Capítulo 8

Máquinas de Turing

8.1 Introducción

Hasta ahora hemos visto clases de lenguajes relativamente simples. Lo que vamos a ver ahora es preguntarnos qué lenguajes pueden definirse por cualquier equipo computacional.

Vamos a ver qué pueden hacer las computadoras y los problemas que no pueden resolver, a los que llamaremos *indecidibles*.

Por ejemplo, podemos pensar en un programa sencillo de computadora que imprima "hola". De igual forma, podemos pensar en otro programa que imprima "hola" cuando encuentre un entero positivo $n > 2$, que cumpla: $x^n + y^n = z^n$, para x, y y z enteros positivos.

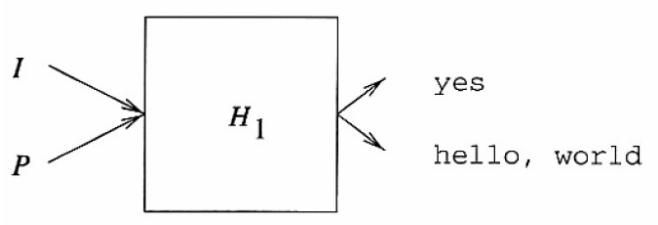
La solución entera de la ecuación de arriba se conoce como el último teorema de Fermat, que llevo a los matemáticos 300 años resolver.

El poder analizar cualquier programa de computadora y decidir si va a imprimir un letrero como "hola" es en general indecidible.

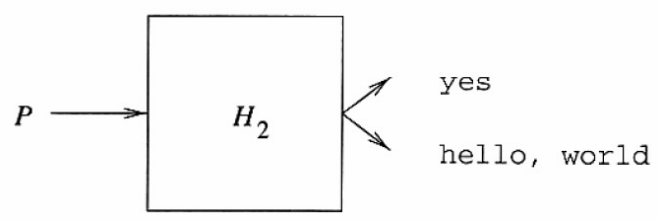
La idea de la prueba es relativamente simple. Necesitamos tener un programa H que toma de entrada otro programa P y una entrada a ese programa I y regresa "si" o "no" dependiendo de si el programa P imprime "hola" o

no.

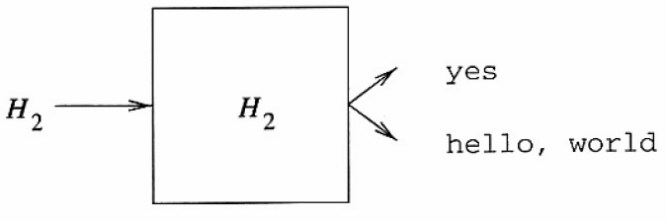
Podemos pensar igualmente en un programa $H1$ que imprima "si" cuando el programa P imprima "hola" e imprima "hola" cuando no.



Ahora podemos pensar en otro programa $H2$ que toma solo de entrada a P e imprime "si" cuando P imprime "hola" e imprime "hola" cuando P no imprime "hola".



Ahora si le damos $H2$ como entrada a $H2$ llegamos a una contradicción.



Muchas veces para probar si un problema es indecidible, se transforma a otra del cual ya se sabe que es indecidible.

Por ejemplo, si queremos probar que un programa va a llamar a una función "foo" es o no es indecidible. La idea es diseñar un programa que con cierta entrada llame a la función "foo" cuando otro programa imprima "hola".

8.2 Máquina de Turing

El propósito de la teoría de indecibilidad no es sólo establecer cuales problemas son indecidibles, sino también dar una guía sobre qué es lo que se puede hacer o no con programación.

También tiene que ver con problemas, que aunque decidibles, son intratables.

A finales del s. XIX y principios del s. XX, D. Hilbert lanzó la pregunta abierta, si era posible encontrar un algoritmo que determinara el valor de verdad de una fórmula en lógica de primer orden aplicada a los enteros.

En 1931, K. Gödel probó su teorema de incompletes usando un argumento parecido al de H2 que vimos arriba, para probar que no se podía construir dicho algoritmo.

En 1936, A. Turing publicó su máquina de Turing como un modelo para cualquier tipo de computación (aunque todavía no existían las computadoras). La hipótesis de Church o la tésis de Church-Turing dice que lo que las máquinas de Turing (y para tal caso las computadoras modernas) pueden computar son las funciones recursivamente enumerables.

