Complejidad Computacional

Santiago Figueira

Departamento de Computación - FCEN - UBA

clase 2

Clase 2

- Funciones computables
- Tiempo de cómputo
- Codificación de máquinas
- Variantes de máquinas

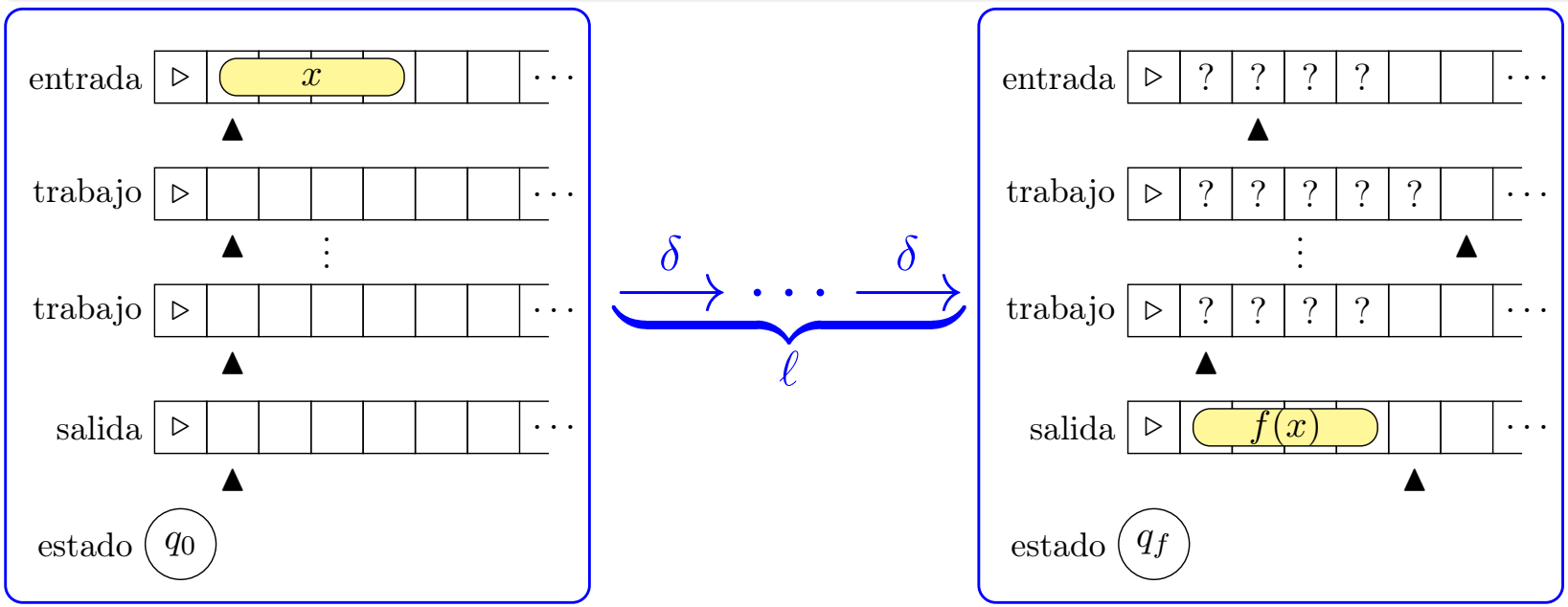
Funciones computables

Función computada por una máquina

Definición

Sean $f : \Gamma^* \to \Gamma^*$ y $M = (\Gamma', Q, \delta)$ una máquina. Decimos que M **computa** f si para todo $x \in \Gamma^*$, hay un cómputo $C_0, \ldots, C_\ell$ de M a partir de x y en C_ℓ la cinta de salida tiene escrito $f(x)$ seguido de blancos.

En este caso, notamos $\boldsymbol{M(x)}$ para representar $f(x)$. Decimos que f es **computable** si existe una máquina que la computa.

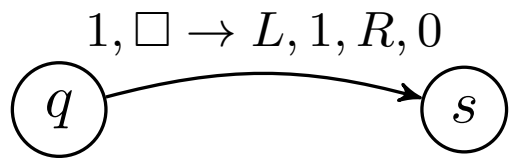


Máquinas representadas como autómatas

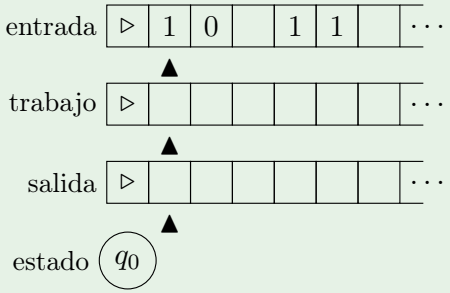
Podemos representar las máquinas como autómatas.

