Capítulo 1

Máquinas de Turing

En este capítulo se presentan las máquinas de Turing. Una máquina de Turing es más poderosas computacionalmente que un autómata de Pila. A primera vista una máquina de Turing se parece a los autómatas vistos en los capítulos anteriores. Al igual que un autómata de estados finitos, o uno de Pila, una máquina de Turing tiene un número finito de estados y una cinta de donde pueden leer. No tienn una pila, pero puede escribir sobre la misma cinta donde lee. Además puede moverse hacia adelante (derecha) o hacia atrás (izquierda) sobre la cinta. En cada instante, la máqiuna de Turing , dependiendo del estado en el que está y del símbolo que está leyendo, sobreescribe el símbolo leído, pasa a otro estado, y se mueve a la izquierda o a la derecha en la cinta. Estó último es lo que le agrega poder computacional a las máquinas de Turing.

1.1. Defincione̊s

La descripción formal de una Máquina de Turing se enuncia a continucación:

Definición 1. Una máquina de Turing es una cuádrupla: (Q, Σ, q_I, δ) donde:

- Q es un conjunto finito de estados,
- Σ es un alfabeto finito que incluye el símbolo # que representa el blanco. En todas las posiciones de la cinta hay siempre algo escrito, así sea un blanco.
- $q_I \in Q$ es el estado inicial
- $\delta : (Q \times \Sigma) \to (Q \cup \{\bigcirc\} \times (\Sigma \times (\to, \leftarrow))$ es una función (parcial) de transición.

La función de transición de estados sirve para indicar el comportamiento de la máquina al estar en un estado dado y leyendo un símbolo en la cinta. Esta es una función parcial. Esto quiere decir, que puede haber parejas (estado, símbolo) para las cuales la función no está definida.

El comportamiento de la máquina está dado por la función de transición:

- Si $\delta(q, \sigma) = (q_1, \rho, \to)$, esto quiere decir que que si está en un estado q, leyendo σ, escriba ρ, mueva la cabeza lectora a la derecha y pase al estado q_1.
- Si $\delta(q, \sigma) = (q_1, \rho, \leftarrow)$, esto quiere decir que que si está en un estado q, leyendo σ, escriba ρ, mueva la cabeza lectora a la izquierda y pase al estado q_1.
- Si $\delta(q, \sigma) = (\bigcirc, \rho, \to)$, esto quiere decir que que si está en un estado q, leyendo σ, escriba ρ, mueva la cabeza lectora a la derecha y detenga la ejecución.
- Si $\delta(q, \sigma) = (\bigcirc, \rho, \leftarrow)$, esto quiere decir que que si está en un estado q, leyendo σ, escriba ρ, mueva la cabeza lectora a la izquierda y detenga la ejecución.

La ejecución termina normalmente, si se ejecutan cualquiera de las dos últimas instrucciones descritas arriba. También puede detenerse si para algún estado y algún símbolo, no está especificado qué acción se debe tomar. Si se sale por el extremo izquierdo de la cinta terminaría en error.

1.1.1. Configuración

La configuración de una máquina de Turing está dada por el estado en que se encuentra, el contenido de la cinta de entrada, y la posición de la cabeza lectora sobre la cinta. Es, entonces, una tripla en: $(Q \cup \{\bigcirc\} \times \Sigma^* \times \mathbb{N})$. La cinta de entrada es infinita, al decir que el contenido es ω, estamos suponiendo que ω está al principio de la cinta y que después de ω sólo hay blancos. Podría haber blancos en cualquier parte de ω, inclusive al final. Lo importante es que después de ω aparezcan sólo blancos.

Suponga que: $\omega = \sigma_1 \dots \sigma_{n-1}\sigma_n$; estamos leyendo σ_i; y estamos en el estado q. Podemos representar la configuración así: $(q, \sigma_1 \dots \underline{\sigma_i} \dots \sigma_{n-1}\sigma_n)$, subrayando el símbolo que se está leyendo. Si se está en una configuración en que el estado es $\bigcirc$, se sabrá que el cálculo ha terminado sin problemas. Si estamos en una configuración: $(q, \sigma_1 \dots \underline{\sigma_i} \dots \sigma_{n-1}\sigma_n)$, y $\delta(q, \sigma_i)$ no está definido, la ejecución está detenida.