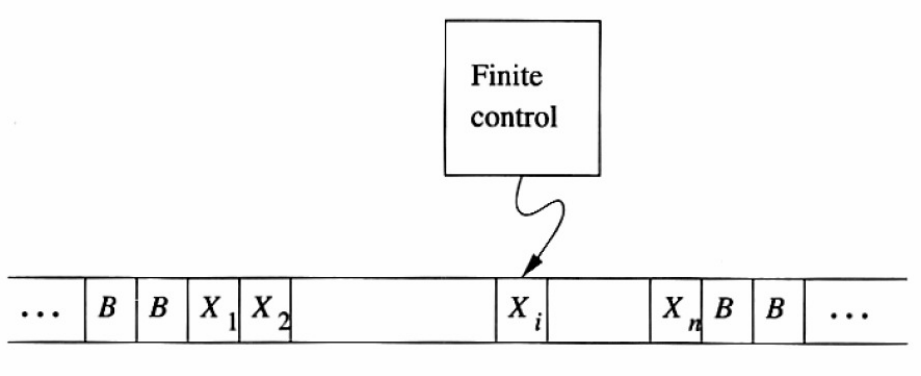
Una máquina de Turing consiste de un control finito que puede estar en cualquier estado de un conjunto finito de estados.

Se tiene una cinta dividida en celdas, cada celda con un símbolo. Inicialmente, la entrada (cadena finita de símbolos del alfabeto) se coloca en la cinta, el resto de las celdas tienen el símbolo especial vacío.

La cabeza de la cinta está siempre sobre una celda y al principio está sobre la celda más a la izquierda con el primer símbolo de la cadena de entrada.

Un movimiento o transición puede cambiar de estado (o quedarse en el estado actual), escribir un símbolo (reemplazando el símbolo que existía o

dejando el mismo) y mover la cabeza a la izquierda o derecha.



Formalmente, una máquina de Turing es una séptupla: $M = (Q, \Sigma, \Gamma, \delta, q0, B, F)$, donde:

- Q: es un conjunto finito de estados
- Σ: es un conjunto finito de símbolos de entrada
- Γ: es el conjunto de símbolos de la cinta. Σ es siempre un subconjunto de Γ
- δ: la función de transición $\delta(q, X) = (p, Y, D)$, donde p es el siguiente estado en Q, Y es el símbolo en Γ que se escribe en la celda que está viendo la cabeza de la cinta y D es la dirección (izq. o der.).
- q_0: es el estado inicial
- B: es el símbolo de vacío, que esta en Γ pero no en Σ
- F: es el conjunto de estados finales o de aceptación.

8.2.1 Descripciones instantáneas o IDs para las máquinas de Turing

Como la cinta es infinita, se representan sólo los símbolos entre los B's (a veces se pueden incluir algunos B's) y se incluye un símbolo especial para

indicar la posición de la cabeza. Por ejemplo: $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n$ representa in ID donde:

q es el estado de la máquina de Turing.

La cabeza de la cinta está viendo el i-ésimo símbolo a la izquierda $X_1X_2 \ldots X_n$ es el pedazo de cinta entre los símbolos más a la izquierda y más a la derecha que no son vacíos.

Usamos la misma notación de ID que para los PDAs: $\vdash$ y $\overset{*}{\vdash}$.

Supongamos que $\delta(q, X_i) = (p, Y, L)$, el siguiente movimiento es a la izquierda (L). Entonces:

$X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n \vdash X_1X_2 \ldots X_{i-2}pX_{i-1}YX_{i+1} \ldots X_n$

Excepciones:

1. Si $i = 1$ entonces M se mueve al B a la izquierda de $X_1 : qX_1X_2 \ldots X_n \vdash pBYX_2 \ldots X_n$
2. Si $i = n$ y $Y = B$, entonces el símbolo que se re-escribe sobre X_n se une a la cadena infinita de B's y no se escribe: $X_1X_2 \ldots X_{n-1}qX_n \vdash X_1X_2 \ldots X_{n-2}pX_{n-1}$

Ahora, supongamos que $\delta(q, X_i) = (p, Y, R)$, movimiento hacia la derecha (R): $X_1X_2 \ldots X_{i-1}qX_iX_{i+1} \ldots X_n \vdash X_1X_2 \ldots X_{i-1}YpX_{i+1} \ldots X_n$