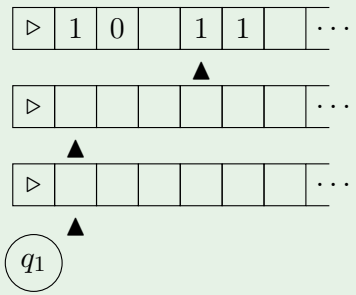
- los nodos del autómata son los estados
- las transiciones representan δ así:

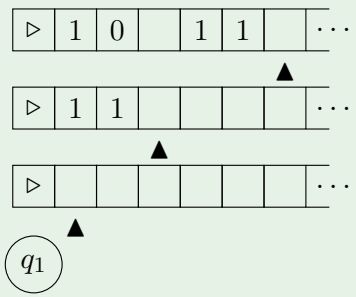
$$\delta(q, 1, \square) = (L, 1, R, 0, s) =$$

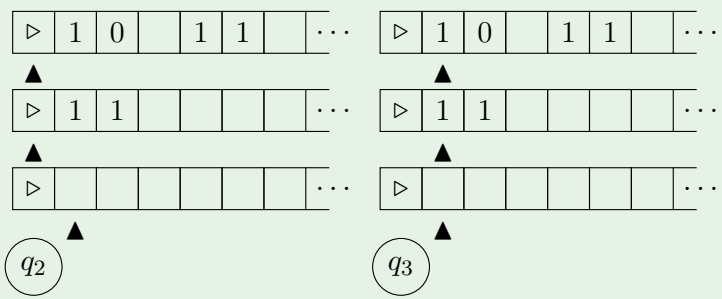


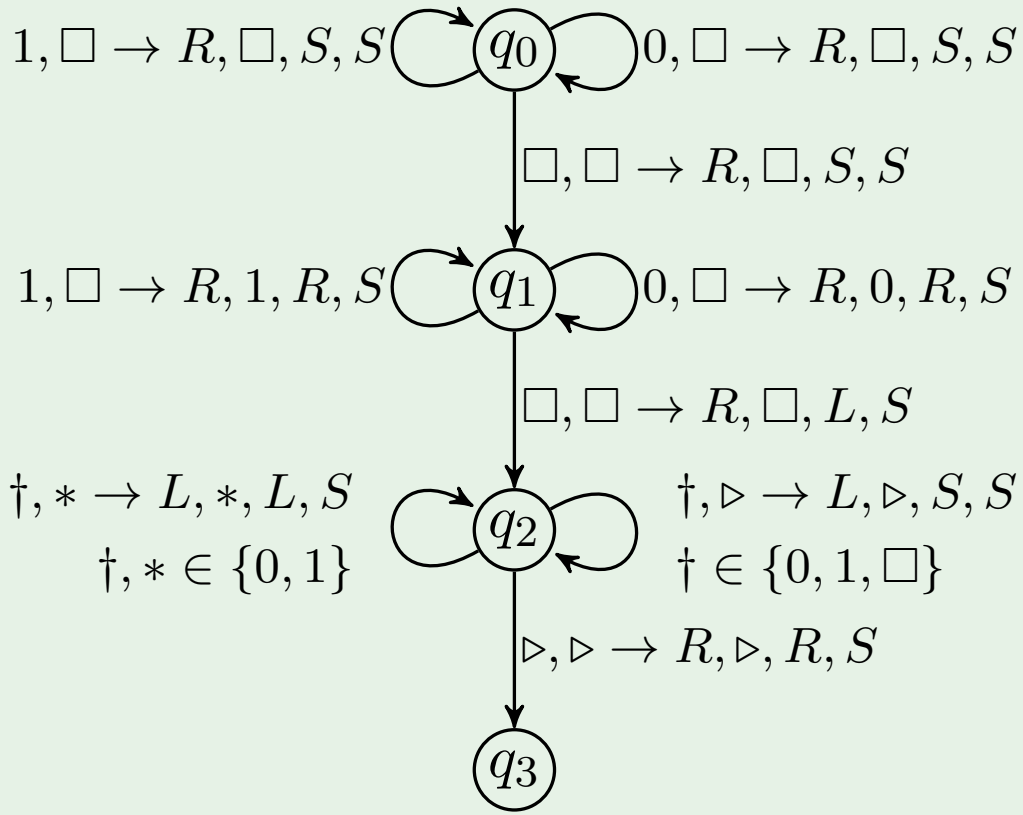
Ejemplo

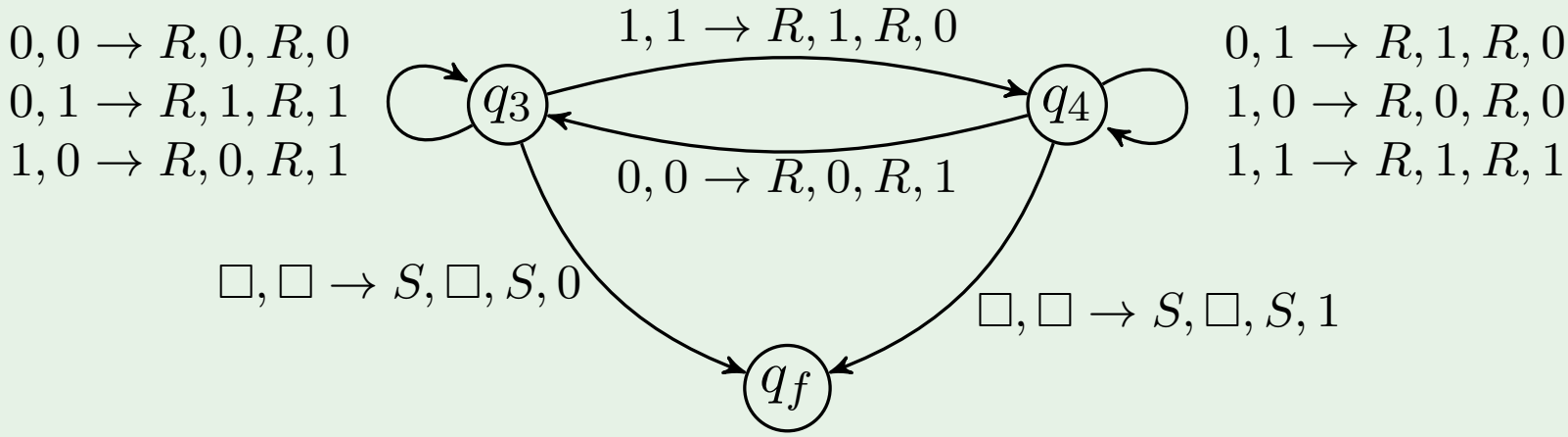

entrada
trabajo
salida
estado
q_0

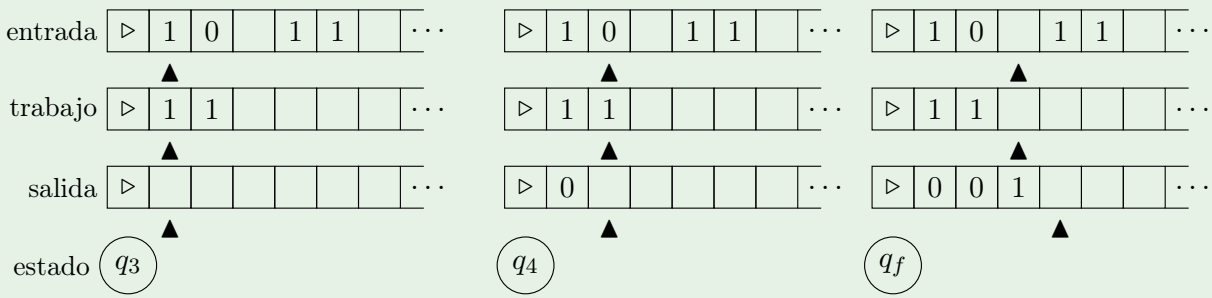

q_1


q_1


q_2
q_3


$1, \square \to R, \square, S, S$
q_0
$0, \square \to R, \square, S, S$
$\square, \square \to R, \square, S, S$
$1, \square \to R, 1, R, S$
q_1
$0, \square \to R, 0, R, S$
$\square, \square \to R, \square, L, S$
$\dagger, * \to L, *, L, S$
$\dagger, * \in \{0, 1\}$
q_2
$\dagger, \triangleright \to L, \triangleright, S, S$
$\dagger \in \{0, 1, \square\}$
$\triangleright, \triangleright \to R, \triangleright, R, S$
q_3





Dados $x, y \in \{0,1\}^n$, con entrada $x\square y$, devuelve $x + y$, si representamos los números en binario con el dígito menos significativo a la izquierda.

Función parcial

Definición

Una **función parcial** es una función que puede indefinirse en algunos puntos. Notamos

- $f(x) \uparrow$ cuando f está indefinida en x
