Máquinas de Turing

Máquina de Turing (MT): sexteta $(S, \Sigma, \Gamma, \delta, \iota, h)$, en que:

- S es un conjunto finito de *estados*.
- Σ es el (*alfabeto de la máquina*), sin el símbolo *blanco* $\triangle$.
- Γ es el conjunto de (*símbolos de cinta*), $\triangle \in \Gamma \supseteq \Sigma$ y $L, R \notin \Gamma$.
- $\delta : (S - \{h\}) \times \Gamma \to S \times (\Gamma \cup \{L, R\})$ es la *función de transición*.
- $\iota \in S$ es el estado *inicial*.
- $h \in S$ es el estado de *alto ("halt")*, $h \neq \iota$.

La *cinta* de una MT es una sucesión infinita de celdas. En cada celda se guarda un elemento de Γ. La MT cuenta con una *cabeza* de cinta, posicionada en una celda. Al ejecutar la MT una transición $\delta(p, \gamma) = (q, \nu)$, con la cabeza posicionada en una celda que tiene al símbolo γ, si $\nu \in \Gamma$ la máquina escribe el símbolo ν en dicha celda; si $\nu = L$ ó $\nu = R$, la máquina mueve la cabeza de cinta una celda a la izquierda o a la derecha, respectivamente. Una MT ejecuta transiciones mientras no alcance el estado h. Si no llega a h, es porque la computación no termina (debido a la ejecución de un ciclo de transiciones), o porque la transición bajo ejecución indica un movimiento hacia la izquierda de la primera celda, en este último caso se dice que la ejecución termina anormalmente.

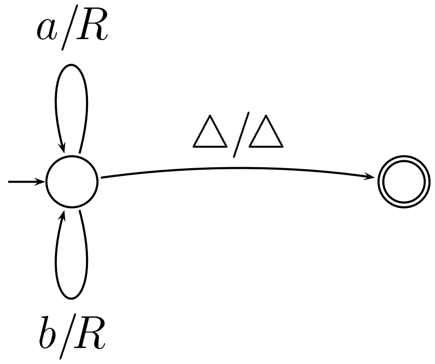
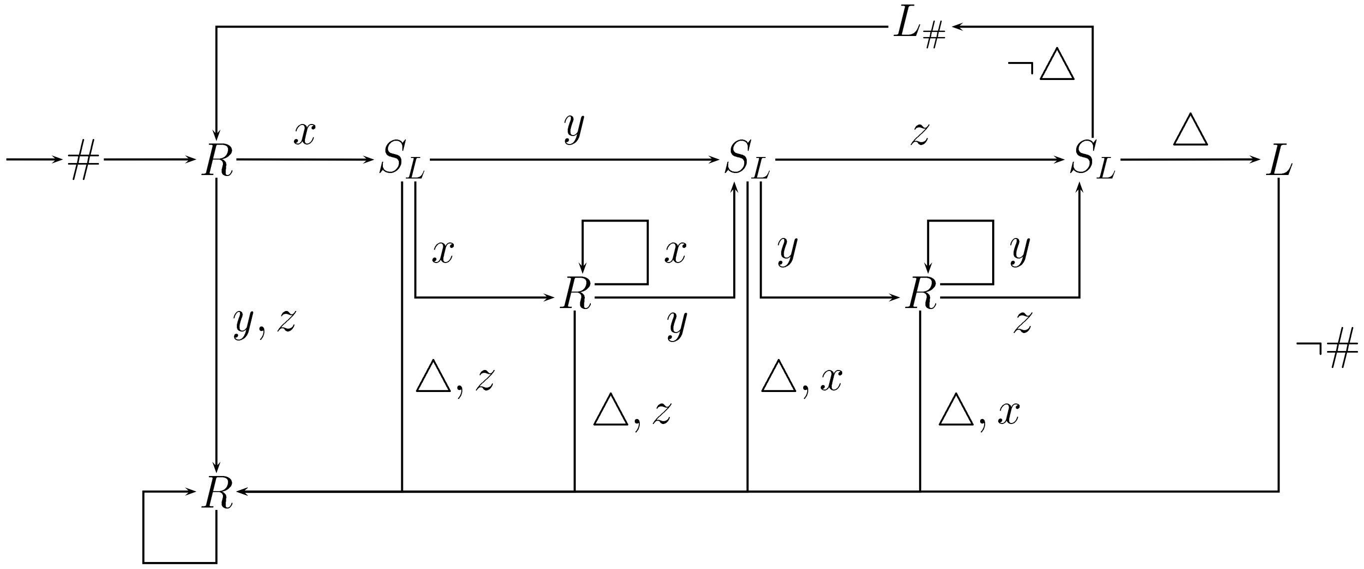


Diagrama de transiciones de una MT con $\Gamma = \{a, b, \triangle\}$. Mueve la cabeza lectora hacia la derecha hasta que encuentra un espacio en blanco.



Máquina de Turing M tal que $L(M) = \{x^n y^n z^n : n \in \mathbb{N}\}$, con $\Gamma = \{x, y, z, \triangle, \#\}$. Inicia con la cabeza en la primera celda de la cinta, con la palabra por analizarse a partir de la segunda celda.

Para λ, representada mediante $\underline{\triangle}\triangle\triangle\ldots$, a la salida de R se tendrá $\#\underline{\triangle}\triangle\ldots$ y de ahí la máquina pasa al estado de alto.

Si $n \geq 1$, el ciclo superior se encarga de quitar una x, una y y una z. Por ejemplo, para $\underline{\triangle}xxyyzz\triangle\triangle\ldots$ una vez ejecutado el primer bloque R se tendrá $\#\underline{x}xyyzz\triangle\triangle\ldots$. Después de una vuelta completa el contenido de la cinta será $\#\underline{x}yz\triangle\triangle\ldots$. La ejecución entra de nuevo al ciclo, pero a la salida del último bloque S_L se tendrá $\#\underline{\triangle}\triangle\ldots$, por lo que se pasa a ejecutar el bloque L, dando por resultado $\underline{\#}\triangle\triangle\ldots$; entonces la máquina pasa al estado de alto. Nótese que si después de ejecutarse L bajo la cabeza no quedara $\#$, en la cinta se tendría una cadena no terminada en z, se ejecutaría la transición hacia el ciclo en la parte inferior izquierda de la figura, encargado de mover eternamente la cabeza hacia la derecha, y la computación no terminaría. La ejecución también entra en este ciclo cuando la máquina no encuentra x, y o z en una celda donde uno de esos símbolos debería ocurrir.

Máquinas de Turing como aceptadoras de lenguajes

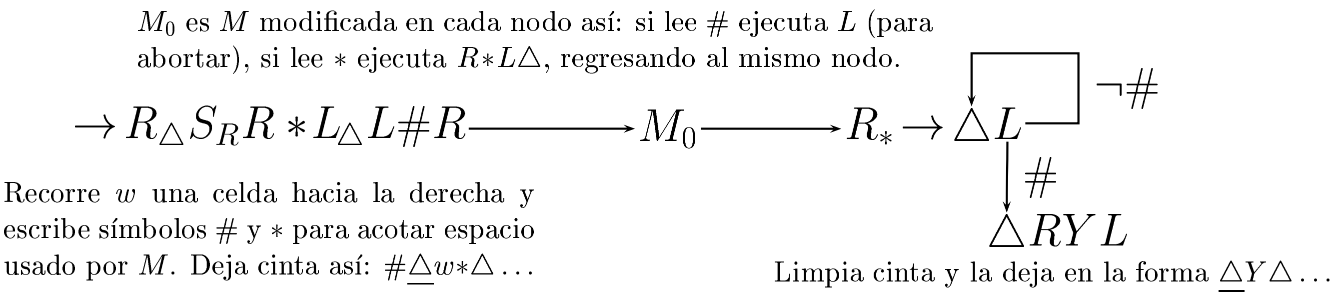
Definición A. Sea M una MT y $w = x_1x_2\cdots x_n \in \Sigma^\star$. *$M$ acepta a w* si y solo si, representada w a partir de la segunda celda de la cinta (en la forma $\triangle x_1x_2\cdots x_n\triangle\triangle\cdots$) y posicionada la cabeza de cinta en la primera celda, mediante una computación M logra pasar de su estado inicial a su estado de alto. El *lenguaje aceptado* por M es $L(M) = \{w \in \Sigma^\star : M \text{ acepta a } w\}$.

