Máquinas de Turing

Horst H. von Brand
vonbrand@inf.utfsm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Contenido

Modelo formal de computación

El modelo «computación es lo que puede hacer un programa C» es bien poco satisfactorio: La descripción formal de C (el estándar ISO/IEC 9899:2018 de 2018) son unas 535 páginas. Ese estándar deja ciertas áreas explícitamente sin definir. Y ya que estamos en eso, ¿por qué no usar Scheme (un lenguaje definitivamente mucho más simple)? ¿O uno popular, como Python o Java?

Se requiere un modelo más simple, susceptible de demostraciones. Y ojalá menos relacionado a la computación como la hacemos hoy. Al fin y al cabo, nos interesa computación como tema abstracto, independiente de algún esquema específico.

Reconstruyendo el razonamiento de Turing

Consideremos una persona, lápiz en mano, efectuando cálculos en papel. Suponemos que sigue al pie de la letra sus instrucciones, todas muy simples (nos interesa dejar fuera posibles efectos de creatividad o inventiva).

Para precisar, sea el papel cuadriculado, con cada casillero un número finito de posibilidades. Para simplificar aún más, suponemos papel de una dimensión (una cinta). El computante solo mira el casillero actual, borra y sobreescribe su contenido, se mueve un casillero a la izquierda o derecha, y va a un nuevo paso en su lista de instrucciones.

En términos de nuestra maquinaria de autómatas, formalizamos lo anterior como *máquina de Turing* (abreviada TM, por el inglés *Turing machine*).

Máquina de Turing

Definición

Una *máquina de Turing* $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ consta de:

- $\boldsymbol{Q}$: Conjunto finito de *estados*.
- $\boldsymbol{\Sigma}$: *Alfabeto de entrada.*
- $\boldsymbol{\Gamma}$: *Alfabeto de la cinta*, $\Sigma \subseteq \Gamma$.
- $\boldsymbol{\delta}$: *Función de transición*, $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$, donde L es mover a la izquierda y R hacia la derecha.
- $\boldsymbol{q_0}$: *Estado inicial*, $q_0 \in Q$.
- $\boldsymbol{B}$: *Blanco* $B \in \Gamma$, $B \notin \Sigma$.
- $\boldsymbol{F}$: *Estados finales* $F \subseteq Q$, y para $q \in F$ es $\delta(q, a) = \varnothing$.

Operación de la máquina de Turing

Este tipo de autómata puede moverse en ambas direcciones en la cinta, no está limitado a solo avanzar como los modelos anteriores. No solo lee los datos, puede modificar. Puede salir del área de la entrada.

Se detiene por ir a un estado final (en cuyo caso acepta), por caer en otra situación en la cual no tiene movidas posibles (con lo que rechaza) o por intentar salir de la cinta moviéndose antes del inicio (con lo que también rechaza).

Variantes de máquina de Turing

Algunas definiciones alternativas limitan al autómata a no escribir nunca B, de forma que se pueda detectar el área no leída/escrita. Otras variantes consideran estados finales como cualquier otro, a ser considerado solo cuando el autómata se detenga (por quedarse sin movidas posibles). Hay casi tantas variantes como textos en el área, lo cual en general no es relevante (es sencillo demostrar que las variantes son equivalentes). Asegúrese de revisar las definiciones relevantes en caso de duda.

Lenguaje aceptado por la máquina de Turing

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una máquina de Turing. Sin pérdida de generalidad, podemos suponer que $Q \cap \Gamma = \varnothing$. Una *descripción instantánea* de M (abreviado ID) es una palabra:

$$\alpha q \beta$$

con $\alpha \in \Gamma^*$, $\beta \in \Gamma^+$ y $q \in Q$.

La idea de la descripción instantánea $\alpha q \beta$ es que M tiene α al lado izquierdo del cabezal, está viendo el primer símbolo de β (por eso exigimos que $\beta \neq \varepsilon$, si fuera necesario agregamos B al final luego del estado).