- $f(x) \downarrow$ cuando f está definida en x

Llamamos **dominio de** f al conjunto

$$\mathrm{dom} f = \{x : f(x) \downarrow\}.$$

Notación: función parcial

A las funciones parciales f las vamos a notar $f : \subseteq \Gamma \to \Gamma$ para marcar que el dominio de f no necesariamente es Γ.

En general, como las máquinas pueden **colgarse** (es decir, no terminar), van a computar funciones *parciales*.

Función *parcial* computada por una máquina

Definición

Sean $f : \subseteq \Gamma^* \to \Gamma^*$ una función parcial y $M = (\Gamma', Q, \delta)$ una máquina. Decimos que M **computa** f si para todo $x \in \Gamma^*$:

- si $f(x) \downarrow$ entonces hay un cómputo $C_0, \ldots, C_\ell$ de M a partir de x y en C_ℓ la cinta de salida tiene escrito $f(x)$ seguido de blancos. Decimos que $M(x) \downarrow$.
- si $f(x) \uparrow$ entonces hay una secuencia infinita de configuraciones $C_0, C_1, \ldots$ tal que C_0 es inicial C_{i+1} es la evolución de C_i en un paso y ningún C_i es final
  - es decir no hay cómputo de M a partir de x
  - M 'se cuelga' con entrada x

Decimos que f es **parcial computable** si existe una máquina que la computa.