Las transiciones se describen a continuación:

- Si la máquina de Turing está en una configuración: $(q, \sigma_1 \dots \underline{\sigma_i} \dots \sigma_{n-1}\sigma_n)$ (con $i < n$), y si $\delta(q, \sigma i) = (q_1, (\rho, \rightarrow))$ entonces pasaría a la configuración: $(q_1, \sigma_1 \dots \rho\underline{\sigma_{i+1}} \dots \sigma_{n-1}\sigma_n)$.
- Si la máquina de Turing está en una configuración: $(q, \sigma_1 \dots \sigma_i \dots \sigma_{n-1}\underline{\sigma_n})$, y si $\delta(q, \sigma i) = (q_1, (\rho, \rightarrow)$ entonces pasaría a la configuración: $(q, \sigma_1 \dots \sigma_{n-1}\rho\underline{\#})$.
- Si la máquina de Turing está en una configuración: $(q, \sigma_1 \dots \underline{\sigma_i} \dots \sigma_{n-1}\sigma_n)$, (con $1 < i$), si $\delta(q, \sigma i) = (q_1, (\rho, \leftarrow)$ entonces pasaría a la configuración: $(q, \sigma_1 \dots \underline{\sigma_{i-1}}\rho \dots \sigma_{n-1}\sigma_n)$.
- Si la máquina de Turing está en una configuración: $(q, \underline{\sigma_1} \dots \sigma_n)$, si $\delta(q, \sigma i) = (q_1, (\rho, \leftarrow)$ entonces la ejecución terminaría abruptamente pues la cabeza lectora no puede moverse a lizquierda a partir de esa posición.

Ecribimos $C \Rightarrow K$, para indicar que se puede pasar en un paso de la configuración C a otra configuración K.

Definición 2. Decimos que C_n es alcanzable en cero o más pasos a partir de C_1, si:

- $C_1 = C_n$
- Existe una configuración C tal que $C_1 \Rightarrow C$ y $C \Rightarrow^* C_n$.

En este caso escribimos: $C_1 \Rightarrow^* C_n$.

1.2. Representación Gráfica

Graficamente, podemos describir una máquina de Turing con un multigrafo donde los nodos representan estados y los arcos las transiciones. Los arcos se etiquetan con $\sigma; \rho, A$, donde σ es lo que lee, ρ es lo que escribe encima de σ y A es la instrucción que puede ser $\Rightarrow$ o $\Leftarrow$.

Si tenemos que $\delta(q, \sigma) = (\rho, p, A)$, esto se dibujaría así:

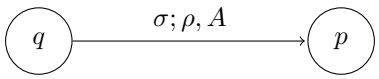

Figura 1.1: Turing Machine Transition

1.3. Extensiones a la definición

Es útil extender la definición de máquinas de Turing para que pueda mover la cabeza sin leer y que pueda leer sin avanzar o retroceder. También, sería útil poder hacer que ignore la cinta al leer o escribir. Esto no le agrega poder computacional pero puede hacer que sea más fácil describir ciertos cálculos.

1.3.1. Operación Nula

Comenzamos extendiendo las Máquinas de Turing, agregando la acción de no mover la cabeza lectora. Si hacemos esto, la función de transición tendría la siguiente forma:

$$\delta : (Q \times \Sigma) \to (Q \cup \{\bigcirc\} \times (\Sigma \times (\rightarrow, \leftarrow, \leftrightarrow))$$

Esto no agrega poder computacional. Podemos ver que si tenemos un máquina de Turing con la operación $\leftrightarrow$, es fácil construir una máquina de Turing con el mismo comporamiento, que no tiene esta operación.