Alternativamente, representada w y arrancando M como en la definición anterior, ésta pudiera cambiarse por la siguiente:

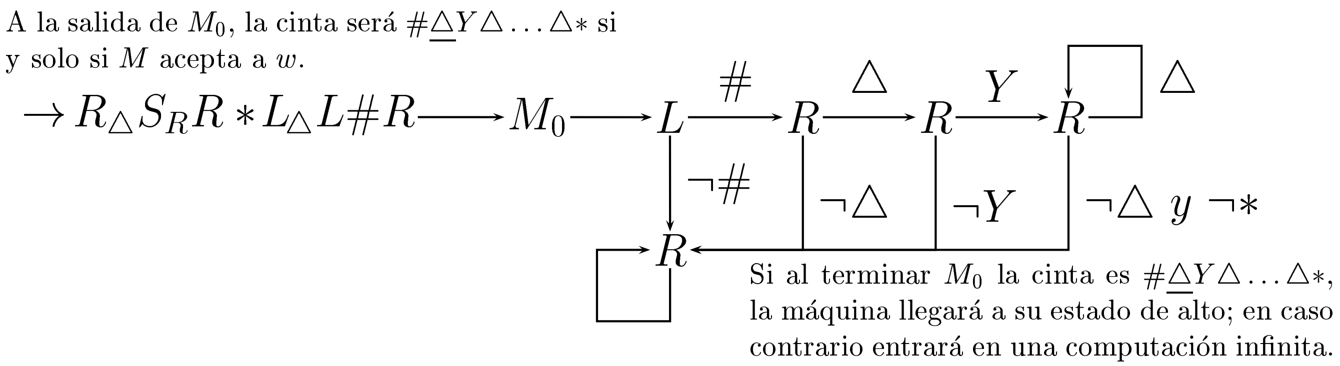
Definición B. M acepta a w si y solo si M para con su cinta en la configuración $\underline{\triangle}Y\triangle\ldots$ (léase ¡sí!).

Nótese que en el sentido de la segunda definición M puede llegar a su estado de alto sin aceptar a la palabra analizada. Sin embargo, veamos que la clase de lenguajes *Turing-aceptables* obtenida a partir de cualquiera de las dos definiciones es la misma.

Si $L(M)$ es un lenguaje aceptado en el sentido de la Def. A, la siguiente MT lo acepta de acuerdo a la Def. B.



Si $L(M)$ es aceptado de acuerdo a la Def. B, tenemos:



Máquina de Turing con k cintas (k-MT).

$$\delta : (S - \{h\}) \times \Gamma^k \to S \times (\Gamma \cup \{L, R\}) \times \{1, \ldots, k\}$$

$\delta(p, x_1, \ldots, x_i, \ldots, x_k) = (q, y, i)$ significa "si M se encuentra en su estado p, con las componentes de $(x_1, \ldots, x_i, \ldots, x_k)$ como símbolos actuales, entonces M debe escribir y en lugar de x_i y cambiarse al estado q".

$\delta(p, x_1, \ldots, x_k) = (q, L, i)$ y $\delta(p, x_1, \ldots, x_k) = (q, R, i)$ se interpretan en forma parecida, indican un movimiento de la cabeza de la cinta i, una celda hacia la izquierda o hacia la derecha, respectivamente.

<u>Teorema 3.1.</u> Para cada k-MT M hay alguna MT tradicional (de una cinta) M' tal que $L(M')=L(M)$.

Demostración. Sea $M = (S, \Sigma, \Gamma, \delta, \iota, h)$. A partir de M se construye $M'=(S', \Sigma, \Gamma', \delta', \iota', h')$ como se explica a continuación.

1) Representación en M' de las k cintas de M.

Se define $\Gamma'=\Gamma \cup (\Sigma \times \{\triangle, 1\})^k \cup \{\#\}$, $(1, \# \notin \Gamma)$.

Colocada $w \in \Sigma^\star$ en la cinta, $\underline{\triangle}w\triangle\ldots$, en su primera etapa M' cambia la representación de w, a un formato en términos de los símbolos de cinta en $(\Sigma \times \{\triangle, 1\})^k$ (tuplas de longitud $2k$), con el propósito de mantener una imagen de las k cintas de M (w en la primera cinta, en blanco las celdas abajo de las de w):

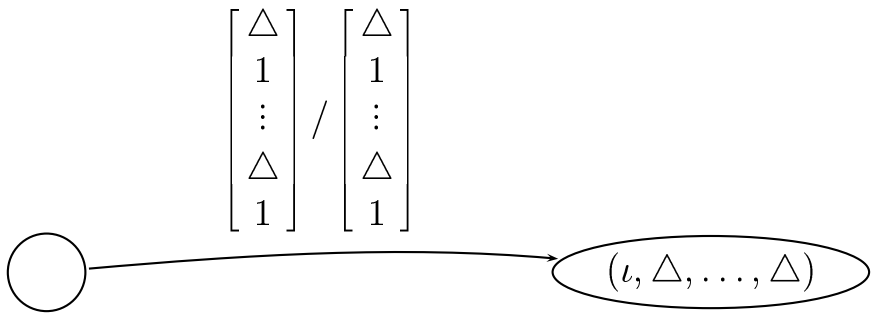
- Mediante $\rightarrow R_\triangle S_R L_\triangle$ desplaza a w una celda hacia la derecha: $\triangle\underline{\triangle}w\triangle\ldots$
- Mediante $\rightarrow L\#R$ escribe $\#$ en la primera celda: $\#\underline{\triangle}w\triangle\ldots$
- Cambia el formato de w (parte superior del siguiente diagrama) y regresa la cabeza a la segunda celda, donde escribe la tupla correspondiente a la posición inicial de las cabezas de M:

$$
\begin{tikzcd}
{} \arrow[r] & R \arrow[r, "\neg\triangle"] \arrow[d, "\triangle"'] & \} \arrow[r, "\sigma"] & \begin{bmatrix}\sigma\\ \triangle\\ \vdots\\ \triangle\\ \triangle\end{bmatrix} \arrow[lll, bend right, start anchor=east, end anchor=north] \\
& L_\triangle \arrow[r] & \begin{bmatrix}\triangle\\ 1\\ \vdots\\ \triangle\\ 1\end{bmatrix} &
\end{tikzcd}
$$

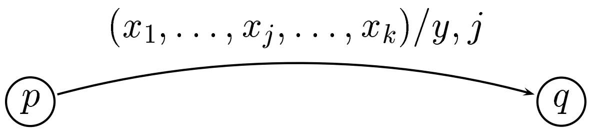
2) Simulación en M' de las acciones de M.