Funciones *totales* vs funciones *parciales*

- las máquinas pueden 'colgarse' a partir de cierta entrada, es decir no llegar nunca a una configuración final
- en general las máquinas computan funciones *parciales*
- pero en la Teoría de la Complejidad estamos interesados en funciones (estándar), no en funciones *parciales*

Notación: función parcial

- 'función' es un mapeo (estándar) de *toda* entrada a una salida
  - las notamos como siempre: $f : \Gamma^* \to \Gamma^*$
- 'función parcial' es un mapeo posiblemente incompleto
  - también las notamos $f : \subseteq \Gamma^* \to \Gamma^*$ para reforzar que el dominio no necesariamente es Γ^*
- a veces, para reforzar, llamamos 'función total' a las primeras (las estándar)

Infinitas máquinas para la misma función

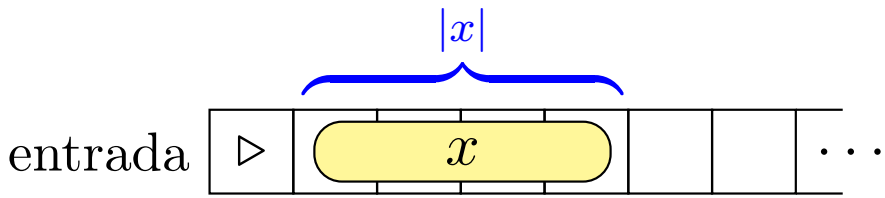
Para una función fija $f : \Gamma^* \to \Gamma^*$ computable, existen infinitas máquinas que la computan

- si existe M tal que computa f, podemos definir otra máquina que agrega estados a los de M y transiciones espurias de modo que no modifique la función que computa

Tamaño de la entrada

Sea $x \in \Gamma^*$.
Vamos a prestar especial atención a la cantidad de celdas que se necesitan para representar x en la cinta de entrada.

Recordar que el **tamaño** de x, notado $|x|$, es la cantidad de símbolos de x.



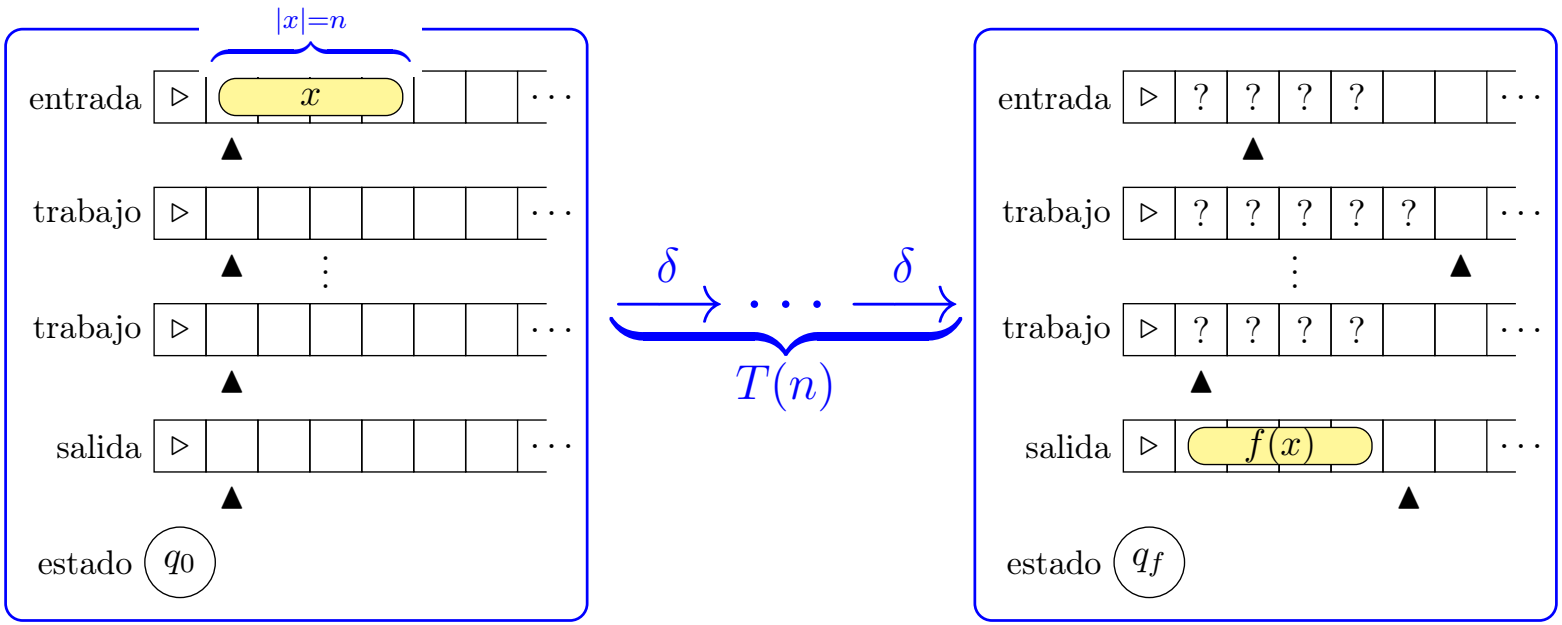
Tiempo de cómputo

Tiempo de cómputo

Definición

Sean $f : \Gamma^* \to \Gamma^*$, $T : \mathbb{N} \to \mathbb{N}$ y $M = (\Gamma', Q, \delta)$ una máquina.

- **M corre en tiempo $T(n)$** si para todo $x \in \Gamma^*$ hay un cómputo de M a partir de x de longitud $\leq T(|x|)$ (en particular, M no se cuelga nunca).
- **M computa f en tiempo $T(n)$** si M corre en tiempo $T(n)$ y M computa f.



