Teoría de la Computación

Clase 23: Máquina de Turing Universal

Mauro Artigiani

11 Octubre 2021

Universidad del Rosario, Bogotá

Un poco de historia

ENIAC

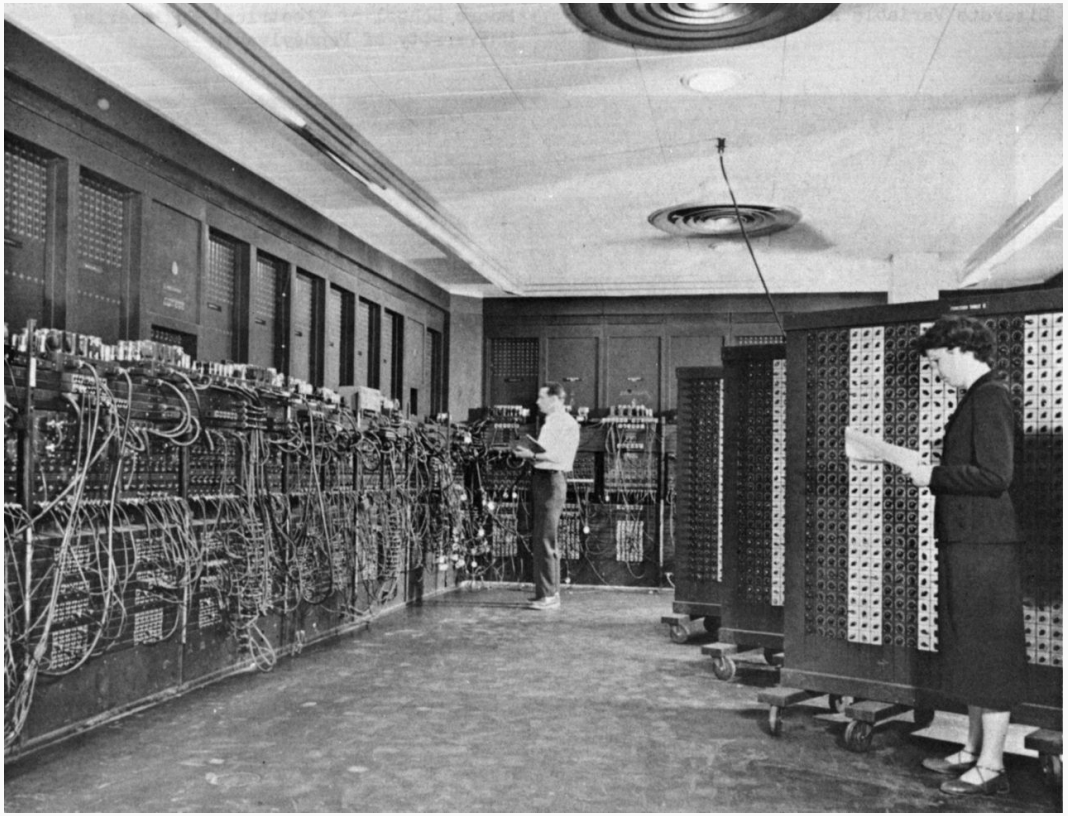
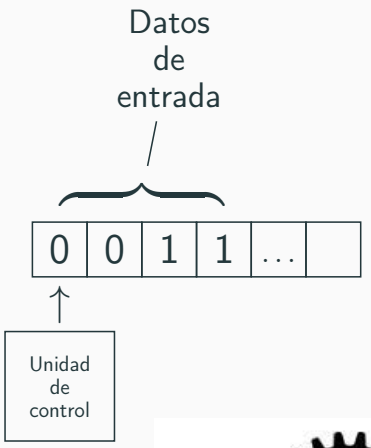


Foto: De Desconocido - U.S. Army Photo, Dominio público, https://commons.wikimedia.org/w/index.php?curid=55124

Hacia la máquina universal

Hay dos tipos de computación:

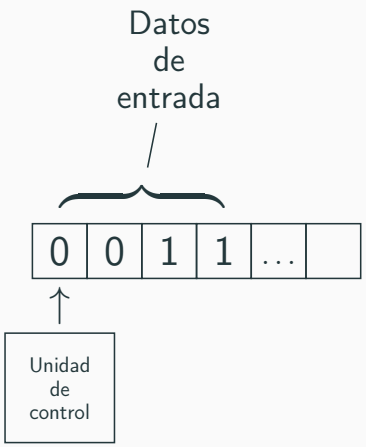
Program-controlled y *Stored-program*

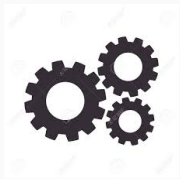


Hacia la máquina universal

Hay dos tipos de computación:

Program-controlled y *Stored-program*



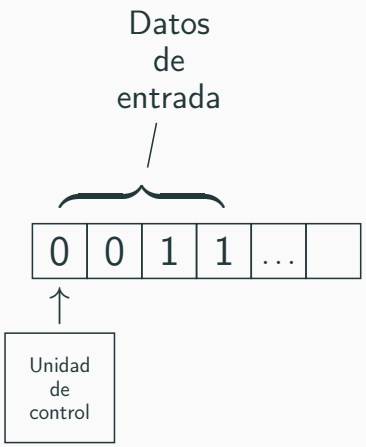


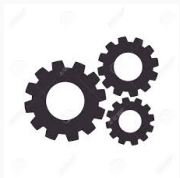
Para cambiar el programa se necesita cambiar la máquina

Hacia la máquina universal

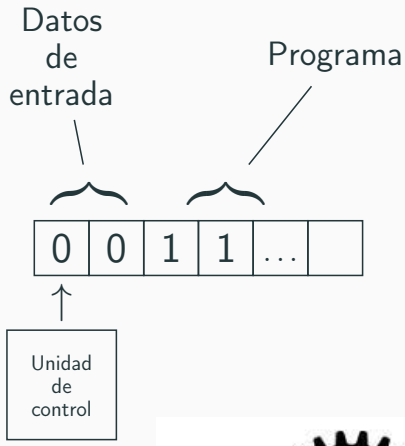
Hay dos tipos de computación:

Program-controlled y *Stored-program*





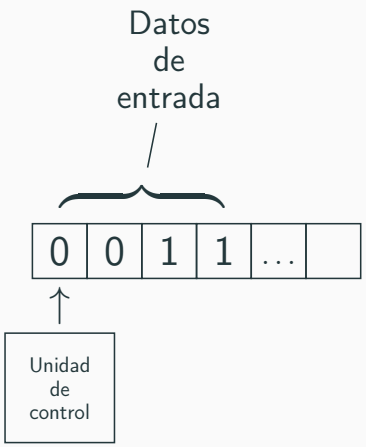
Para cambiar el programa se necesita cambiar la máquina

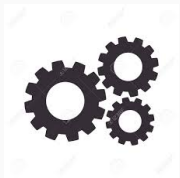


Hacia la máquina universal

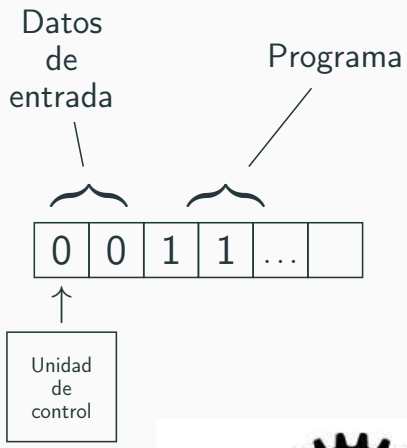
Hay dos tipos de computación:

Program-controlled y *Stored-program*





Para cambiar el programa se necesita cambiar la máquina



Para cambiar el programa se necesita cambiar el input

Influencia de Turing

- Cambridge, 1936: On computable numbers;

Influencia de Turing

- Cambridge, 1936: On computable numbers;
- Princeton, 1936–1938: Trabajó con John von Neumann;