Lenguaje aceptado por la máquina de Turing

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una máquina de Turing. Definimos la *relación de transición* de M por los siguientes casos, donde $\alpha, \gamma \in \Gamma^*$; $\beta \in \Gamma^+$; $q, p \in Q$; $U, X, Y \in \Gamma$:

$$\begin{aligned} \alpha q X \beta &\vdash_M \alpha Y p \beta && \text{si } (p, Y, R) \in \delta(q, X) \\ \alpha q X &\vdash_M \alpha Y p B && \text{si } (p, Y, R) \in \delta(q, X) \\ \alpha U q X \gamma &\vdash_M \alpha p U Y \gamma && \text{si } (p, Y, L) \in \delta(q, X) \end{aligned}$$

Los primeros dos casos cubren movimiento a la derecha (el segundo agrega un B si M pasa del final actual de la cinta), el tercero cubre una movida a la izquierda. No está permitido moverse a la izquierda del principio de la cinta.

Lenguaje aceptado por la máquina de Turing

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una máquina de Turing. El *lenguaje aceptado* por M se define mediante:

$$\mathscr{L}(M) = \{\sigma \in \Sigma^* : q_0\sigma \vdash_M^* \alpha q_f \beta \wedge q_f \in F\}$$

Otros usos

Alternativamente, podemos considerar que una máquina de Turing determinista (a lo más una movida posible en cada situación) define una función: argumento es el contenido inicial de la cinta, valor el contenido de la cinta cuando se detiene. Se les llama *funciones parciales*, no necesariamente definidas para todos los argumentos (la máquina de Turing puede que nunca se detenga y así no entregue un valor). En contraste, a las funciones como las definimos les llaman *funciones totales*.

Otros usos

Si equipamos a la máquina de Turing con una cinta de salida (solo se permite escribir al final y avanzar en ella), y designamos un símbolo # como separador, podemos considerar que genera un lenguaje (las palabras separadas por # en la cinta de salida).

Tesis de Church

Han habido múltiples intentos adicionales de definir *computación*, basándose en ideas diferentes:

Funciones mutuamente recursivas: El *cálculo lambda*, que llevó a LISP (conocido por ustedes por el dialecto Scheme).

Reemplazo de patrones: Los *algoritmos de Markov*. Ideas similares se usan en la configuración de `sendmail`.

Computadores: Modelos como RAM (*Random Access Memory*) o RASP (*Random Access Stored Program*).

Han habido otros intentos, usando otras ideas, de definir «computación», todas resultaron equivalentes.

La tesis de Church asegura que toda función computable es computada por una máquina de Turing.