Excepciones:

1. Si $i = n$ entonces la $i + 1$ celda tiene un B que no era parte de la ID anterior: $X_1X_2 \ldots X_{n-1}qX_n \vdash X_1X_2 \ldots X_{n-1}YpB$
2. Si $i = 1$ y $Y = B$, entonces el símbolo que se re-escribe sobre X_1 se une a la cadena infinita de B's y no se escribe: $qX_1X_2 \ldots X_n \vdash pX_2 \ldots X_n$

Ejemplo: una TM que acepta: $\{0^n1^n | n \geq 1\}$

Nos queda lo siguiente: $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$

Con la siguiente tabla de transición:

	Símbolo				
Estado	0	1	X	Y	B
q_0	(q_1, X, R)	$-$	$-$	(q_3, Y, R)	$-$
q_1	$(q_1, 0, R)$	(q_2, Y, L)	$-$	(q_1, Y, R)	$-$
q_2	$(q_2, 0, L)$	$-$	(q_0, X, R)	(q_2, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	(q_4, B, R)
q_4	$-$	$-$	$-$	$-$	$-$

Por ejemplo, si le damos la entrada 0011 sigue las siguientes transiciones:

$q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash X q_0 0 Y 1 \vdash X X q_1 Y 1 \vdash X X Y q_1 1 \vdash X X q_2 Y Y \vdash X q_2 X Y Y \vdash X X q_0 Y Y \vdash X X Y q_3 Y \vdash X X Y Y q_3 B \vdash X X Y Y B q_4 B$

Mientras que para la cadena 0010, tenemos lo siguiente:

$q_0 0010 \vdash X q_1 010 \vdash X 0 q_1 10 \vdash X q_2 0 Y 0 \vdash q_2 X 0 Y 0 \vdash X q_0 0 Y 0 \vdash X X q_1 Y 0 \vdash X X Y q_1 0 \vdash X X Y 0 q_1 B$

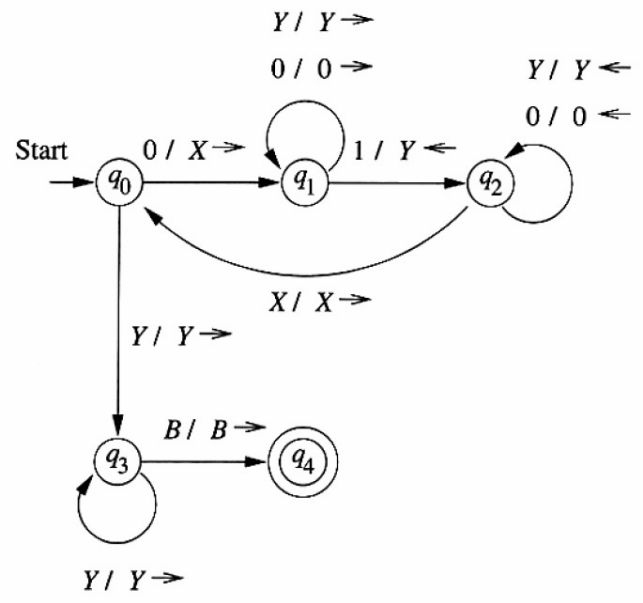
8.2.2 Diagramas de transición para TMs

En un diagrama de transición para TMs los nodos son los estados y los arcos tienen etiquetas de la forma X/YD donde X y Y son símbolos de la cinta y D es la dirección.

Para cada $\delta(q, X) = (p, Y, D)$, tenemos un arco con etiqueta: X/YD que va del nodo q al nodo p.

Lo único que falta es el símbolo vacío que asumimos que es B.

Por ejemplo, el diagrama de transición para la TM anterior es:

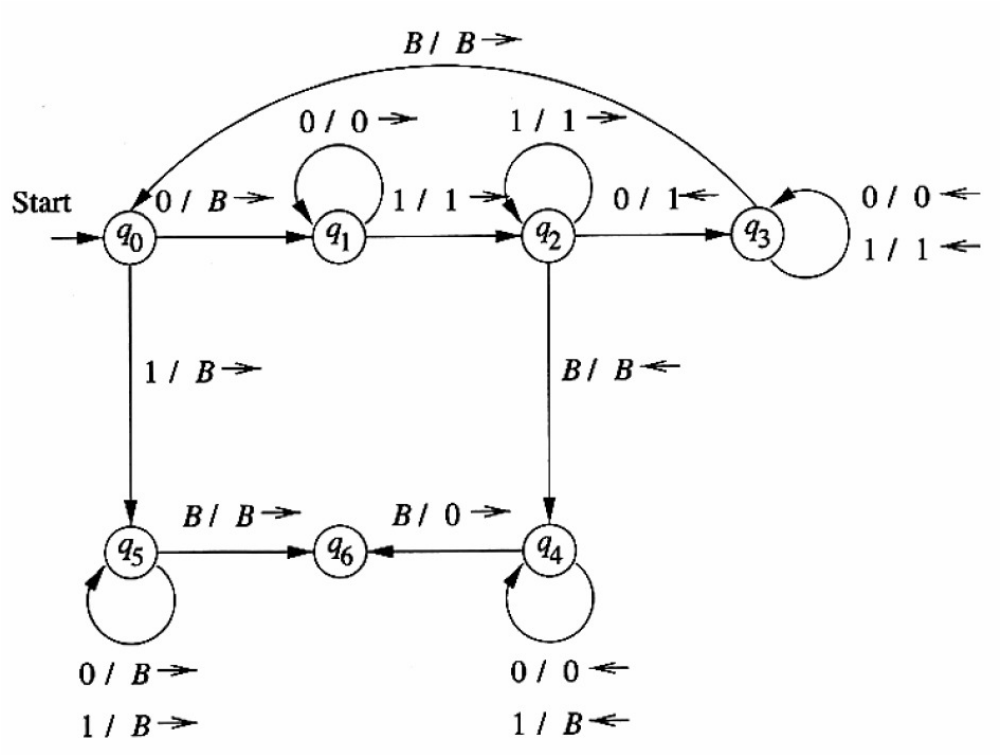


Ejemplo: Diseñar una TM que calcula la función $\dot{-}$ llamada *monus* o substracción propia, que se define como: $m \dot{-} n = max(m - n, 0)$.

La siguiente tabla y diagrama de transición lo definen:

	Símbolo		
Estado	0	1	B
q_0	(q_1, B, R)	(q_5, B, R)	$-$
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	$-$
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
q_6	$-$	$-$	$-$

Diagrama de transición:



8.2.3 Lenguaje de una TM

El lenguaje que aceptan las TMs es el conjunto de cadenas $w \in \Sigma*$ tales que $q_0 w \overset{*}{\vdash} \alpha p \beta$ para un estado p en F y cualesquiera cadenas en la cinta α y β.

Los lenguajes que aceptan las TMs se llama lenguajes *recursivamente enumerables* o lenguajes RE.

Halting: otra forma de aceptar cadenas usado normalmente en las TMs es aceptar por paro (*halting*). Decimos que una TM se para (*halts*) si entra a un estado q leyendo el símbolo de la cinta X y no existe ningún movimiento, i.e., $\delta(q, X)$ no está definido.

Por ejemplo la máquina de Turing que calcula $\dot{-}$, se para (*halts*) con

todas las cadenas de 0's y 1's ya que eventualmente alcanza el estado q_6.

Asumimos que una TM siempre se para en un estado de aceptación.

Ejercicios:

Extensiones a TMs:

Una TM es tan poderosa como una computadora convencional.

Se pueden realizar varias extensiones a una TM que permiten hacer especificaciones más simples, aunque no aumentan su expresividad (reconocen los mismos lenguajes).

8.2.4 Almacenar información en un estado

Podemos usar el control finito de una TM no sólo para indicar la posición actual, sino también para almacenar una cantidad finita de datos.

No se extiende el modelo, lo que hacemos es que pensamos en el estado como una tupla.

Por ejemplo, si queremos aceptar $01^* + 10^*$ podemos almacenar el primer elemento que se lee y asegurarnos que ese elemento no aparezca en el resto de la cadena.