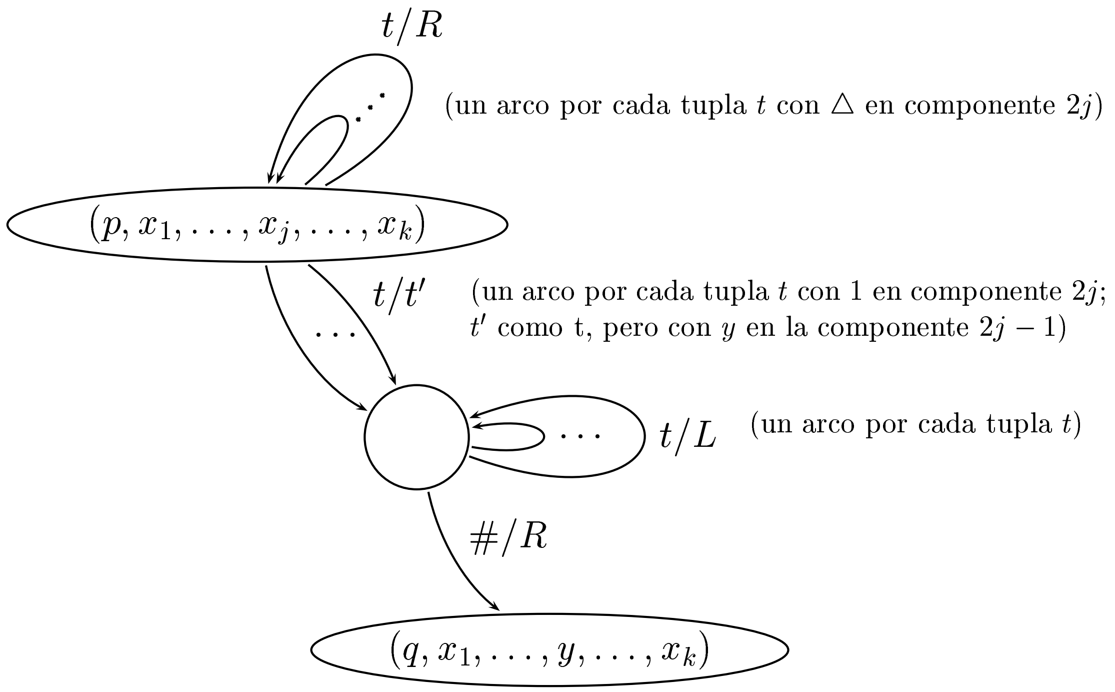
Como conjunto de estados de M' se toma $S' = S_1 \cup (S \times \Gamma^k)$.

Cada estado $(p, x_1, \ldots, x_k) \in S \times \Gamma^k$ (*estado compuesto*) sirve para simular que M se encuentra en su estado p con símbolos actuales $x_1, \ldots, x_k$. S_1 comprende los estados adicionales que se necesiten, como aquellos requeridos en la etapa 1. Para iniciar la simulación de las acciones de M, M' pasa al estado compuesto $(\iota, \triangle, \ldots, \triangle)$, partiendo del estado final de la etapa 1:

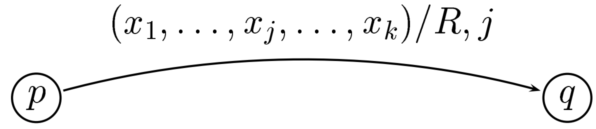


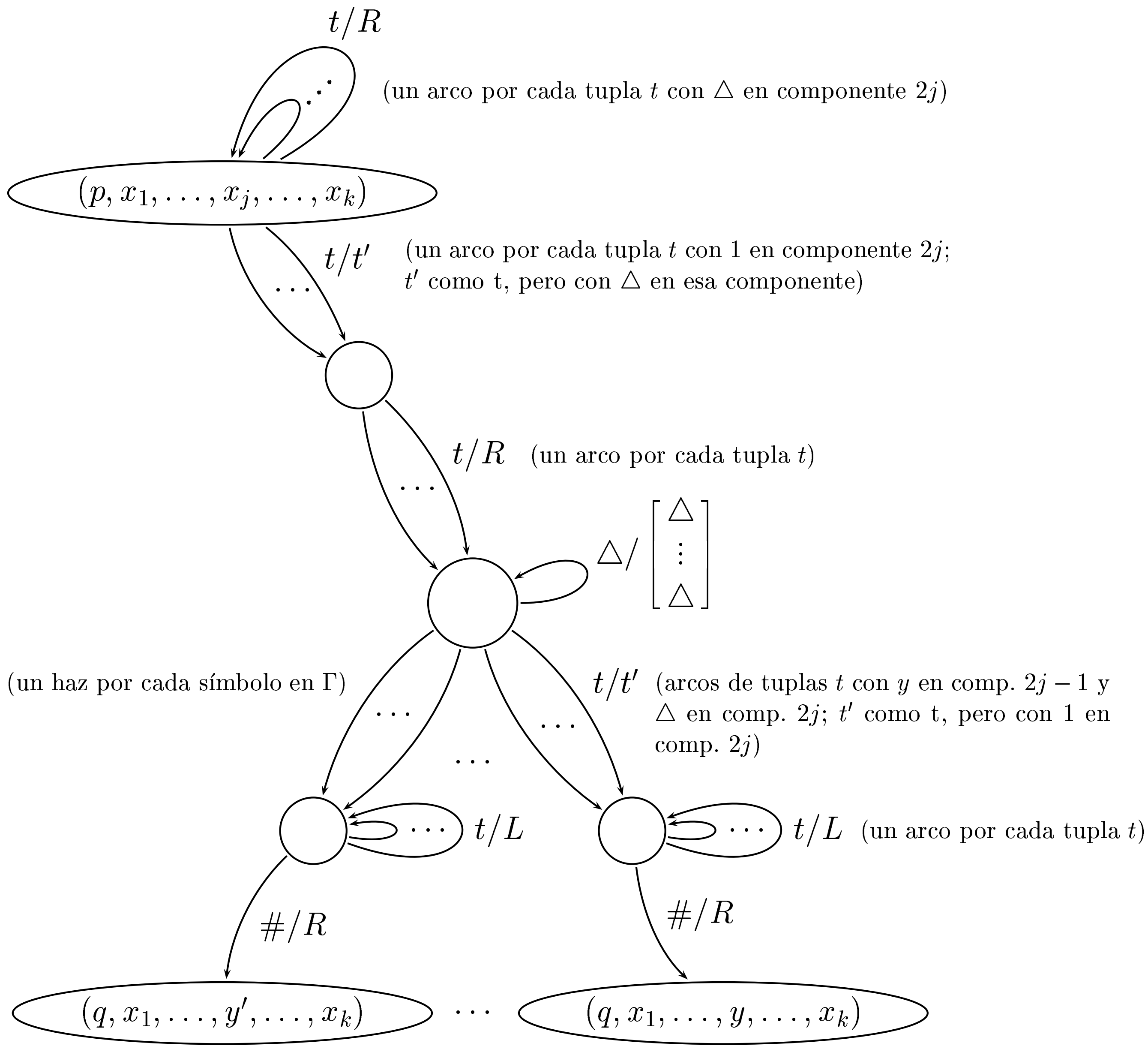
Simulación en M' de transiciones



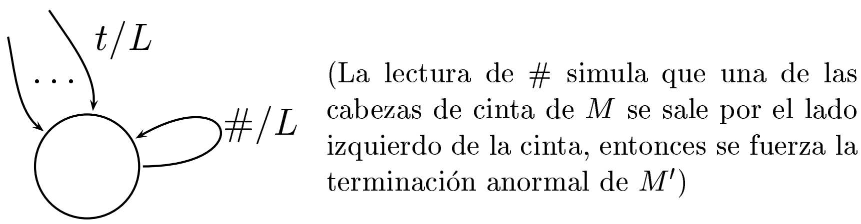


Simulación en M' de transiciones





Cada transición de movimiento hacia la izquierda se trata igual, pero cambiando los arcos incidentes en el nodo central así:



Teorema 3.2. Para cada MT no determinista M hay alguna MT tradicional D tal que $L(D) = L(M)$.