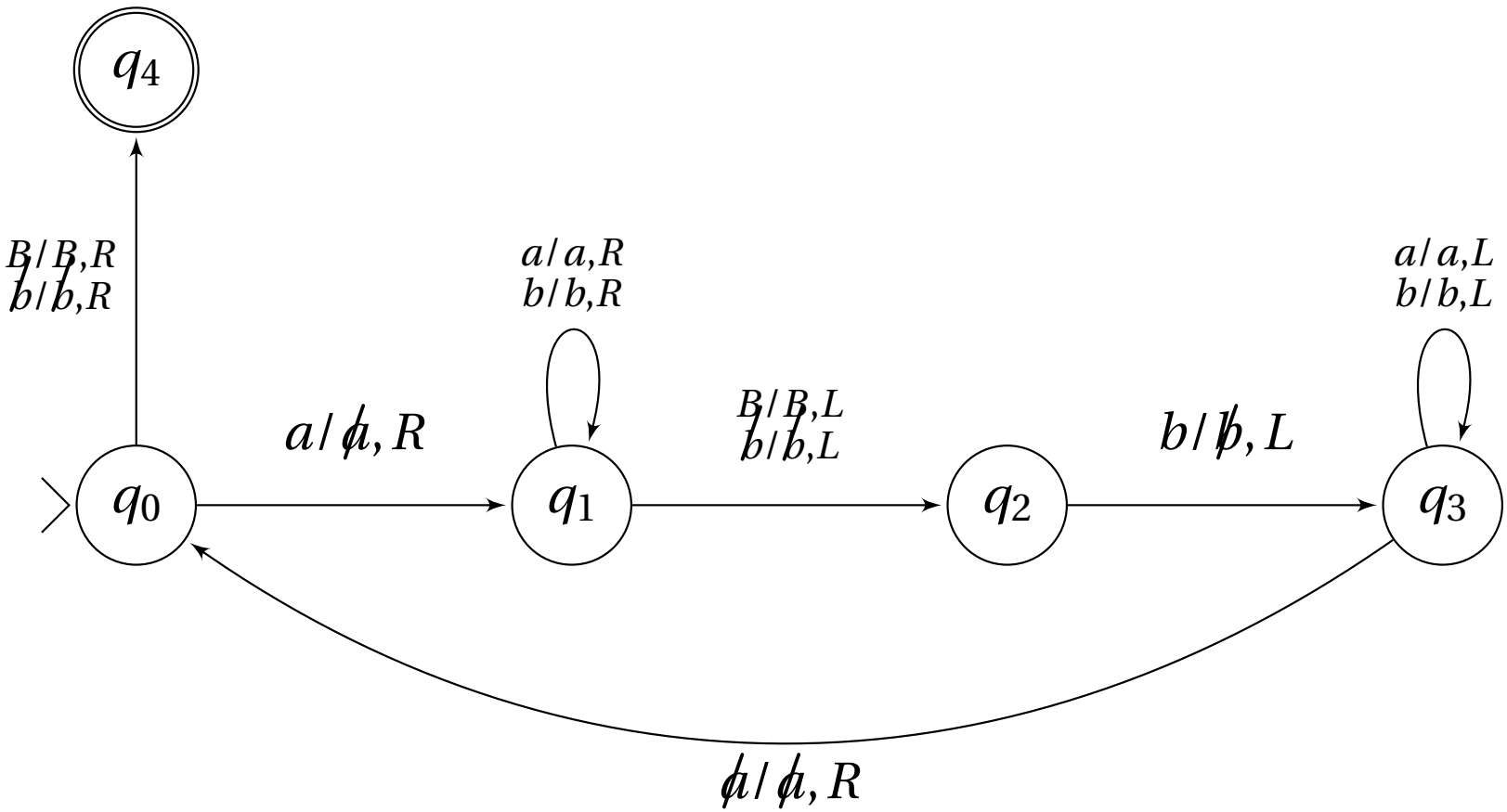
Programar máquinas de Turing

Diseñar una máquina de Turing es una tarea ardua. Incluso algo tan simple como reconocer $\{a^n b^n : n \geq 0\}$ es bastante lioso:

- Idea básica es tarjar una a inicial, avanzar a la última b (antes de B o $\not{b}$) y tarjarla; devolverse y repetir hasta que se acaben.
- Después del ciclo anterior, deben quedar solo letras tarjadas.
- Hay que verificar que la entrada está en $\mathscr{L}(a^* b^*)$.

Usaremos $\Sigma = \{a, b\}$, $\Gamma = \{a, b, \not{a}, \not{b}, B\}$. Notación gráfica adaptada de las usadas antes.

Reconocer $\{a^n b^n : n \geq 0\}$



Reconocer $\{a^n b^n : n \geq 0\}$

Este autómata tarja la primera a (transición q_0 a q_1). Avanza hasta el primer B o $\not b$, sin importar lo que halla en el camino (estado q_1), retrocede (transición q_1 a q_2). Tarja la última b (transición de q_2 a q_3). Retrocede pasando sobre a o b (estado q_3), ubica la última $\not a$ (transición q_3 a q_0), comienza el próximo ciclo.

Este autómata tarja a solo al inicio o luego de $\not a$, tarja b solo al final o justo antes de $\not b$; solo si la cinta está vacía (primer símbolo es B) o si encuentra $\not b$ justo después de $\not a$ (se acabaron las a y las b) va al estado final q_4.