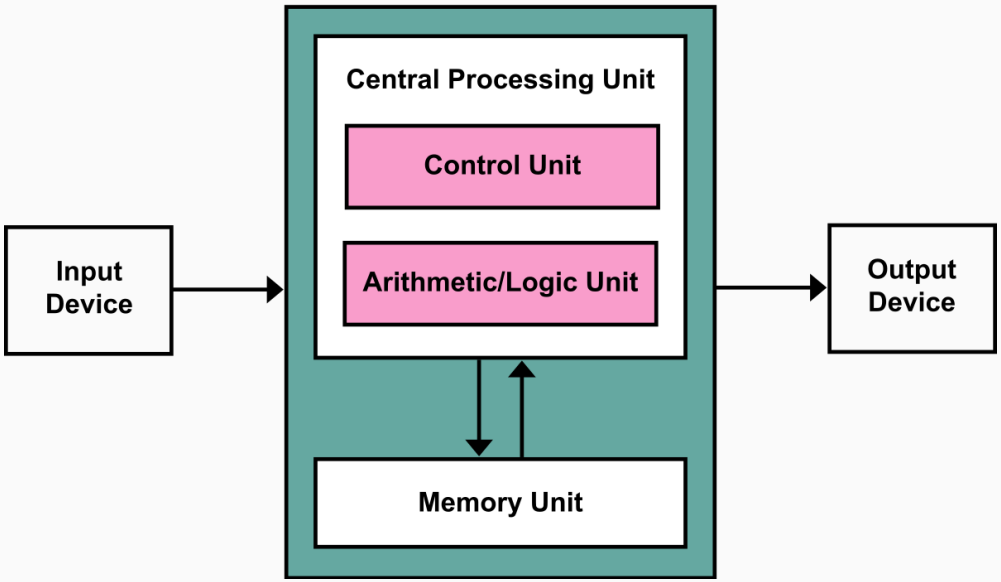
Influencia de Turing

- Cambridge, 1936: On computable numbers;
- Princeton, 1936–1938: Trabajó con John von Neumann;
- Londres, 1945–1948: Diseñó el Automatic Computing Engine (ACE);

Influencia de Turing

- Cambridge, 1936: On computable numbers;
- Princeton, 1936–1938: Trabajó con John von Neumann;
- Londres, 1945–1948: Diseñó el Automatic Computing Engine (ACE);
- Manchester, 1948–1954: Director del Computing Machinery Laboratory

Influencia sobre John von Neumann



El diseño de la arquitectura de von Neumann está inspirada en las ideas de Turing y de sus máquinas.

Universidad de Manchester

- Primer computador *stored-program*;
- Primera computador de punto flotante;
- Primer computador de transistores;
- Primer computador con memoria virtual.

Mark 1

Máquina universal

Una máquina de Turing universal

Queremos construir una máquina de Turing U que se universal, en el sentido que pueda *simular* cualquiera máquina de Turing M.

Una máquina de Turing universal

Queremos construir una máquina de Turing U que se universal, en el sentido que pueda *simular* cualquiera máquina de Turing M.

Más precisamente, queremos codificar de alguna manera la `TM` M en la cinta de U.

Una máquina de Turing universal

Queremos construir una máquina de Turing U que se universal, en el sentido que pueda *simular* cualquiera máquina de Turing M.

Más precisamente, queremos codificar de alguna manera la `TM` M en la cinta de U. Dando un input w a U esta máquina debería utilizar la codificación de M para calcular en su cinta como si fuera la máquina M misma.

Una máquina de Turing universal

Queremos construir una máquina de Turing U que se universal, en el sentido que pueda *simular* cualquiera máquina de Turing M.

Más precisamente, queremos codificar de alguna manera la `TM` M en la cinta de U. Dando un input w a U esta máquina debería utilizar la codificación de M para calcular en su cinta como si fuera la máquina M misma.

Una buena analogía es pensar en U como si fuera el hardware de nuestro computador y M como si fuera un programa memorizado en la memoria de U.

Codificación unaria

Una manera compacta de codificar una máquina de Turing cualquiera es utilizar una codificación unaria.

Codificación unaria

Una manera compacta de codificar una máquina de Turing cualquiera es utilizar una codificación unaria.

Empezamos codificando los alfabetos Σ y Γ.

Codificación unaria

Una manera compacta de codificar una máquina de Turing cualquiera es utilizar una codificación unaria.

Empezamos codificando los alfabetos Σ y Γ.

$$\Gamma = \{s_1, s_2, \ldots, s_m, \ldots, s_p\}$$

$$\Sigma = \{s_2, \ldots, s_m\}$$

$s_{m+1}, \ldots, s_p$ son los símbolos auxiliares utilizados por M

Símbolo	Codificación
s_1 (símbolo ƀ)	1
s_2	11
s_3	111
$\vdots$	$\vdots$
s_m	$\underbrace{11\cdots1}_{m \text{ veces}}$
$\vdots$	$\vdots$
s_p	$\underbrace{11\cdots1}_{p \text{ veces}}$

El símbolo ƀ es el símbolo que utiliza De Castro en lugar de $\sqcup$.

Codificación unaria

Seguimos ahora codificando los estados Q de la máquina de Turing y los movimientos de la unidad de control.

Codificación unaria

Seguimos ahora codificando los estados Q de la máquina de Turing y los movimientos de la unidad de control.

Asumamos, por simplicidad, que la `TM` tenga un único estado inicial q_1 y un único estado de aceptación q_2.