Demostración. A partir de M se construye una 3-MT M' tal que $L(M') = L(M)$ y el Teorema 3.1 garantiza la existencia de D.

En la cinta 1 de M' no se escribe, solo se mantiene la palabra w bajo análisis: $\underline{\triangle} w \triangle \ldots$.

La cinta 2 sirve para simular las diversas computaciones que la máquina indeterminista puede seguir al analizar a w. M' empieza la simulación de cada una de estas posibles computaciones copiando w a la cinta 2, dejando ésta en la forma $\#\underline{\triangle} w \triangle \ldots$. Como en los teoremas anteriores, el símbolo $\#$ sirve para detectar aquellas computaciones de M que terminen anormalmente.

La cinta 3 sirve para representar y administrar las diversas computaciones de M. Numerando las transiciones de M (e.d. los arcos de su diagrama), mediante un bloque similar al de la Figura 3.10 del libro, en dicha cinta se pueden generar sistemáticamente todas las sucesiones finitas de transiciones de M. Cada vez que se genere una sucesión, M' intenta aplicarla sobre la cinta 2; si al hacerlo M' logra llegar a su propio estado de alto, entonces w es aceptada; en caso contrario, M' inicia un nuevo ciclo (prepara cinta 2 y genera siguiente sucesión).

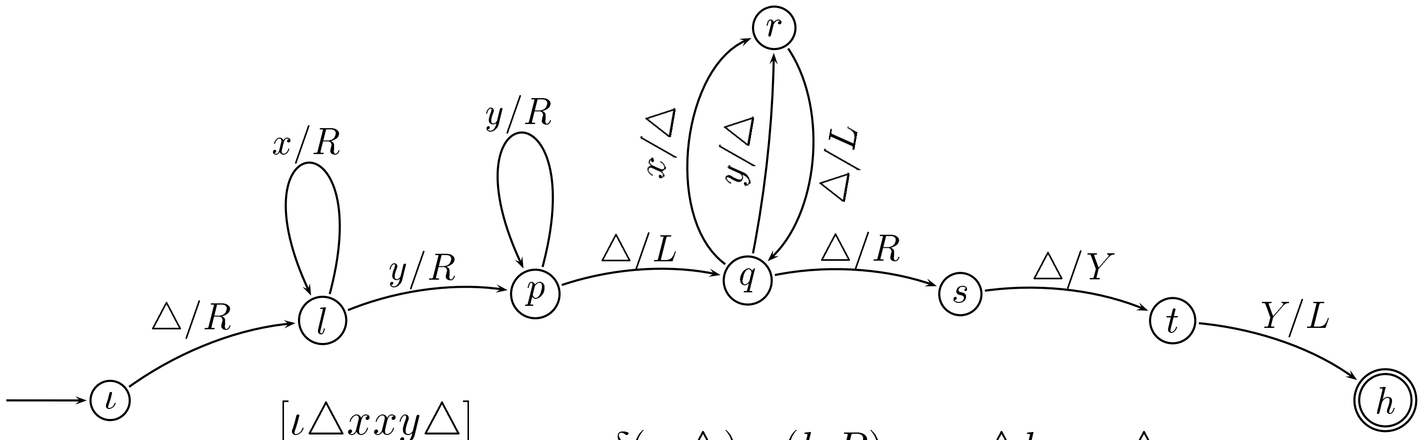
Nótese que si M acepta a w, eventualmente M' simulará una computación de aceptación de w. Si M no acepta a w, M' tampoco, porque su propia computación nunca termina.

Tarea. Hacer los diagramas (de bloques y/o de transición) para M'. (Pista: La cinta 3 también puede usarse para administrar información acerca del estado de la máquina simulada.)

Lengs. Turing-aceptables vs. estructurados por frases.

$$\begin{aligned}
&S \to xyNSz\\
&S \to \lambda\\
&yNx \to xyN\\
&yNz \to yz\\
&yNy \to yyN
\end{aligned}$$

Producciones de una gramática G tal que $L(G)=\{x^n y^n z^n : n\in \mathbb{N}\}$.



$[\iota\triangle xxy\triangle]$		
$[\triangle lxxy\triangle]$	$\delta(\iota,\triangle)=(l,R)$	$\triangle l \to \iota\triangle$
$[\triangle xlxy\triangle]$	$\delta(l,x)=(l,R)$	$xl \to lx$
$[\triangle xxly\triangle]$	$\delta(l,x)=(l,R)$	$xl \to lx$
$[\triangle xxyp\triangle]$	$\delta(l,y)=(p,R)$	$yp \to ly$
$[\triangle xxqy\triangle]$	$\delta(p,\triangle)=(q,L)$	$qy\triangle \to yp\triangle$
$[\triangle xxr\triangle\triangle]$	$\delta(q,y)=(r,\triangle)$	$r\triangle \to qy$
$[\triangle xqx\triangle\triangle]$	$\delta(r,\triangle)=(q,L)$	$qx\triangle \to xr\triangle$
$[\triangle xr\triangle\triangle\triangle]$	$\delta(q,x)=(r,\triangle)$	$r\triangle \to qx$
$[\triangle qx\triangle\triangle\triangle]$	$\delta(r,\triangle)=(q,L)$	$qx\triangle \to xr\triangle$
$[\triangle r\triangle\triangle\triangle\triangle]$	$\delta(q,x)=(r,\triangle)$	$r\triangle \to qx$
$[q\triangle\triangle\triangle\triangle\triangle]$	$\delta(r,\triangle)=(q,L)$	$q\triangle\triangle \to \triangle r\triangle$
$[\triangle s\triangle\triangle\triangle\triangle]$	$\delta(q,\triangle)=(s,R)$	$\triangle s \to q\triangle$
$[\triangle tY\triangle\triangle\triangle]$	$\delta(s,\triangle)=(t,Y)$	$tY \to s\triangle$
$[h\triangle Y\triangle\triangle\triangle]$	$\delta(t,Y)=(h,L)$	$h\triangle Y \to \triangle tY$

MT que acepta al lenguaje $\{x^m y^n : m\in\mathbb{N}, n\in\mathbb{N}^+\}$. La tabla muestra la secuencia de configuraciones al procesarse la palabra xxy, las transiciones aplicadas y las producciones de una derivación de $[\iota\triangle xxy\triangle]$ a partir de $[h\triangle Y\triangle]$.