Teorema 3. Toda máquina de Turing extendida con la operación $\leftrightarrow$ puede convertirse a una máquina de Turing equivalente que no tiene esta operación.

Considere una máquina de Turing con la operación $\leftrightarrow$

$$T = (Q, \Sigma, q_I, \delta)$$

Podemos construir una máquina de Turing equivalente que no contiene esta operación así:

$$T' = (Q', \Sigma, q_I, \delta')$$

Sea $Q' = Q \cup \{q_p : p \in Q\}$
Suponemos que $Q \cap Q' = \emptyset$
Definimos δ' así:

- Las transiciones que no tienen $\leftrightarrow$ no cambian: $\delta'(q, \sigma) = (p, \rho, op)$ si $\delta(q, \sigma) = (p, \rho, op)$ y $op \neq \leftrightarrow$
- Las transiciones tienen NOP, se cambian por una transición al estado q_p moviéndose a la derecha: $\delta'(q, \sigma) = (q_p, \rho, \rightarrow)$ si $\delta(q, \sigma) = (p, \rho, \leftrightarrow)$.
- Agregamos transiciones para cada q_p para que pase al estado p moviendose a la izquierda con cualquier símbolo y volviendo a escribir el mismo, para así no cambiar la cinta. $\delta'(q_p, \rho) = (p, \rho, \leftarrow)$ para cada $p \in Q$ y cada $\rho \in \Sigma$

1.3.2. Ignorar la cinta

Podemos también extender la máquina para ignorar la cinta. Es decir, para que ejecute la acción independientemente de lo que hay en la cinta y para que no escriba nada en la cinta.

LA función de transición entonces quedaría así:

$$\delta : (Q \times (\Sigma \cup \{\lambda\}) \to (Q \cup \{\bigcirc\} \times (((\Sigma \cup \{\lambda\}) \times (\rightarrow, \leftarrow, \leftrightarrow))$$

Donde
$\delta(q, \lambda) = \ldots$ indica que para cualquier símbolo del alfabeto se ejecute la acción.
$\delta(q, c) = (\lambda \ldots)$ que no se escribe nada y se deja tal como está.
Es fácil ver que estas transiciones.λ pyuden reemplazarse por una transición por cada símbolo del alfabeto para quedar con una máquina sin transiciones $|lambda$.

Sin embargo estas transiciones-λ podrían agregar no determinismo si tenemos una mezcla de transiciones-λ con transiciones normales saliendo de un mismo estado. Lo cual en sí no es problema, ya que las Máquinas de Turing no-determinísticas tienen el mismo poder computacional que las determinísticas, Sin ambargo, podría ser problemático para su simulación. Si queremos evitar esto se debe garantizar que:

- Si existe una transición que salga de un estado q, que se active al leer λ, no debe existir ninguna otra que se active con algun $\sigma \in \Sigma$
- Si existe una transición que salga de un estado q, y que escriba λ, no debe existir una que escriba otra cosa.

Teorema 4. Si tenemos una Máquina de Turing con transiciones que se ejecutan al leer λ (es decir, ignorando la cinta) y que permiten escribir λ (es decir, no escribir nada), podemos construir una máquina de Turing equivalente que no lee λ ni escribe λ.

Lo que queremos hacer, entonces, es cambiar las transiciones que tienen λ por transiciones equivalentes que no tienen λ.

- Cambiamos las transiciones que leen σ (un símbolo distinto de λ) y no escriben nada (transiciones de la forma: $\delta(q, \sigma) = (p, \lambda, op)$) por la transición: $\delta(q, \sigma) = (q, \sigma)$, una transición en la que se escribe lo mismo que se lee.
- Cambiamos las transiciones que ignora la cinta y escribe ρ (un símbolo distinto de λ) - transiciones de la forma: $\delta(q, \lambda) = (p, \rho, op)$ por transiciones: $\delta(q, \sigma) = (p, \rho, op)$ (una por cada $\sigma \in \Sigma$).
- Cambiamos las transiciones que ni leen y escriben (transiciones de la forma: $\delta(q, \lambda) = (p, \lambda, op)$) por transiciones: $\delta(q, \sigma) = (p, \sigma, op)$, una por cada $\sigma en \Sigma$

Formalmente: $\delta' = \delta$
$\{((q, \lambda), (p, \lambda, op))\} \cup \{((q, \sigma), (p, \sigma, op)) : \sigma \in \Sigma, \}$

1.3.3. Otras extensiones

Existen otras extensiones de Máquinas de Turing; ninguna de las cuales agrega poder computacional. Simplemente, agregan poder de expresión.