Codificación unaria

Seguimos ahora codificando los estados Q de la máquina de Turing y los movimientos de la unidad de control.

Asumamos, por simplicidad, que la `TM` tenga un único estado inicial q_1 y un único estado de aceptación q_2.

Los estados de una MT, $q_1, q_2, q_3, \ldots, q_n$, se codifican también con secuencias de unos:

Estado	Codificación
q_1 (inicial)	1
q_2 (final)	11
$\vdots$	$\vdots$
q_n	$\underbrace{11\cdots1}_{n \text{ veces}}$

Las directrices de desplazamiento $\rightarrow$, $\leftarrow$ y $-$ se codifican con 1, 11 y 111,

Codificación unaria

Veamos un ejemplo de como codificar la función de transición δ.

Codificación unaria

Veamos un ejemplo de como codificar la función de transición δ.

Sean: $Q = \{q_1, q_2, \ldots, q_n\}$; $\Gamma = \{\text{b}, a, b\}$

Codificación unaria

Veamos un ejemplo de como codificar la función de transición δ.

Sean: $Q = \{q_1, q_2, \ldots, q_n\}$; $\Gamma = \{\text{ƀ}, a, b\}$

Utilizamos los de antes para codificar cada transición, separando cada bloque de 1s por 0.

$$\delta(q_1, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, a) = (q_3, a, \rightarrow)$$
$$\delta(q_3, b) = (q_4, b, \rightarrow)$$
$$\delta(q_4, \text{ƀ}) = (q_2, \text{ƀ}, -)$$

$$0101101110110100111011011101101001110111011110111010011110101101011 10$$

$$0101^2 01^3 01^2 01001^3 01^2 01^3 01^2 01001^3 01^3 01^4 01^3 01001^4 0101^2 0101^3 0.$$

Codificación binaria

Veamos ahora otra posible codificación, utilizando esta vez una codifica binaria.

Codificación binaria

Veamos ahora otra posible codificación, utilizando esta vez una codifica binaria.

La codificación binaria permite hacer una implementación más parsimoniosa y concisa.

Codificación binaria

Veamos ahora otra posible codificación, utilizando esta vez una codifica binaria.

La codificación binaria permite hacer una implementación más parsimoniosa y concisa. Como veremos pero, esta codificación no es exactamente universal.

Codificación binaria

Veamos ahora otra posible codificación, utilizando esta vez una codifica binaria.

La codificación binaria permite hacer una implementación más parsimoniosa y concisa. Como veremos pero, esta codificación no es exactamente universal.

Si Q (o Γ) tiene m símbolos y $m \leq 2^l$, es posible hacer su codificación mediante l-bits. Por ejemplo, si $Q = \{q_0, q_1, q_2, q_3\}$, entonces podemos codificar q_0 como 00, q_1 como 01, etc.

Estructura de la UTM

La cinta de nuestra máquina de Turing universal (o UTM) se ve así.

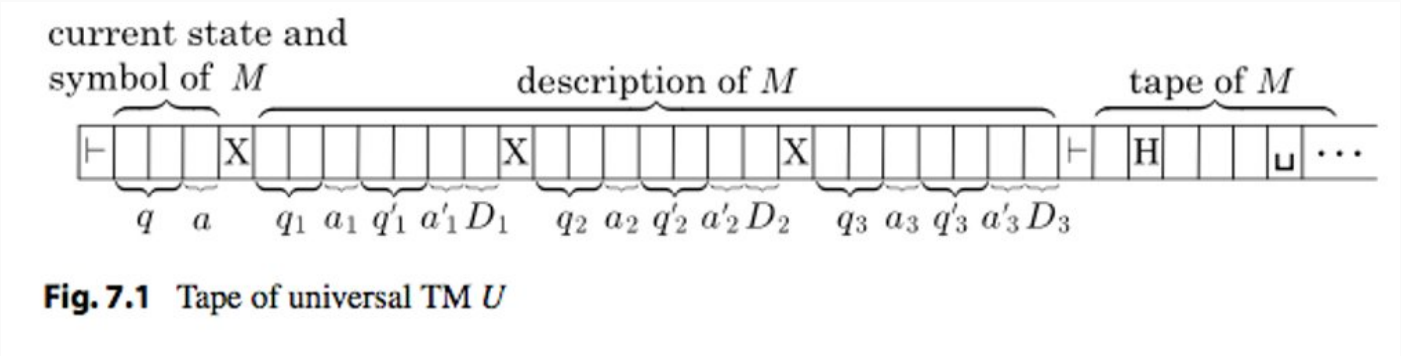


Fig. 7.1 Tape of universal TM U

De izquierda a derecha:

1. El estado actual (en codifica binaria) y el símbolo bajo la unidad de control de M;

Estructura de la UTM

La cinta de nuestra máquina de Turing universal (o UTM) se ve así.