Algunas tareas básicas

Las construcciones que haremos luego requerirán alguna de las siguientes piezas:

Insertar un símbolo: Sobreescribir el símbolo actual con el símbolo a insertar, marcado para poder volver a él, guardar el actual en el estado, ir a la derecha; guardar el símbolo en el estado y sobreescribir con el guardado, ir a la derecha. Al llegar a B (fin de la entrada), volver al símbolo marcado y borrar la marca.

Eliminar un símbolo: Idea similar a la anterior.

Reemplazar una secuencia por otra: Combinar las anteriores adecuadamente.

Extensiones de máquinas de Turing

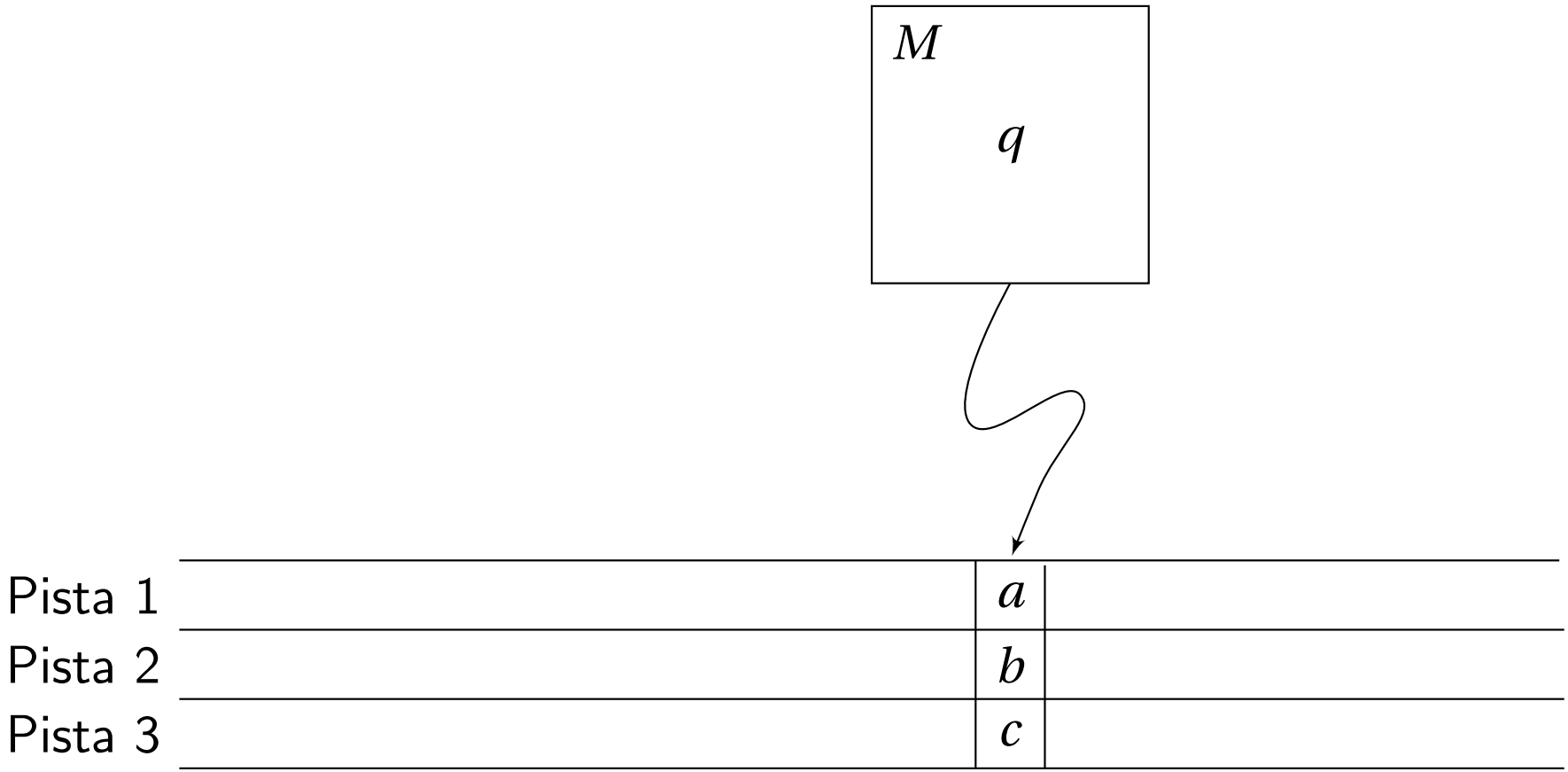
Extenderemos el modelo básico en varios pasos, obteniendo un modelo mucho más fácil de programar. Demostraremos en cada paso que el modelo así extendido puede simularse con el anterior, no agrega nuevas capacidades.