- Máquinas con una cinta infinita en ambas direcciones
- Máquinas no determinísiticas
- Máquinas con dos cintas
- Máquinas con más de dos cintas (pero el número de cintas debe ser predeterminado)

1.4. Combinando Máquinas

Se pueden combinar máquinas de Turing para crear máquinas más complejas.

1.5. Equivalencias

En esta sección veremos cómo simular los autómatas vistos en capítulos anteriores con máquinas de Turning.

1.5.1. Autómata Finito

Digamos que tenemos un autómata finito determinístico definido así:

$$M = (Q, \Sigma, q_I, F, \delta_M)$$

donde $\{\#, \boxtimes, \boxtimes\} \not\subset \Sigma$ y $\bigcirc \notin Q$. Entonces la máquina de Turing defininida así:

$$T = (Q \cup \{\bigcirc\}, \Sigma \cup \{\#, \boxtimes, \boxtimes\}, q_I, \delta_T)$$

donde:

- $\delta_T(q, \sigma) = (\delta(q, \sigma), (\sigma, \rightarrow))$ para $\sigma \in \Sigma$
- $\delta_T(q, \#) = (\bigcirc, \boxtimes)$ para $q \in F$
- $\delta_T(q, \#) = (\bigcirc, \boxtimes)$ para $q \notin F$

Simula el comportamiento del autómata M.
Sea $\omega = \sigma_1 \dots \sigma_n \boxtimes$, Si T comienza en una configuración $(q_I, \underline{\sigma_1} \dots \sigma_n)$:

- Si $\omega \in L(M)$, entonces la máquina de Turing terminaría en la siguiente configuración: $(\bigcirc, \sigma_1 \dots \sigma_n \underline{\boxtimes}$
- Si $\omega \in L(M)$, entonces la máquina de Turing terminaría en la siguiente configuración: $(\bigcirc, \sigma_1 \dots \sigma_n \underline{\boxtimes}$

1.5.2. Transductor

1.5.3. Autómata de Pila

Haremos la equivalencia para un autómata de pila simplificado.
Un autómata de pila simplificado es de la forma:

$$M = (Q, \Sigma, \Gamma, s, f, \Delta)$$

Donde:

$$\Delta \subseteq (Q \times (\Sigma \cup \lambda) \times (\Gamma \cup \lambda)) \times (Q \times (\Gamma \cup \lambda))$$

Y:

$$((q, a, b), (p, c)) \in \Delta \Rightarrow (b = \lambda \Leftrightarrow c \neq \lambda)$$

Recuerde que lo que indica esta regla es simplemente que las reglas sólo pueden empilar o desempliar un único símbolo. Sólo pueden ser *push* o *pop*. No puede ser *Skip*, *pushOn*, *changeTop*.

La máqina de Turning extendida correspondiente se construiría así:

Vamos a construir una máquina de Turing tal que para todo $\omega \in L(M)$ si la máquina comienza en una configuración $\underline{\#}\omega$ en el estado inicial terminará en un estado: $\beta \boxtimes \phi$ y para cada $\omega \notin L(M)$ si la máquina comienza en una configuración $\bigcirc, \underline{\#}\omega$ en el estado inicial terminará en un estado: $\bigcirc, \beta \boxtimes \phi$ o trancada por no poder seguir.