T. 3.3. Cada lenguaje Turing-aceptable es estucturado por frases.

Demostración. Dada $M = (\mathcal{S}, \Sigma, \Gamma, \delta, \iota, h)$, la siguiente gramática $G = (V, T, S_G, R)$ genera a $L(M)$:

- Símbolos no terminales: $V = \{S_G\} \cup \{[,]\} \cup \mathcal{S} \cup \Gamma \cup \{Y\}$.
- Símbolos terminales: $T = \Sigma$.
- Producciones:

$$
\begin{array}{ll}
S_G \to [h\triangle Y\triangle] & \\
\triangle] \to \triangle\triangle] & \\
qy \to px & \text{para cada. } \delta(p,x) = (q,y) \\
xq \to px & \text{para cada } \delta(p,x) = (q,R) \\
qyx \to ypx & \text{para cada } \delta(p,x) = (q,L) \text{ y cada } y \in \Gamma \\
[\iota\triangle \to \lambda & \\
\triangle\triangle] \to \triangle] & \\
\triangle \to \lambda &
\end{array}
$$

Si $w \in L(M)$, hay secuencia $[\iota\triangle w\triangle], \ldots, [h\triangle Y\triangle]$ de configuraciones de M. Entonces

$$S_G \Rightarrow [h\triangle Y\triangle] \stackrel{*}{\Rightarrow} [\iota\triangle w\triangle] \Rightarrow w\triangle] \Rightarrow w$$

(después de aplicar $S_G \to [h\triangle Y\triangle]$, se usa $\triangle] \to \triangle\triangle]$ tantas veces como se requiera para que $[h\triangle Y\triangle \ldots \triangle]$ sea tan larga como la configuración mas larga de la secuencia considerada; entonces se aplican las producciones correspondientes a las transiciones de la secuencia, siguiendo el orden contrario al de la construcción de ésta; después, de ser necesario, se aplica $\triangle\triangle] \to \triangle]$ hasta obtener $[\iota\triangle w\triangle]$.)

Si $w \in L(G)$, a partir de una derivación $S_G \stackrel{*}{\Rightarrow} w$ se construye una secuencia de configuraciones de M, correspondiente a la aceptación de w.

T. 3.4. Cada lenguaje estructurado por frases es Turing-aceptable.

Demostración. Modificando ligeramente la prueba del Teorema 3.1, se puede obtener una para el caso de correspondientes máquinas de Turing no deterministas. Dada una gramática G estructurada por frases, para probar el presente teorema se muestra la existencia de una 2-MT no determinista N, tal que $L(N)=L(G)$. Entonces para N hay alguna 1-MT no determinista N' tal que $L(N')=L(N)$, y por el Teorema 3.2 se tendrá una MT convencional M tal que $L(M) = L(N')=L(G)$.

Por otro lado, la aplicación de cualquier regla de reescritura de la gramática G, $\alpha \to \beta$, puede efectuarla una MT convencional. Es decir, la máquina puede localizar en su cinta cualquier ocurrencia de la cadena α y reemplazar ésta por la cadena β. (Tarea: construya una máquina A para la producción $yNz \to yxMz$; use A como bloque de otra máquina que sustituya todas las ocurrencias de yNz en w, partiendo de la configuración $[\iota\triangle w\triangle]$.)

La máquina N se construye así: Dada $w \in \Sigma^\star$, N arranca con la cinta 1 configurada en la forma $\underline{\triangle} w\triangle \ldots$ y la cinta 2 en blanco: $\underline{\triangle}\triangle \ldots$. Escribe el símbolo inicial de G en la cinta 2 (además de símbolos auxiliares que le puedan servir). Entonces aplica repetidamente las producciones de G (la programación de esta actividad resulta muy simple gracias al indeterminismo). Si en algún momento en la cadena de la cinta 2 no ocurren símbolos no terminales, se compara dicha cadena con w; si las cadenas son iguales, la máquina para; en caso contrario, mueve hacia la izquierda cualquiera de las cabezas de cinta, hasta terminar anormalmente la ejecución. De esta manera, alguna de las (posibles) computaciones termina (es decir, w es aceptada) si y solo si $w \in L(G)$.

Teorema 3.5. Para cada alfabeto Σ hay algún lenguaje sobre Σ que no es estructurado por frases.

Demostración. Sea L un lenguaje (sobre Σ) estructurado por frases. La demostración consiste en mostrar que hay alguna MT con símbolos de cinta en $\Sigma \cup \{\triangle\}$ y con L como lenguaje de aceptación. El conjunto de MT con símbolos de cinta en $\Sigma \cup \{\triangle\}$ se puede numerar fácilmente. Entonces, como $|\mathbb{N}| < |\mathcal{P}(\Sigma^\star)|$, se infiere que hay lenguajes sobre Σ que no son estructurados.

Sabemos que hay alguna MT M tal que $L(M) = L$. Si M no tiene a $\Sigma \cup \{\triangle\}$ como conjunto de símbolos de cinta, a partir de ella se construye otra, M', que si lo tenga, de la manera siguiente.

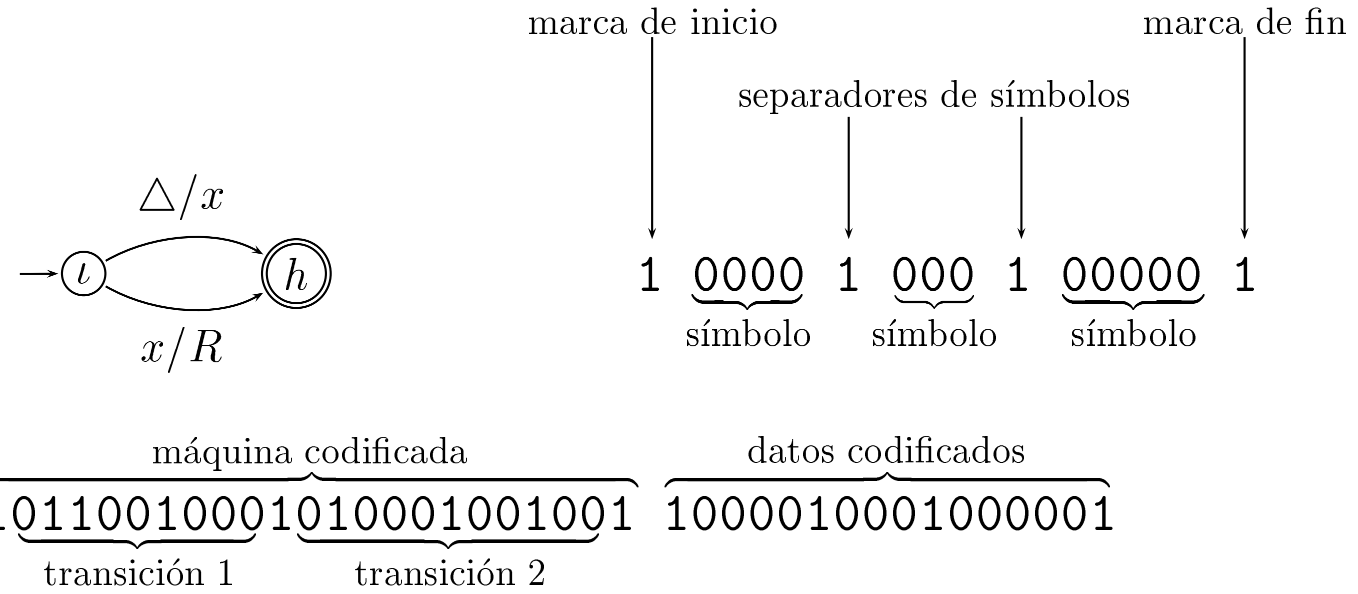
La máquina M' tiene dos componentes, A y B. La primera se encarga de traducir el contenido inicial de la cinta, $\underline{\triangle} w \triangle \ldots$, a un código en términos de $\triangle$ y un símbolo $x \in \Sigma$. Este código se construye representando los símbolos de Σ mediante las cadenas $x, xx, \ldots, x^n$, en que $n = |\Sigma|$. Al traducir, A separa con un $\triangle$ las representaciones de símbolos contiguos en w. En particular, dos símbolos $\triangle$ consecutivos sirven para simular al símbolo $\triangle$ (de M). La componente A es la misma para toda M y llega a su estado de alto para toda $w \in \Sigma^\star$. Además, A se puede construir de tal suerte que sus símbolos de cinta sean de $\Sigma \cup \{\triangle\}$.