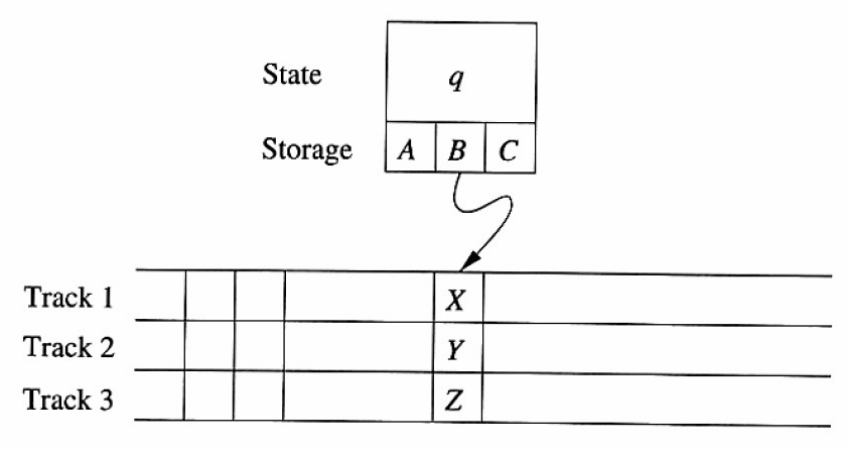
El $M = (Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], \{[q_1, B]\})$

Su función de transición es:
$\delta([q_0, B], a) = ([q_1, a], a, R)$ para $a = 0$ o $a = 1$.
$\delta([q_1, a], ca) = ([q_1, a], ca, R)$ donde ca es el complento de a. Si $a = 0$ entonces $ca = 1$, y viceversa.
$\delta([q_1, a], B) = ([q_1, B], B, R)$ para $a = 0$ o $a = 1$, llegando al estado de aceptación: $[q_1, B]$.

8.2.5 Tracks múltiples

Otro truco es pensar que tenemos varios caminos paralelos. Cada una tiene un símbolo y el alfabeto de la cinta de la TM es una tupla con un componente por cada camino.

El siguiente dibujo ilustra una TM con almacenamiento de información en el control y con caminos múltiples:



Un uso común de esto es usar un camino para guardar información y otro para guardar una marca. Por ejemplo, si queremos reconocer: $L_{wcw} = \{wcw | w \in (0+1)^+\}$

$M = (Q, \Sigma, \Gamma, \delta, [q_1, B], [B, B], \{[q_9, B]\})$

- Q: el conjunto de estados en $\{q_0, q_1, \ldots, q_9\} \times \{0, 1, B\}$, osea pares que tienen un estado de control (q_i) y un componente de dato ($0, 1$ o *blank*).
- Γ: los símbolos de la cinta son $\{B, *\} \times \{0, 1, c, B\}$, donde el primer componente es vacío (*blank*) o $*$ (para marcar símbolos del primer y segundo grupo de 0's y 1's).
- Σ: los símbolos de entrada son $[B, 0]$ y $[B, 1]$.
- δ: la función de transición es (donde a y b pueden ser 0 o 1):
  1. $\delta([q_1, B], [B, a]) = ([q_2, a], [*, a], R)$

2. $\delta([q_2, a], [B, b]) = ([q_2, a], [B, b], R)$
3. $\delta([q_2, a], [B, c]) = ([q_3, a], [B, c], R)$
4. $\delta([q_3, a], [*, b]) = ([q_3, a], [*, b], R)$
5. $\delta([q_3, a], [B, a]) = ([q_4, B], [*, a], L)$
6. $\delta([q_4, B], [*, a]) = ([q_4, B], [*, a], L)$
7. $\delta([q_4, B], [B, c]) = ([q_5, B], [B, c], L)$
8. $\delta([q_5, B], [B, a]) = ([q_6, B], [B, a], L)$
9. $\delta([q_6, B], [B, a]) = ([q_6, B], [B, a], L)$
10. $\delta([q_6, B], [*, a]) = ([q_1, B], [*, a], R)$
11. $\delta([q_5, B], [*, a]) = ([q_7, B], [*, a], R)$
12. $\delta([q_7, B], [B, c]) = ([q_8, B], [B, c], R)$
13. $\delta([q_8, B], [*, a]) = ([q_8, B], [*, a], R)$
14. $\delta([q_8, B], [B, B]) = ([q_9, B], [B, B], R)$