Notación de 'O grande'

Definición

Sean $f, g : \mathbb{N} \to \mathbb{N}$. Decimos que $\boldsymbol{f = O(g)}$ si existe c tal que para todo n suficientemente grande tenemos

$$f(n) \leq c \cdot g(n).$$

Notación de 'O grande'

Definición

Sean $f, g : \mathbb{N} \to \mathbb{N}$. Decimos que $\boldsymbol{f = O(g)}$ si existe c tal que para todo n suficientemente grande tenemos

$$f(n) \leq c \cdot g(n).$$

Ejemplos

- $4(n+2) = O(n)$
- $1000000n + 1000000 = O(n)$
- $n \log n = O(n^2)$
- $5n^2 + 3n + 1 = O(n^2)$
- $2^n \neq O(n^k)$ para ningún k

Siempre que sea necesario, usaremos $\log n$ como abreviatura de $\lceil \log n \rceil$.

Cómputo en tiempo $O(T(n))$ y tiempo polinomial

Definición

***M* corre en tiempo $O(T(n))$** si existe una constante c tal que para todo $x \in \Gamma^*$, salvo finitos, hay un cómputo de M a partir de x de longitud a lo sumo $c \cdot T(|x|)$.

Definición

M **corre en tiempo polinomial** si existe un polinomio p tal que M corre en tiempo $O(p)$:

Es equivalente a decir:

existe constante c tal que M corre en tiempo $O(n^c)$

Funciones computables en tiempo $O(T(n))$ / polinomial

Definición

Una función $f : \Gamma^* \to \Gamma^*$ es **computable en tiempo $T(n)$ [en tiempo $O(T(n))$]** si existe una máquina $M = (\Gamma', Q, \delta)$ tal que

- M computa f y
- M corre en tiempo $T(n)$ [en tiempo $O(T(n))$].

Definición

Una función $f : \Gamma^* \to \Gamma^*$ es **computable en tiempo polinomial** si existe una máquina $M = (\Gamma', Q, \delta)$ tal que

- M computa f y
- M corre en tiempo polinomial.

Definición

Decimos que un lenguaje $\mathcal{L}$ es **decidible** en tiempo $T(n)$, $O(T(n))$, polinomial, si $\chi_{\mathcal{L}}$ es computable en tiempo $T(n)$, $O(T)$, polinomial.

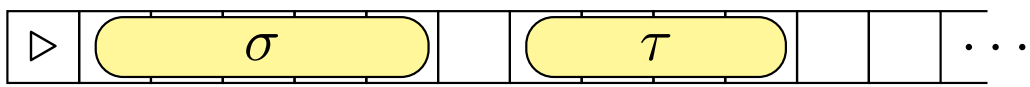
Múltiples parámetros

Buscamos computar funciones de varios parámetros, por ejemplo

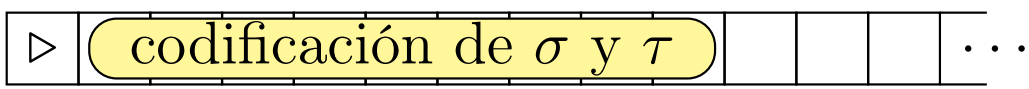
$$f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

No podemos codificar la entrada (σ, τ) como $\sigma\tau$ porque no se entiende dónde termina σ y dónde empieza τ

- podemos usar 3 símbolos: usar los blancos de la cinta de entrada y representar (σ, τ) como $\triangleright\sigma\square\tau\square\square\square\cdots$, o sea



- podemos usar solo 2 símbolos (0 y 1) y usar codificaciones autodelimintantes sobre $\{0,1\}^*$



Elegimos la segunda por comodidad de notación.

Funciones construibles en tiempo

Definición

Una función $T : \mathbb{N} \to \mathbb{N}$ es **construible en tiempo** si

- $T(n) \geq n$
- la función $1^n \mapsto [T(n)]$ es computable en tiempo $O(T(n))$

Funciones construibles en tiempo

Definición

Una función $T : \mathbb{N} \to \mathbb{N}$ es **construible en tiempo** si

- $T(n) \geq n$
- la función $1^n \mapsto [T(n)]$ es computable en tiempo $O(T(n))$

Ejemplos de funciones construibles en tiempo

- $n \log n$
- n
- n^2
- 2^n

Notación: f, T

A partir de ahora

- f siempre va a ser una función $\{0,1\}^* \to \{0,1\}^*$
- $T : \mathbb{N} \to \mathbb{N}$ siempre va a ser una función construible en tiempo

(salvo que se diga lo contrario)

Codificación de máquinas

Codificación de máquinas

Recordar que una máquina M con k cintas ($k \geq 3$) es una tripla (Σ, Q, δ), donde

- Q es un conjunto finito de estados
- Σ es el alfabeto (siempre finito)
- la función de transición es

$$\delta : Q \times \Sigma^{k-1} \to \underbrace{\{L, R, S\}}_{\text{entrada}} \times \underbrace{\Sigma^{k-2} \times \{L, R, S\}^{k-2}}_{\text{trabajo}} \times \underbrace{(\Sigma \cup \{S\})}_{\text{salida}} \times Q$$

Queremos representar máquinas con palabras en $\{0, 1\}^*$.

- nos restringimos al alfabeto estándar $\Sigma = \{0, 1, \triangleright, \square\}$ pero se puede hacer con cualquier alfabeto.