B se encarga de simular a M. Únicamente utiliza los símbolos de cinta x y $\triangle$. Esto se logra representando los demás símbolos de cinta de M, así como otros que pudieran necesitarse para la simulación, mediante las cadenas $x^{n+1}, \ldots, x^{n+k}$.

B llega a su estado de alto si y solo si M también lo hace así. Entonces, por construcción, M' (es decir, $\rightarrow AB$) para sí y solo si M para, o sea que $L(M') = L(M) = L$.

Como A es común a todas las máquinas M' así construidas, éstas se pueden numerar, listando primero aquellas para las que B sea de 2 estados, después las de 3 estados y así sucesivamente.

Un sistema para codificar máquinas de Turing.



Sea una MT $M = (S, \Sigma, \Gamma, \delta, \iota, h)$, con $\Gamma = \Sigma \cup \{\triangle\}$.

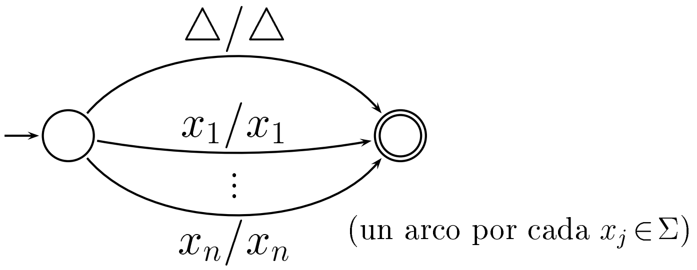
Se ordenan los elementos de S en la forma $\iota, h, s_3, \ldots, s_m$ y se asocian con las cadenas $0, 00, 000, \ldots 0^m$, respectivamente.

Se ordenan los elementos de $\Gamma \cup \{L, R\}$ en la forma $\triangle, L, R, x_1, \cdots, x_n$ y se asocian con las cadenas $\lambda, 0, 00, 000, \ldots 0^n$, respectivamente.

El código de una transición (p, x, q, ν) se construye listando los códigos de p, x, q y ν, separándolos con el símbolo 1.

El código de M se forma separando con 1 los códigos de sus transiciones, ordenadas respecto a p y x, y agregando un 1 al inicio y otro al final.

El código de M se puede interpretar también como la representación binaria de un número natural. Sea $\mathcal{M}$ el conjunto de máquinas de Turing con alfabeto Σ y símbolos de cinta en $\Sigma\cup\{\triangle\}$. Aunque hay números naturales cuya representación binaria (sin ceros no significativos) no es el código de alguna máquina en $\mathcal{M}$, podemos definir de la siguiente manera una función $i \mapsto M_i$ de N en $\mathcal{M}$. Si la representación binaria de i es el código de alguna máquina, M_i es esta máquina; en caso contrario, M_i es la máquina cuyo diagrama se muestra en la figura:



Usando esta función, definimos otra, $w \mapsto M_w$, de $\Sigma^\star$ en $\mathcal{M}$, tal que $M_w = M_{|w|}$.

Consideremos ahora al lenguaje $L_0 = \{w \in \Sigma^\star : M_w \text{ no acepta a } w\}$.

<u>Afirmación.</u> L_0 no es Turing-aceptable.

Demostración (por reducción al absurdo). Supongamos que L_0 es Turing-aceptable. Entonces hay alguna MT $M \in \mathcal{M}$ (ver dem. del Teorema 3.5) que acepta a L_0. Si tomamos una cadena $w \in \Sigma^\star$ cuya longitud $|w|$ tenga por representación binaria el código de M, tendremos que $M = M_w$. Entonces $L_0 = L(M_w)$.

Si $w \in L(M_w)$, M_w acepta a w, de donde obtenemos —por def. de L_0— $w \notin L_0 = L(M_w)$. Por otro lado, si $w \notin L(M_w)$, M_w no acepta a w, de donde obtenemos —también por def. de L_0— $w \in L_0 = L(M_w)$. O sea que $w \in L(M_w) \Leftrightarrow w \notin L(M_w)$, lo que es absurdo. Por lo tanto, L_0 no es Turing-aceptable.

Consideremos ahora al complemento de L_0:

$$\overline{L_0} = \{w \in \Sigma^\star : M_w \text{ acepta a } w\}.$$

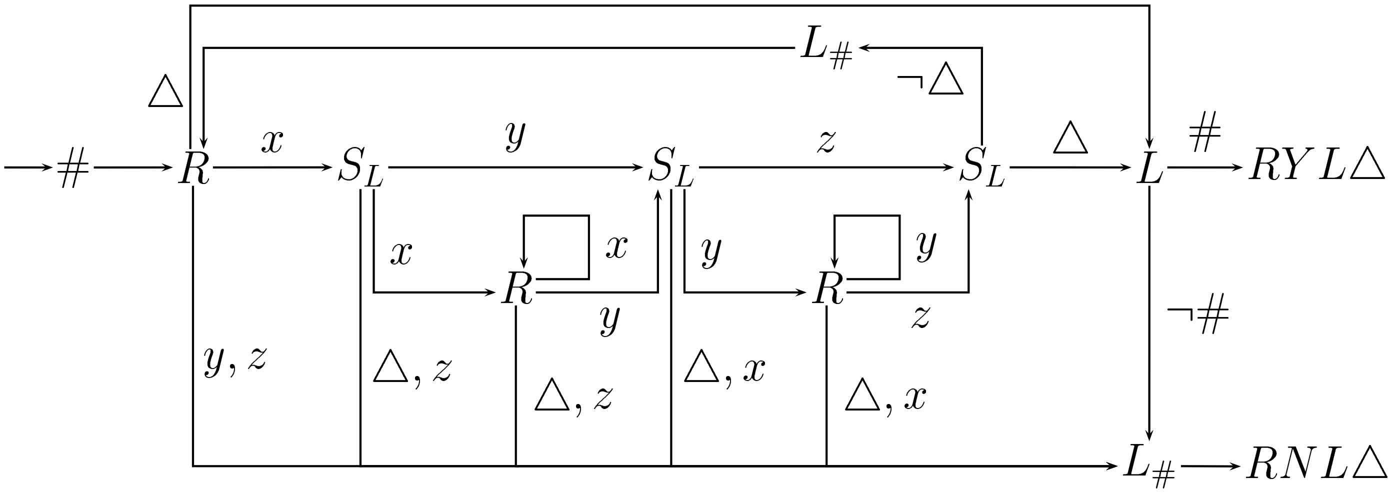
<u>Afirmación.</u> $\overline{L_0}$ es Turing-aceptable.

Demostración. Se construye una MT M_{pre} que procesa una cadena $w \in \Sigma^\star$ de la siguiente manera.

Genera el código binario que representa a M_w. (Para ello construye la representación binaria de $|w|$ y la analiza; si esta representación es el código de una MT, es precisamente el de M_w; en caso contrario, cambia el código por el de la máquina presentada al definir la función $i \mapsto M_i$.) También genera el código binario de w. Al final deja en la cinta el código de M_w seguido del de w.

La máquina $\rightarrow M_{pre}T_u$ acepta a $\overline{L_0}$, pues T_u simulará la aplicación de M_w a w, parando exitosamente si y solo si M_w acepta a w.

Ejemplos de lenguajes Turing-decidibles.



