.