Codificación de máquinas

Codificación de estados, movimientos y símbolos

Numeramos los estados de Q desde 0 hasta $|Q| - 1$ y representamos cada estado n con $[n]$

- reservamos el 0 para q_0
- reservamos el 1 para q_f

Codificamos cada símbolo de $\Sigma \cup \{L, R, S\}$ con

- $[0] = 000$
- $[1] = 001$
- $[\square] = 010$
- $[\triangleright] = 011$
- $[L] = 100$
- $[R] = 101$
- $[S] = 110$

Codificación de máquinas

Codificación de δ

Codificamos una tupla

$$\vec{v} = (E, t_1, \ldots, t_{k-2}, T_1, \ldots, T_{k-2}, Z, q) \in$$

$$\underbrace{\{L, R, S\}}_{\text{entrada}} \times \underbrace{\Sigma^{k-2} \times \{L, R, S\}^{k-2}}_{\text{trabajo}} \times \underbrace{(\Sigma \cup \{S\})}_{\text{salida}} \times Q$$

con

$$\langle \vec{v} \rangle = \langle [E], [t_1], \ldots, [t_{k-2}], [T_1], \ldots, [T_{k-2}], [Z], [q] \rangle \in \{0, 1\}^*$$

Codificación de máquinas

Codificación de δ

Codificamos una tupla

$$\vec{v} = (E, t_1, \ldots, t_{k-2}, T_1, \ldots, T_{k-2}, Z, q) \in$$

$$\underbrace{\{L, R, S\}}_{\text{entrada}} \times \underbrace{\Sigma^{k-2} \times \{L, R, S\}^{k-2}}_{\text{trabajo}} \times \underbrace{(\Sigma \cup \{S\})}_{\text{salida}} \times Q$$

con

$$\langle \vec{v} \rangle = \langle [E], [t_1], \ldots, [t_{k-2}], [T_1], \ldots, [T_{k-2}], [Z], [q] \rangle \in \{0, 1\}^*$$

Codificamos

$$\delta : Q \times \Sigma^{k-1} \to \underbrace{\{L, R, S\}}_{\text{entrada}} \times \underbrace{\Sigma^{k-2} \times \{L, R, S\}^{k-2}}_{\text{trabajo}} \times \underbrace{(\Sigma \cup \{S\})}_{\text{salida}} \times Q$$

con $\langle \delta \rangle =$

$$\langle \{ \langle [q], [a_1], \ldots, [a_{k-1}], \langle \delta(q, a_1, \ldots, a_{k-1}) \rangle \rangle : q \in Q, a_1, \ldots, a_{k-1} \in \Sigma \} \rangle$$
$$\in \{0, 1\}^*$$

Máquinas $\longleftrightarrow$ palabras en binario

Codificamos $M = (\Sigma, Q, \delta)$ con k cintas ($k \geq 3$) con

$$\langle M \rangle = \langle [|Q|], [k], \langle \delta \rangle \rangle \in \{0,1\}^*$$

- toda palabra $x \in \{0,1\}^*$ representa alguna máquina
  - si x no respeta el patrón $\langle M \rangle$ para alguna $M = (\Sigma, Q, \delta)$, entonces suponemos que representa una máquina trivial fija cualquiera (por ejemplo una que termina ni bien empieza y devuelve la palabra vacía)
- identificamos máquinas con palabras $x \in \{0,1\}^*$; hablamos de 'la máquina x' o 'la x-ésima máquina' para referirnos a la única máquina M tal que $\langle M \rangle = x$
- toda máquina representa una función parcial
- para toda función parcial existen infinitas máquinas que la computan
- toda función parcial se codifica con infinitas palabras

Numerables máquinas, numerables funciones computables

- hay una cantidad numerable de máquinas
- hay una cantidad numerable de funciones $\{0,1\}^* \to \{0,1\}$ computables
- hay una cantidad no numerable de funciones $\{0,1\}^* \to \{0,1\}$
- por lo tanto, debe haber funciones $\{0,1\}^* \to \{0,1\}$ no computables
- ¿qué ejemplo concreto conocemos?

Variantes de máquinas

Máquinas sobre alfabetos no estándar

Proposición

Sea Γ un alfabeto. Si f es computable en tiempo $T(n)$ por una máquina $M = (\Gamma, Q, \delta)$, entonces f es computable en tiempo $O(\log |\Gamma| \cdot T(n))$ por una máquina $M' = (\Sigma, Q', \delta')$ donde $\Sigma = \{0, 1, \triangleright, \square\}$.

Máquinas sobre alfabetos no estándar

Proposición

Sea Γ un alfabeto. Si f es computable en tiempo $T(n)$ por una máquina $M = (\Gamma, Q, \delta)$, entonces f es computable en tiempo $O(\log|\Gamma| \cdot T(n))$ por una máquina $M' = (\Sigma, Q', \delta')$ donde $\Sigma = \{0, 1, \triangleright, \square\}$.

Idea de la prueba.

Podemos codificar cada símbolo de Γ en binario usando $\lceil \log|\Gamma| \rceil$ celdas. $\square$

Máquinas de cinta única

Máquinas de cinta única: tienen una sola cinta con una cabeza de lectura y escritura. Para estas máquinas, la función de transición es

$$\delta : Q \times \Sigma \to \Sigma \times \{L, R, S\} \times Q$$

(con restricciones para no pasarse del comienzo de cinta).