El lenguaje $L = \{x^n y^n z^n : n \in \mathbb{N}\}$ es Turing-decidible. Si $w \in L$, la MT mostrada deja $\underline{\triangle} Y \triangle \ldots$ en la cinta, en caso contrario la deja en la forma $\underline{\triangle} N \triangle \ldots$

$$S \to MN$$
$$M \to MP$$
$$N \to PN$$
$$M \to x$$
$$N \to y$$
$$P \to x$$
$$P \to y$$

	S	M	N	P
1		x	y	x y
2	xy	xx xy	xy yy	
3	xxy xyy	xxx xxy xyx xyy	xxy yxy xyy yyy	
4	$xxxy$ $xyxy$ $xxyy$ $xyyy$	$xxxx$ $xxyx$ $xyxx$ $xyyx$ $xxxy$ $xxyy$ $xyxy$ $xyyy$	$xxxy$ $xyxy$ $xxyy$ $xyyy$ $yxxy$ $yyxy$ $yxyy$ $yyyy$	

Cada lenguaje L l.c. es Turing-decidible. Dada una gramática en forma normal de Chomsky para $L - \{\lambda\}$, se puede construir una MT M que decida si $w \in \Sigma^\star$ está en L o no. Si $w = \lambda \in L$, M acepta a w. Si $w \neq \lambda$, M construye todas las cadenas de longitud menor o igual que $|w|$ generables a partir de cada símbolo no terminal, primero las de longitud 1, usando éstas construye las de longitud 2 y así sucesivamente hasta obtener las de longitud $|w|$. Si w se encuentra entre las palabras de longitud $|w|$ generadas a partir del símbolo inicial, M responde afirmativamente, en caso contrario responde negativamente. Si la gramática tiene n símbolos no terminales, conviene usar una máquina de $n+1$ cintas; en la primera cinta se mantiene una copia de w; cada una de las otras cintas se usa para mantener las palabras generadas a partir de un respectivo símbolo no terminal.

El problema del alto ("halting problem").

Sea $\mathcal{B}$ el conjunto de máquinas de Turing con alfabeto $\Sigma = \{0, 1\}$ y símbolos de cinta en $\Sigma \cup \{\triangle\}$. Sea $\rho : \mathcal{B} \to \Sigma^\star$ tal que $\rho(M)$ es igual al código binario de M. Denotemos mediante $\mathcal{C}$ al conjunto $\{\rho(M) \in \Sigma^\star : M \in \mathcal{B}\}$. $\mathcal{C}$ es un lenguaje sobre Σ, por lo que tiene sentido preguntarnos ¿$\rho(M) \in L(M)$?. Esta pregunta también se puede expresar en la forma ¿M acepta a $\rho(M)$?, o bien ¿M aplicada a $\rho(M)$ alcanza su estado de alto? Como la aplicación de M a $\rho(M)$ se puede simular mediante una MT universal, sería ideal tener una MT que se encargara de responder a nuestra pregunta, para cada M. Entonces formulamos el problema (conocido como "halting problem"): ¿Existe alguna MT tal que para cada M decida si M aplicada a $\rho(M)$ alcanza o no su estado de alto? A continuación veremos que no hay tal MT.

Definamos el siguiente lenguaje:

$$L_h = \{\rho(M) \in \mathcal{C} : M \text{ acepta a } \rho(M)\}$$

<u>Afirmación.</u> L_h no es Turing-decidible.

Demostración (por reducción al absurdo). Supongamos que L_h es Turing-decidible. Entonces hay alguna MT $M_h \in \mathcal{B}$ (ver dem. T.3.5) que hace alto para toda entrada $\rho(M)$, con cinta $\underline{\triangle}1\triangle\ldots$ si $\rho(M) \in L_h$, ó $\underline{\triangle}0\triangle\ldots$ en caso contrario.

La siguiente máquina, M_0, también pertenece a $\mathcal{B}$:

$$M_0: \quad \begin{tikzcd} {} \arrow[r] & M_h \arrow[r] & R \arrow[r, "1"] & R \arrow[loop right] \end{tikzcd}$$

M_0 hace alto si y solo si a la salida de M_h la cinta es $\underline{\triangle}0\triangle\ldots$

Veamos, siguiendo el diagrama, qué pasa cuando la entrada de M_0 es $\rho(M_0)$.

Si $\rho(M_0) \in L_h$, a la salida de M_h la cinta será $\underline{\triangle}1\triangle\ldots$, por lo que la computación se vuelve infinita al entrar al ciclo de la derecha, o sea que M_0 no acepta a $\rho(M_0)$, de donde se sigue —por def. de L_h— que $\rho(M_0) \notin L_h$.

Por otro lado, si $\rho(M_0) \notin L_h$, a la salida de M_h la cinta será $\underline{\triangle}0\triangle\ldots$, por lo que M_0 pasa a su estado de alto (no puede entrar al ciclo), o sea que M_0 acepta a $\rho(M_0)$, de donde se obtiene —también por def. de L_h— que $\rho(M_0) \in L_h$.

Tenemos entonces que $\rho(M_0) \in L_h \Leftrightarrow \rho(M_0) \notin L_h$, lo cual es absurdo. Por lo tanto, L_h no es Turing-decidible.

Funciones recursivas (de $\mathbb{N}^n$ a $\mathbb{N}$).

Funciones recursivas primitivas.

- Son primitivas recursivas las funciones:

  $C_k^{(n)}(x_1, \ldots, x_n) = k$ (constantes)
  $P_i^{(n)}(x_1, \ldots, x_i, \ldots, x_n) = x_i \ (1 \le i \le n)$ (proyecciones)
  $' : \mathbb{N} \to \mathbb{N}$ (sucesor)

- Si $g_1, \ldots, g_m : \mathbb{N}^n \to \mathbb{N}$ y $h : \mathbb{N}^m \to \mathbb{N}$ son recursivas primitivas, entonces $h \circ (g_1, \ldots, g_m) : \mathbb{N}^n \to \mathbb{N}$ es recursiva primitiva.
- Si $g : \mathbb{N}^n \to \mathbb{N}$ y $h : \mathbb{N}^{n+2} \to \mathbb{N}$ son recursivas primitivas, entonces $f : \mathbb{N}^{n+1} \to \mathbb{N}$,

  $$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$$
  $$f(x_1, \ldots, x_n, y') = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$$

  es recursiva primitiva.

Definición mediante minimalización (no acotada).

Sea $g : \mathbb{N}^{n+1} \to \mathbb{N}$ función total. Mediante la notación

$$f(x_1, \ldots, x_n) = \mu z[g(x_1, \ldots, x_n, z) = 0]$$

se define la función parcial $f : \mathbb{N}^n \to \mathbb{N}$ tal que $f(x_1, \ldots, x_n)$ es la mínima z tal que $g(x_1, \ldots, x_n, z) = 0$, si tal z existe.

Funciones μ-recursivas.

Las funciones constantes, proyecciones y sucesor son μ-recursivas. También lo son aquellas que se obtienen a partir de funciones μ-recursivas mediante composición, recursión primitiva y minimalización (con g total, en este último caso).

Máquinas de Turing para computar funciones (parciales) de $\mathbb{N}^n \to \mathbb{N}$.

Las MT con alfabeto de cinta $\{0, 1, \triangle\}$ se pueden usar para calcular funciones parciales $f : \mathbb{N}^n \to \mathbb{N}$ de la siguiente manera.

En la cinta, cada $x \in \mathbb{N}$ se representa en ***código unario***: $\overline{x} = 0^{x+1}$. Por ejemplo, $\overline{2} = 000$, $\overline{1} = 00$ y $\overline{0} = 0$.