Múltiples pistas

Podemos imaginar cada símbolo de cinta como un vector de símbolos, uno por pista. Usos posibles son poner marcas en las pistas adicionales que permitan por ejemplo volver a un punto indicado.

Para simular este modelo con una máquina de Turing común, considere que Γ es un conjunto de objetos $[a_1, a_2, \ldots, a_n]$, donde $x \in \Sigma$ y $a_1, \ldots, a_n$ son los símbolos en las pistas.

Múltiples pistas



Múltiples cabezales

Podemos considerar múltiples cabezales que trabajan sobre la misma cinta, cada uno de los cuales se mueve independientemente. (Claro que tendremos que acordar qué ocurre si varios cabezales intentan escribir símbolos distintos en una posición dada.)

Lo simulamos mediante una pista extra por cada cabezal, en la que marcamos la posición del cabezal respectivo. Una movida se simula partiendo del cabezal más a la izquierda, en una pasada recogemos los símbolos que cada cabezal ve. En una segunda pasada escribimos los nuevos símbolos bajo cada cabezal y movemos las marcas que representan los cabezales según indica la movida elegida.

Múltiples cintas

Consideremos ahora una máquina de Turing que opera sobre varias cintas independientes.

Simular esto mediante una máquina de Turing con múltiples pistas significa asignar dos pistas por cinta: una pista del par es el contenido de la cinta, la otra contiene una única marca que indica la posición actual del cabezal en esa cinta.

RAM (Random Access Machine)

Buscamos simular un lenguaje «de máquina» (formalmente a esta clase de modelo se le conoce como RAM, *Random Access Machine*, ya que permite acceder a ubicaciones arbitrarias de la memoria). Las instrucciones se ejecutan en secuencia desde la primera salvo que se indique lo contrario.

RAM (Random Access Machine)

Registros: Hay un número fijo n de registros, $R_0, R_1, \ldots, R_n$ que no especificaremos. Cada registro contiene un número natural o cero, inicialmente cero.

Memoria: Un número ilimitado de casilleros, numerados desde cero, cada uno de los cuales puede contener un número natural o cero. Al comenzar el procesamiento los primeros casilleros contienen los datos de entrada, los demás contienen cero.

RAM (Random Access Machine)

Instrucciones: Alguna de las siguientes:

$$R_i \leftarrow \textit{constante}$$

$$R_i \leftarrow \mathrm{mem}[R_j]$$

$$R_i \leftarrow R_j$$

$$\mathrm{mem}[R_j] \leftarrow R_i$$

$$R_i \leftarrow \ominus R_j$$

$$R_i \leftarrow R_i \oplus R_j$$

$$\textbf{goto } \textit{posición}$$

$$\textbf{if } R_i = 0 \textbf{ goto } \textit{posición}$$

$$\textbf{halt}$$

La instrucción **halt** detiene la máquina, el contenido de la posición de memoria 0 es el resultado.

Simulación de RAM

Podemos simular RAM mediante una máquina de Turing:

- Una cinta por cada registro.
- Memoria es una cinta que contiene pares $(\text{dirección}, \text{contenido})$. Al acceder, revise si el casillero existe y actualice el contenido o inicialice con cero.
- Las operaciones se pueden efectuar en cintas adicionales.