Máquinas de cinta única

Máquinas de cinta única: tienen una sola cinta con una cabeza de lectura y escritura. Para estas máquinas, la función de transición es

$$\delta : Q \times \Sigma \to \Sigma \times \{L, R, S\} \times Q$$

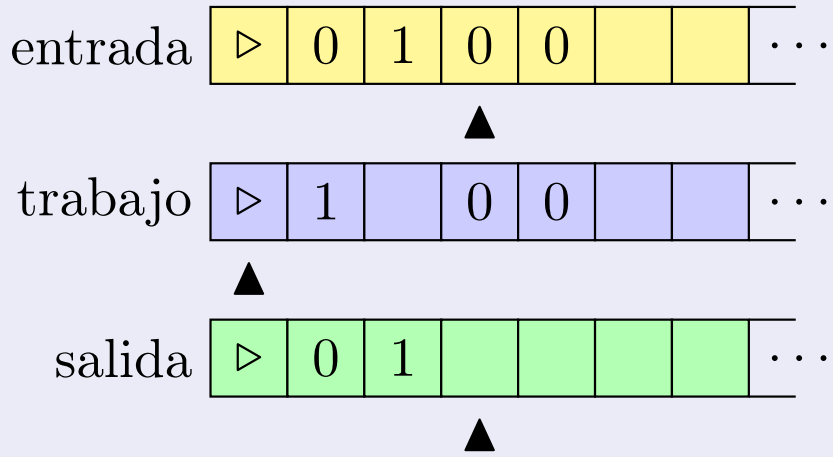
(con restricciones para no pasarse del comienzo de cinta).

Proposición

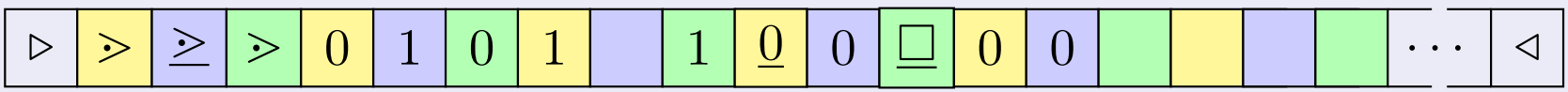
Si f es computable en tiempo $T(n)$ por una máquina estándar de $k \geq 3$ cintas (entrada, salida y $k-2$ cintas de trabajo), entonces f es computable en tiempo $O(T(n)^2)$ por una máquina de cinta única.

Idea de la prueba.

Codificamos las cintas de la máquina M que computa f (alfabeto estándar $\Sigma = \{0, 1, \triangleright, \square\}$)

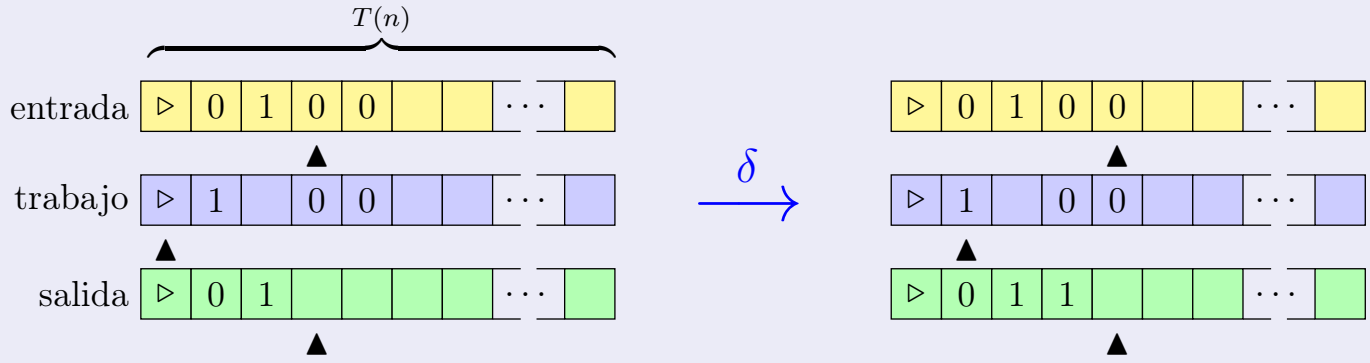


con k cintas en una máquina M' de cinta única con alfabeto $\Gamma = \{\triangleright, 0, 1, \square, \underline{\triangleright}, \underline{0}, \underline{1}, \underline{\square}, \triangleleft\}$.