8.2.6 Subrutinas

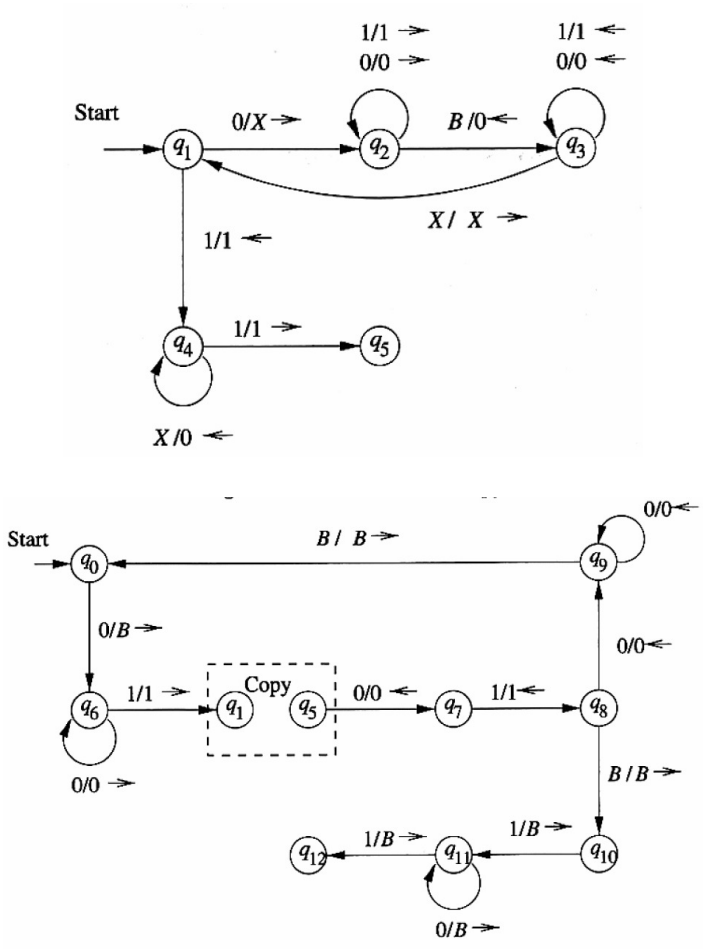
Una subrutina de una TM es un conjunto de estados que realiza algún proceso útil. Su llamado ocurre cuando se hace una transición a su estado inicial. Si se requiere "llamar" varias veces desde diferentes estados, se tienen que hacer varias copias con estados diferentes.

Por ejemplo, la siguiente TM implementa la multiplicación, y empieza con una cadena 0^m10^n1 en su cinta y termina con la cadena 0^{mn} en la cinta.

El núcleo es una subrutina llamada *Copia* que copia un bloque de 0's al final. Copia convierte una ID de forma $0^{m-k}1^q_1 0^n 10^{(k-1)n}$ a la siguiente ID $0^{m-k}1q_5 0^n 10^{kn}$.

Las siguientes figuras ilustran la subrutina *Copia* y la TM completa para

multiplicación:



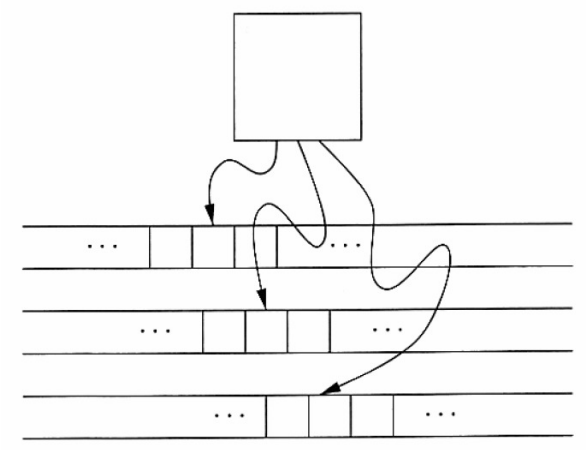
8.2.7 TM con múltiples cintas

Tiene un número finito de cintas. En el estado inicial los símbolos de entrada son colocados en la primera cinta y la cabeza del resto de las cintas en un B arbitrario.

En un movimiento, el control entra a un nuevo estado y en cada cinta se escribe un símbolo en la celda que está leyendo, cada cabeza de cada cinta hace un movimiento que puede ser a la izquierda, derecha o quedarse donde

está.

Las TM con cintas múltiples aceptan entrando a un estado de aceptación.



Se puede demostrar que todo lenguaje aceptado por una TM con múltiples cintas es recursivamente enumerable.

Básicamente para una TM con k cintas se simula una TM de una cinta con $2k$ caminos, donde la mitad de los caminos guardan el contenido de las cintas y la otra mitad guardan una marca que indica la posición de las cabezas de cada cinta.