RASP (Random Access, Stored Program)

Un modelo afín se conoce como RASP (del inglés *Random Access Stored Program*), la idea es similar pero se codifican las operaciones como enteros y éstos se almacenan en memoria, desde donde se ejecutan. Note que esto permite escribir programas que manipulan programas, e incluso que se modifican a sí mismos.

La simulación mediante una máquina de Turing es una extensión obvia de lo anterior.

Relación con los modelos anteriores

Un NFA no es más que una máquina de Turing que solo se mueve hacia la derecha o deja fijo el cabezal (para movidas ε, lo que puede simularse con una movida a la derecha y volver). Da lo mismo con qué sobreescribe lo leído, jamás vuelve a verlo. Si el estado es final para el NFA, en la máquina de Turing vamos de él al estado final con B.

Un PDA es una máquina de Turing de dos cintas, la segunda cinta es la pila. En la cinta de entrada solo se mueve hacia la derecha o queda fija, la pila se administra como tal.

Máquinas de Turing deterministas y no-deterministas

Una máquina de Turing es determinista si siempre es $|\delta(q, a)| \leq 1$ (a lo más una movida posible).

Teorema

Si L es aceptado por una máquina de Turing, es aceptado por una máquina de Turing determinista.

Máquinas de Turing deterministas y no-deterministas

Demostración.

La idea es ir generando sistemáticamente las descripciones instantáneas de M conforme procesa σ. Para ello supondremos que $Q \cap \Gamma = \varnothing$, y agregaremos una nueva marca de separación #. Iremos escribiendo descripciones instantáneas separadas por #, marcamos $\overset{\checkmark}{\#}$ el final de la última descripción instantánea ya procesada.

Máquinas de Turing deterministas y no-deterministas

Una ronda de procesamiento hace lo siguiente, suponiendo ubicados inicialmente en el primer separador no marcado:

- Avanza hasta el estado, copiando símbolos al final de la cinta.
- Registra el estado y el símbolo en el estado de la máquina que simula.
- Copia al final de la cinta el resultado de la siguiente movida para esa situación.
- Copia el resto de esa descripción instantánea, incluyendo la separación, al final de la cinta.
- Vuelve al primer separador sin marcar, y repite para las demás movidas posibles.
- Al acabar las movidas, vuelve al primer separador sin marcar, lo marca y avanza al siguiente separador sin marcar.

Máquinas de Turing deterministas y no-deterministas

Si como parte de la simulación debe escribir un estado final, se detiene y acepta.
Lo que hace es generar el árbol de las posibles descripciones instantáneas a lo ancho, y acepta en cuanto halla una descripción instantánea de aceptación.

Es claro que la máquina de Turing esbozada es determinista, y acepta el mismo lenguaje que la no-determinista que simula. □

Codificar palabras y máquinas de Turing

Podemos representar los símbolos de cualquier alfabeto en binario (símbolos 0 y 1) usando $\lceil \log_2 |\Gamma| \rceil$ bits para cada uno. Modificar una máquina de Turing para que use la codificación binaria en vez de la representación original es sencillo.

Codificar palabras y máquinas de Turing

Podemos representar la máquina de Turing $M = (Q, \{0,1\}, \{0,1,B\}, \delta, q_0, F)$ siguiendo lo siguiente:

- El conjunto de estados es $Q = \{1, \ldots, N\}$, estado inicial es $q_0 = 1$, único estado final es $q_f = 2$. (Los nombres de los estados no importan. En nuestra variante, no hay movidas desde estados finales.)
- Representamos δ como un conjunto de quintetos: si $(p, b, d) \in \delta(q, a)$, agregamos el quinteto (q, a, p, b, d) (acá q y p son estados, a, b son símbolos de cinta, d es un dirección de movimiento).