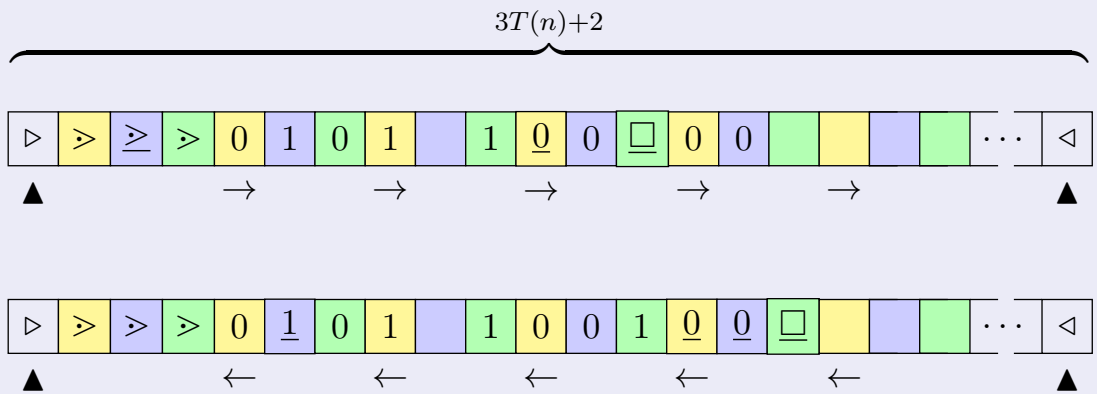


Símbolo subrayado indica la posición de la cabeza.
M recorre en tiempo $T(n)$, entonces a lo sumo usa $T(n)$ celdas de cada una de sus cintas.
M' codifica toda la información en las primeras $k \cdot T(n)$ celdas. No usa más que eso.

Cada vez que M hace esto:



M' barre de izquierda a derecha y de derecha a izquierda.



M' con entrada x hace esto:

- Obtiene $|x| = n$ y calcula $T(n)$. Mueve la cabeza hasta la posición $k \cdot T(n) + 1$ y escribe $\triangleleft$. Esto le va a servir para marcar la parte de la cinta única que va a usar. Luego vuelve a la primera posición.
- Codifica la configuración inicial de M con entrada x en su cinta (esto incluye a x) usando exactamente $k \cdot T(n)$ celdas.
- Maneja el estado de M internamente, sin usar la cinta
- Repite lo siguiente hasta que M llegue a el estado final:
  - Barre la cinta de izquierda a derecha obteniendo los símbolos leídos por cada cabeza (subrayados)
  - Dependiendo de qué hace la función de transición de M con esa información, elige qué hacer con cada cabeza.
  - Barre la cinta de derecha a izquierda volcando esa nueva información (cambia a lo sumo 1 símbolo por cada cinta, las posiciones de las cabezas y el estado).
- Borra todo lo que hay en su cinta salvo la información de la salida de $M(x)$

M' corre en tiempo $O(T(n)^2)$ y calcula f. Luego simular M' con otra máquina sobre el alfabeto estándar. $\square$

Máquinas *oblivious*

Oblivious = que no se da cuenta, ajena, inconsciente

Definición

Una máquina M es ***oblivious*** si para cada entrada x y para cada $i \in \mathbb{N}$,

- la posición de las cabezas de las cintas de entrada y trabajo en el i-ésimo paso del cómputo de M con entrada x solo depende de i y de $|x|$ (pero no de x), y
- las funciones que computan esas posiciones a partir de $i, |x|$ son computables en tiempo polinomial.

Máquinas *oblivious*

Oblivious = que no se da cuenta, ajena, inconsciente

Definición

Una máquina M es ***oblivious*** si para cada entrada x y para cada $i \in \mathbb{N}$,

- la posición de las cabezas de las cintas de entrada y trabajo en el i-ésimo paso del cómputo de M con entrada x solo depende de i y de $|x|$ (pero no de x), y
- las funciones que computan esas posiciones a partir de $i, |x|$ son computables en tiempo polinomial.

Proposición

Si f es computable en tiempo $T(n)$ por una máquina estándar entonces hay una máquina *oblivious* que computa f en tiempo $O(T(n)^2)$.

Demostración.

La idea de la prueba anterior: M' es *oblivious*. □

Máquinas con cintas bi-infinitas

Máquinas con cintas bi-infinitas: tienen cintas infinitas en ambas direcciones en lugar de ser infinitas a derecha.

Proposición

Si f es computable por una máquina con cintas bi-infinitas en tiempo $T(n)$, entonces f es computable por una máquina estándar en tiempo $O(T(n))$.

Máquinas con cintas bi-infinitas

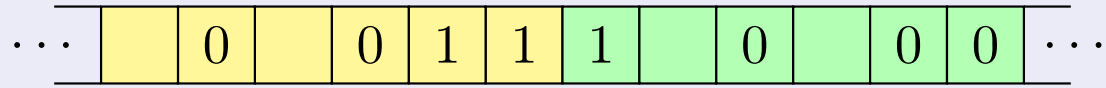
Máquinas con cintas bi-infinitas: tienen cintas infinitas en ambas direcciones en lugar de ser infinitas a derecha.

Proposición

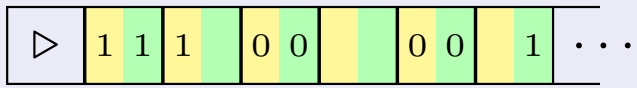
Si f es computable por una máquina con cintas bi-infinitas en tiempo $T(n)$, entonces f es computable por una máquina estándar en tiempo $O(T(n))$.

Idea de la prueba.

Podemos 'plegar' cada cinta: convertir



en



sobre un alfabeto $\{\triangleright\} \cup \{0, 1, \square\}^2$. Según qué porción de la cinta bi-infinita esté procesando, usar primer o segundo símbolo de $00, 01, 10, 11$. Luego, traducir al alfabeto estándar. □