8.2.8 TM no determinista

En una TM no determinista (NTM) la función de transición es ahora un conjunto de tripletas y puede seleccionar cualquier elemento de ese conjunto en cada transición. $\delta(q, X) = \{(q_1, Y_1, D_1), (q_2, Y_2, D_2), \ldots, (q_k, Y_k, D_k)\}$

Una NTM acepta una cadena w si existe una secuencia de selecciones de movimientos que nos llevan a un ID con un estado de aceptación.

De nuevo se puede probar que para cada NTM existe una TM (determinista).

Básicamente se siguen las "m" posibles opciones en cada movimiento

siguiendo un esquema tipo *breadth-first*.

Ejercicios:

8.3 Máquinas de Turing restringidas

Se pueden imponer ciertas restricciones a la TM y de todos modos mostrar que aceptan el mismo lenguaje.

8.3.1 TM con cinta semi-infinita

La cinta es infinita en un sentido, la cadena de entrada se coloca al principio de la cinta la cual no continua a la izquierda. También se incluye la restricción de no poder escribir B (*blank*) en la cinta.

Se puede demostrar que un TM normal se puede simular con una TM semi-infinita usando dos caminos, uno que simula la cinta a la izquierda de la cadena de entrada y otro que simula la otra parte de la cinta.

X_0	X_1	X_2	$\ldots$
$*$	X_{-1}	X_{-2}	$\ldots$

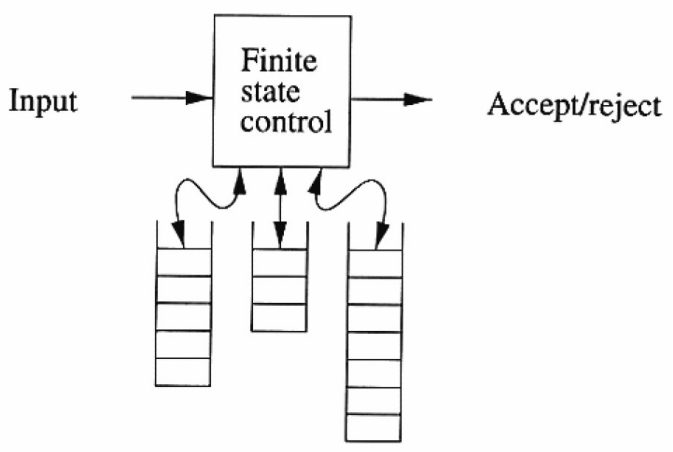
Los estados en la TM semi-infinita son los que tiene la TM original junto con U o L para representar arriba (U) o abajo (L), además de un par de estados para preparar la TM semi-infinita.

Las transiciones en la cinta de arriba son iguales a las de la TM original, y las de abajo son contrarias (si la TM original se mueve a la derecha, la TM semi-infinita inferior se mueve a la izquierda y al revés, si se mueve a la izquierda la TM original la otra se mueve a la derecha).

Sólo se tiene que tener cuidado en las situaciones en donde se cambia de un camino al otro.

8.3.2 Máquinas multistack

Podemos pensar en una generalización de los PDAs cuando consideramos varios *stacks*.



Una máquina de *k-stacks* es un PDA determinista con k *stacks*. Un movimiento en esta máquina, cambia el estado y reemplaza el símbolo de arriba de cada *stack* con una cadena normalmente diferente para cada *stack*.

La transición sería: $\delta(q, a, X_1, X_2, \ldots, X_k) = (p, \gamma_1, \gamma_2, \ldots, \gamma_k)$. Donde en estado q con X_i hasta arriba de cada uno de los $i = 1, 2, \ldots, k$ *stacks*, se consume el símbolo a y se reemplaza cada X_i por γ_i.

Se puede demostrar que un PDA con 2 *stacks* puede simular una TM.

En la demostración se asume que al final de la cadena de entrada existe un símbolo especial que no es parte de la entrada.

Lo primero que se hace es que se copia la cadena al primer *stack*, se hace *pop* de este *stack* y *push* en el segundo *stack*, con esto el primer elemento de la cadena de entrada está hasta arriba del segundo *stack*, y luego se empiezan a simular las transiciones de estados.