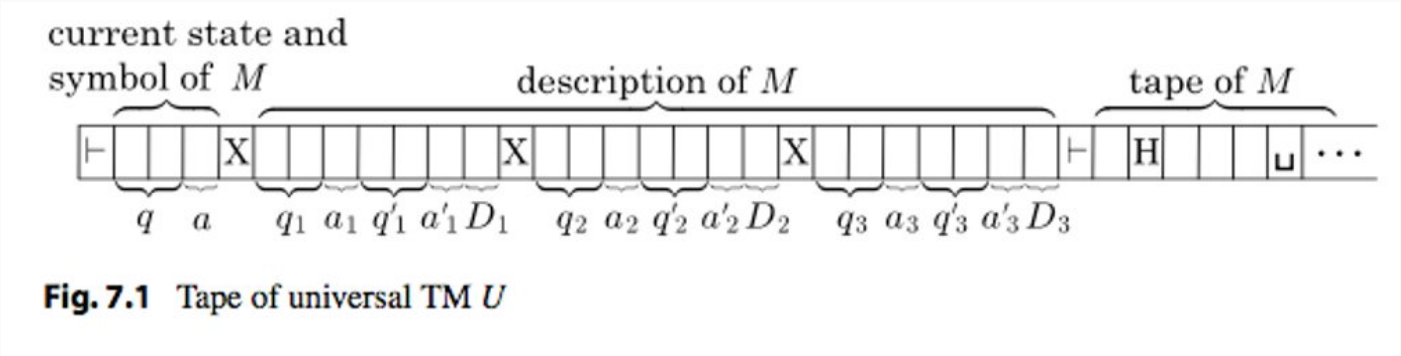


Fig. 7.1 Tape of universal TM U

De izquierda a derecha:

1. El estado actual (en codifica binaria) y el símbolo bajo la unidad de control de M;
2. La codificación de la máquina M en la forma (q, a, q', a', D) si $\delta(q, a) = (q', a', D)$; si $D = L$ escribimos 0, si $D = R$ escribimos 1;

Estructura de la UTM

La cinta de nuestra máquina de Turing universal (o UTM) se ve así.

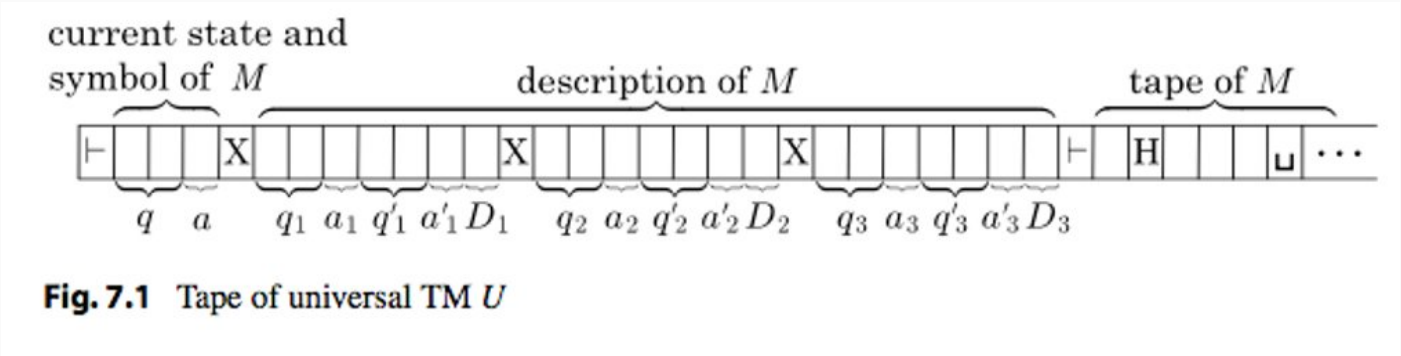


Fig. 7.1 Tape of universal TM U

De izquierda a derecha:

1. El estado actual (en codifica binaria) y el símbolo bajo la unidad de control de M;
2. La codificación de la máquina M en la forma (q, a, q', a', D) si $\delta(q, a) = (q', a', D)$; si $D = L$ escribimos 0, si $D = R$ escribimos 1;
3. La cinta de M, menos la posición actual, remplazada por H.