Dada una MT M, para cada $n \in \mathbb{N}$ se define la función (parcial) $\varphi^{(n)} : \mathbb{N}^n \to \mathbb{N}$ *computada* por M de la siguiente manera: $\varphi^{(n)}(x_1, x_2, \ldots, x_n) = y$ si partiendo con la cinta configurada en la forma $\underline{\triangle}1\overline{x_1}1\overline{x_2}1\ldots1\overline{x_n}1\triangle\ldots$, M para con la cinta en la configuración $\underline{\triangle}1\overline{y}1\triangle\ldots$

Por conveniencia, en lugar de $\varphi^{(n)}(x_1, x_2, \ldots, x_n)$ también escribiremos $M^{(n)}(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$. (Esta última expresión también podemos leerla como la indicación de que M inicia la ejecución con la cinta configurada en la forma antes dicha.)

A las funciones así definidas se les llama *Turing-computables*. Se demuestra que coinciden con las funciones μ-recursivas.

Representemos mediante $[x_1, x_2, \ldots, x_n]$ a la MT que transforma la cinta $\underline{\triangle}w\ldots$ en $\underline{\triangle}1\overline{x_1}1\overline{x_2}1\ldots1\overline{x_n}1w\ldots$ (si w empieza con 1, no se agrega el 1 que aparece entre $\overline{x_n}$ y w).

A la manera de clases de MT antes tratadas, a cada máquina M aquí considerada se le asocia el número natural (de código binario) $\rho(M)$. También como se hizo para aquellas clases, se define una función que a cada número natural e le asocia una máquina M_e.

Definiendo las funciones $s_{n,m} : \mathbb{N}^{m+1} \to \mathbb{N}$ así:

$$s_{n,m}(e, y_1, \ldots, y_m) = \rho(\rightarrow [y_1, \ldots, y_m] M_e^{(m+n)}),$$

tendremos que

$$M^{(n)}_{s_{n,m}(e,y_1,\ldots,y_m)}(\overline{x_1}, \ldots, \overline{x_n}) = M_e^{(m+n)}(\overline{y_1}, \ldots, \overline{y_m}, \overline{x_1}, \ldots, \overline{x_n}). \quad (1)$$

<u>Teorema (de la recursión).</u> Sea $M^{(n+1)}$ una MT de $n+1$ argumentos. Entonces existe $e \in \mathbb{N}$ tal que

$$M^{(n+1)}(\overline{e}, \overline{x_1}, \ldots, \overline{x_n}) = M_e^{(n)}(\overline{x_1}, \ldots, \overline{x_n}). \quad (2)$$

Demostración. Consideremos la MT S que transforma la cinta

$$\underline{\triangle} 1 \overline{v} 1 \overline{x_1} 1 \overline{x_2} 1 \ldots 1 \overline{x_n} 1 \triangle \ldots$$

en

$$\underline{\triangle} 1 \overline{s_{n,1}(v,v)} 1 \overline{x_1} 1 \overline{x_2} 1 \ldots 1 \overline{x_n} 1 \triangle \ldots$$

y sea $z = \rho(\rightarrow S M^{(n+1)})$, entonces

$$M_z^{(n+1)}(\overline{v}, \overline{x_1}, \ldots, \overline{x_n}) = M^{(n+1)}(\overline{s_{n,1}(v,v)}, \overline{x_1}, \ldots, \overline{x_n})). \quad (3)$$

Tomando

$$e = s_{n,1}(z, z), \quad (4)$$

se obtiene la igualdad (2) así:

$$\begin{aligned} M^{(n+1)}(\overline{e}, \overline{x_1}, \ldots, \overline{x_n}) &\overset{(4)}{=} M^{(n+1)}(\overline{s_{n,1}(z,z)}, \overline{x_1}, \ldots, \overline{x_n}) \\ &\overset{(3)}{=} M_z^{(n+1)}(\overline{z}, \overline{x_1}, \ldots, \overline{x_n}) \\ &\overset{(1)}{=} M^{(n)}_{s_{n,1}(z,z)}(\overline{x_1}, \ldots, \overline{x_n}) \\ &\overset{(4)}{=} M_e^{(n)}(\overline{x_1}, \ldots, \overline{x_n}). \end{aligned}$$

Teorema (de punto fijo). Sea M una MT que computa a la función (total) $\varphi : \mathbb{N} \to \mathbb{N}$. Entonces $\exists e \in \mathbb{N}\ \forall x \in \mathbb{N}$

$$M_e(\overline{x}) = M_{\varphi(e)}(\overline{x}).$$

Demostración. Se construye una máquina M' tal que

$$M'(\overline{z}, \overline{x}) = M_{\varphi(z)}(\overline{x}). \tag{5}$$

Ésto se puede hacer con auxilio de una máquina universal U de la siguiente manera:

- M' inicia con la configuración de cinta $\underline{\triangle}1\overline{z}1\overline{x}1$
- Se simula la aplicación $M(\overline{z})$. Para ello, M' escribe el código de M, traduce $\overline{z}$ al código requerido por U y, mediante una aplicación de U, obtiene $\overline{\varphi(z)}$.
- Se simula la aplicación $M_{\varphi(z)}(\overline{x})$. Para ello, M' construye el código de $M_{\varphi(z)}$, traduce $\overline{x}$ a la forma requerida por U y aplica U a estos códigos.

Entonces, por el teorema de la recursión, existe $e \in \mathbb{N}$ tal que

$$M_e(\overline{x}) \overset{(2)}{=} M'(\overline{e}, \overline{x}) \overset{(5)}{=} M_{\varphi(e)}(\overline{x}).$$

Teorema de Rice. Sea $\mathcal{F}$ cualquier conjunto propio y no vacío de funciones Turing-computables (de una variable). Entonces no es Turing-decidible el conjunto

$$C_{\mathcal{F}}=\{e : M_e \text{ computa a } \varphi\in\mathcal{F}\}.$$

Demostración. (por reducción al absurdo). Supongamos que $C_{\mathcal{F}}$ es Turing-decidible. Entonces hay MT M que decide, es decir,

$$M(\overline{e}) = \begin{cases} 1, & \text{si } M_e \text{ computa alguna } \varphi\in\mathcal{F}; \\ 0, & \text{en caso contrario.} \end{cases}$$

Sean F y G máquinas de Turing que computan a f y g, respectivamente, con $f\in\mathcal{F}$ y $g\notin\mathcal{F}$. Construimos M' tal que

$$M'(\overline{t},\overline{x}) = \begin{cases} G(\overline{x}), & \text{si } M(\overline{t}) = 1; \\ F(\overline{x}), & \text{si } M(\overline{t}) = 0. \end{cases}$$

Por el teorema de la recursión, existe e tal que $M_e(\overline{x})=M'(\overline{e},\overline{x})$, igualdad que combinada con las anteriores nos permite obtener:

$$M_e(\overline{x}) = M'(\overline{e},\overline{x}) = \begin{cases} G(\overline{x}), & \text{si } e\in C_{\mathcal{F}}; \\ F(\overline{x}), & \text{si } e\notin C_{\mathcal{F}}. \end{cases} \tag{6}$$

Ahora podemos obtener

$$e\in C_{\mathcal{F}} \overset{(6)}{\Longrightarrow} M_e \text{ computa a } g \overset{g\notin\mathcal{F}}{\Longrightarrow} e\notin C_{\mathcal{F}}$$

y también

$$e\notin C_{\mathcal{F}} \overset{(6)}{\Longrightarrow} M_e \text{ computa a } f \overset{f\in\mathcal{F}}{\Longrightarrow} e\in C_{\mathcal{F}}.$$

Tenemos entonces que $e\in C_{\mathcal{F}} \Leftrightarrow e\notin C_{\mathcal{F}}$, lo cual es absurdo. Por lo tanto, $C_{\mathcal{F}}$ no es Turing-decidible.