El primer *stack* vacío nos representa todos los *blanks* a la izquierda de la cadena de entrada y en general lo que está a la izquierda de donde apunta la cabeza de la cinta de la TM.

Si TM reemplaza X por Y y se mueve a la derecha, PDA introduce (*pushes*) Y en el primer *stack* y saca (*pops*) X del segundo 'emphstack.

Si TM reemplaza X por Y y se mueve a la izquierda, PDA saca (*pops*) el primer elemento del primer *stack* (digamos Z) y reemplaza X por ZY en el segundo *stack*.

8.3.3 Máquinas contadoras (*counter machines*)

Una máquina contadora tiene la misma estructura que una máquina *multi-stack*, sólo que en lugar de *stacks* tiene contadores.

Un contador tiene algún entero positivo y sólo puede distinguir entre 0 (cero) o un contador diferente de cero.

En cada movimiento, cambia de estado y suma o resta 1 del contador (que no puede volverse negativo).

También podemos verlo como una máquina *multistack* restringida que tiene dos símbolos de stack Z_0 (marca el fondo) y X. El tener $X_i Z_0$ nos identifica al contador i.

Se puede demostrar que los lenguajes aceptados por una máquina de un contador son los CFL.

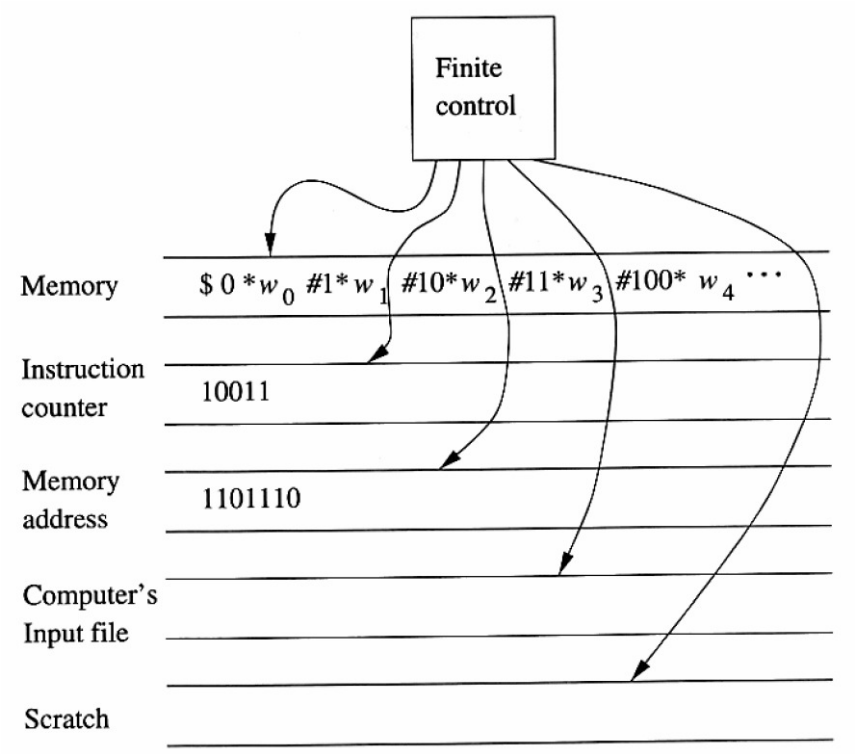
Se puede demostrar que cualquier lenguaje recursivamente enumerable puede ser aceptado por una máquina de dos contadores.

8.4 Máquinas de Turing y Computadoras

Una computadora puede simular una máquina de Turing. Aunque con un número muy grande de símbolos y cadenas en principio infinitas, se podrían tener problemas de memoria, se puede codificar la TM (TM universal) y simular en una computadora convencional sin problemas de memoria.

Una TM puede simular una computadora usando varias cintas para tomar

en cuenta los diferentes procesos que se realizan en una computadora (para representar la memoria de la computadora, la siguiente instrucción a realizar, si se requiere copiar cierto contenido de la memoria, cambiarlo, etc.).



Se puede demostrar que si una computadora tiene sólo instrucciones que incrementan la longitud del tamaño de las palabras en 1 y las instrucciones de tamaño w se pueden realizar en una TM con cintas múltiples en $O(k^2)$ pasos, entonces una TM parecida a la mostrada arriba pueden simular n pasos de la computadora en $O(n^3)$ pasos.