Como U simula M

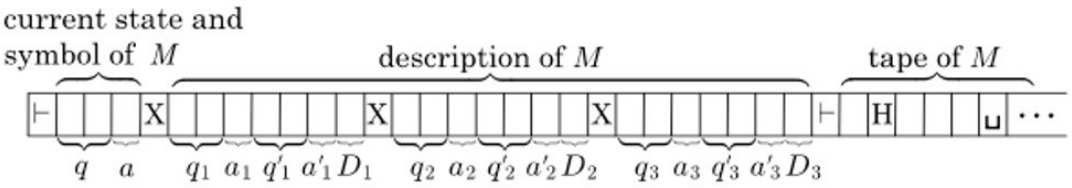


Fig. 7.1 Tape of universal TM U

1. Buscar una 5-tupla (q, a, q', a', D) en la región de descripción de máquina tal que (q, a) coincida con la información guardada en la región de condición de máquina.

Como U simula M

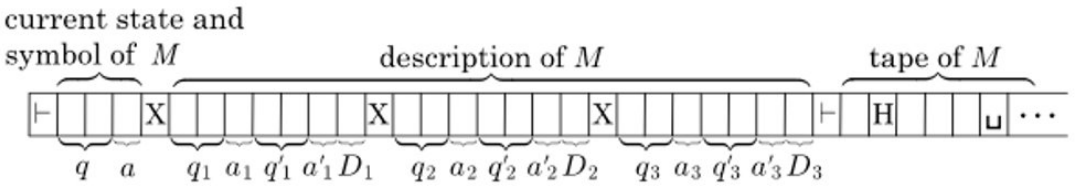


Fig. 7.1 Tape of universal TM U

1. Buscar una 5-tupla (q, a, q', a', D) en la región de descripción de máquina tal que (q, a) coincida con la información guardada en la región de condición de máquina.
2. Escribir (q', a') en la región de condición de máquina encontrada en el paso 1. Adicionalmente, "memorizar como estado" la dirección D encontrada en la 5-tupla.

Como U simula M

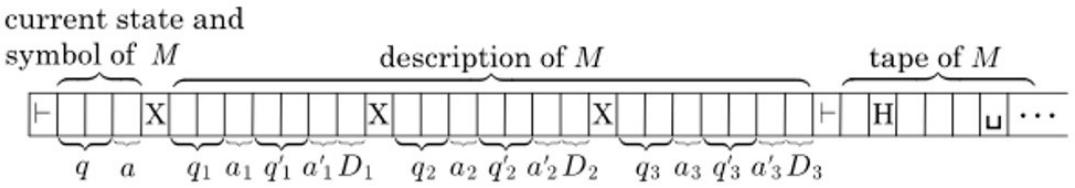


Fig. 7.1 Tape of universal TM U

3. Escribir el símbolo S en lugar del a' puesto en la región de condición de máquina en el paso 2 y "memorizar como estado" el símbolo a'. Moverse a la región de cinta y cambiar H por a', dirigir la unidad de control en la dirección D.

Como U simula M

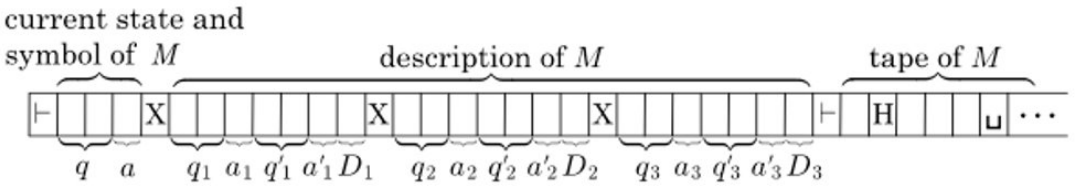


Fig. 7.1 Tape of universal TM U

3. Escribir el símbolo S en lugar del a' puesto en la región de condición de máquina en el paso 2 y "memorizar como estado" el símbolo a'. Moverse a la región de cinta y cambiar H por a', dirigir la unidad de control en la dirección D.
4. "Memorizar como estado" el símbolo a'' que se encuentra debajo de la unidad de control y en su lugar poner H. Ir a la izquierda hasta encontrar el símbolo S y cambiarlo por a''. Ir al paso 1.

Resumen

Resumen

Hoy aprendimos:

- La diferencia entre máquinas *program-controlled* y *stored-program*;
- La codificación unaria y binaria de una `TM`.
- Las ideas clave para la implementación de una `TM` universal.