Codificar palabras y máquinas de Turing

Simplemente listando los quintetos queda descrita la máquina de Turing: los alfabetos están fijos, estado inicial y final son implícitos.

Claro que a una máquina de Turing corresponden muchísimas descripciones: los nombres de los estados no son fijos, y podemos listar los quintetos en cualquier orden. Pero esto no importa en lo que sigue.

Codificar palabras y máquinas de Turing

Un quinteto (q, a, p, b, d) que representa una posible movida de una máquina de Turing podemos representarlo codificando:

Estados: El estado número q por 0^q.

Símbolos: 0 por 0, 1 por 00, B por 000.

Direcciones: L es 0, R es 00.

Las partes del quinteto se separan por 1. Listamos los quintetos, separados por 11. Iniciamos y cerramos la descripción con 111.

Con esto tenemos una codificación binaria de máquinas de Turing. Para dar una representación como máquina de Turing a todas las palabras binarias, diremos que si no corresponde el formato es la máquina de Turing sin movidas (acepta $\varnothing$).

Máquina de Turing universal

Tenemos una manera de representar una máquina de Turing como una palabra binaria. Diseñaremos una máquina de Turing que interprete esa descripción sobre datos dados. Es *universal*, en que es capaz de hacer lo que cualquier máquina de Turing hace.

Máquina de Turing universal

La máquina de Turing universal, cariñosamente llamada U, parte en su cinta con una descripción de una máquina de Turing como definida antes (que llamaremos M en lo que sigue, su descripción como palabra la llamaremos $\langle M \rangle$), seguida inmediatamente por los datos del caso. Usaremos varias cintas, cuyo uso iremos describiendo conforme veamos la necesidad para ellas.

Máquina de Turing universal

Cintas y su uso

Cinta de trabajo: La cinta de M que estamos simulando.

Cinta original: La usaremos para hacer referencia a $\langle M \rangle$.

Cinta de estado: Si M está en el estado q, contendrá 10^q1 y posiblemente más símbolos (sobrantes de operaciones anteriores).

Contador: Para copiar la entrada a una nueva cinta de trabajo, requerimos poder volver al comienzo. Tomamos una cinta en la cual escribimos 1 y luego agregamos un 0 para cada símbolo copiado de la entrada. Retrocediendo luego en las cintas de trabajo, la de entrada y el contador podemos parar al llegar al inicio de la entrada y el final de $\langle M \rangle$. Podemos reusarla como cinta de estado luego.

Máquina de Turing universal

Operación

Antes que nada, verificar que la entrada realmente comienza con una descripción de una máquina de Turing. Si no, se detiene sin aceptar.
Para comenzar, luego de copiar la entrada a la cinta de trabajo y posicionarse en su inicio, copia 101 (estado 1, inicial) a la cinta de estado. Retrocedemos al comienzo de la descripción de la máquina a simular.

Máquina de Turing universal

Operación

En un ciclo:

- Usando no determinismo, ubica el comienzo de un quinteto en $\langle M \rangle$.
- Verifica que el estado del quinteto coincide con el estado simulado y que el símbolo leído es el correcto.
- Copia el nuevo estado al comienzo de la cinta de estado, se posiciona a su comienzo. Si el estado es 2 (final), se detiene y acepta.
- Sobreescribe el símbolo leído en la cinta de trabajo con el nuevo símbolo indicado en el quinteto elegido.
- Mueve el cabezal de la cinta de trabajo como indica el quinteto elegido.
- Mueve el cabezal al comienzo de la descripción de la máquina simulada.

Máquina de Turing universal

Operación

Si M se detiene por quedarse sin movidas o caerse de la cinta, lo mismo ocurre con la simulación.

Resumen

- Definimos el modelo de máquina de Turing, esbozamos varios usos.
- Discutimos la tesis de Church.
- Definimos modelos más cómodos de diseñar que el modelo básico.
- Codificamos máquinas de Turing, y esbozamos la construcción de una máquina de Turing universal.
- Planteamos modelos RAM y RASP, cercanos a un procesador actual.