Comenzamos:

$$T = (Q_T, \Sigma_T, q_{IT}, \delta_T$$

Inicialmente:

- $Q_T = Q \cup \{q_{IT}\}$
- $\Sigma_T = \Sigma \cup \Gamma\{\$\}$ donde \$ es un símbolo que no está ni en Σ ni en *Gamma*.

Vamos a agregar unas reglas que le permiten a la máquina pasar de la configuración:
$(q_{IT}, \underline{\#}\omega)$ a la configuración: $(q_I, \omega\$)$
donde está parado en el primer símbolo de ω o en el blanco siguiente si ω es vacío.

Estas serían las reglas $\delta(q_{IT}, \#) = (q_{mark}, (\#, \Rightarrow))$ y para cada $\sigma \in \Sigma$ agregamos la regla $\delta(q_{mark}, \sigma) = (q_{mark}, (\sigma, \Rightarrow))$ y además la siguiente: $\delta(q_{mark}, \#) = (q_{markR}, (\$, \leftarrow))$ El estado $markR$ sirve para devolvernos a principio: $\delta(q_{markR}, \sigma) = (q_{markR}, (\sigma, \leftarrow))$ $\delta(q_{markR}, \#) = (q_I, (\#, \rightarrow))$

Vamos a agregar estados que nos servirán para empilar un valor en la pila y volver a donde íbamos leyendo.

$delta(q_{push_\rho}, \sigma) = (delta(q_{push_\rho}, (\sigma, \rightarrow)))$
$delta(q_{push_\rho}, \$) = ((p_{push_\rho}, \mu, \rightarrow)))$
$delta(q_{push_\rho}, \$) = ((p_{push_\rho}, \mu, \rightarrow)))$

1.6. Ejemplos

Es interesante ver lo poderosa que es la máquina de Turing. En esta sección, veremos cómo se pueden resolver problemas con este formalismo.

Ejemplo 1.1. Considere el lenguaje $\{a^n b^n c^n : n > 0\}$. Este es un lenguaje que no es independiente del contexto (ni regular). Para reconocer este lenguaje debemos leer una a y aseguranos de que haya tanto una b como una c por esa a.
Para facilidad, vamos a suponer que comenzamos en una configuración que tiene la siguiente forma:

$$\underline{(}\#)\omega$$

Donde

$$\omega \in \{a, b, c\}*$$

La cadena está en el lenguaje si:

$$\omega = a^n b^n c^n$$

Considere la siguiente máquina de Turing:
$M = (\{q_I, S, A, FindB\}, \{a, b, x, c, t\}, q_I, \delta)$
Donde:

- Empieza moviendose a la derecha $\delta(q_I, \#) = (S, (\rightarrow, \#))$
- En el estado S: si está en blanco, indica que la cadena era λ; si es a, arranca el proceso; si es culquier otro símbolo, la cadena no es aceptada:
  - $\delta(S, \#) = (\bigcirc, (\bigcirc, \boxtimes))$
  - $\delta(S, a) = (A, (\rightarrow, a))$
  - $\delta(S, \sigma) = (\bigcirc, (\bigcirc, \boxtimes))$ para $\sigma \notin \{a, \#\}$
- Si está en el estado A llegó a este estado al principio o al haber leido y marcado la primera a y la b correspondiente.
  - $\delta(A, a) = (FindB, (\rightarrow, \#))$
  - $\delta(A, x) = (B, (\bigcirc, x))$
  - $\delta(A, \sigma) = (\bigcirc, \boxtimes))$ para $\sigma \notin \{a, x\}$
- Si está en el estado $FindB$:
  - $\delta(FindB, b) = (RetA, (\leftarrow, x))$
  - $\delta(FindB, \#) = (\bigcirc, (NOP, \boxtimes))$
  - $\delta(FindB, \sigma) = (FindB, (\rightarrow, \sigma))$ para $\sigma \notin \{b, \#\}$
- Si está en el estado $RetA$:
  - $\delta(RetA, \#) = (A, (\leftarrow, \#))$
  - $\delta(RetA, \sigma) = (RetA, (NOP, \sigma))$ para $\sigma \neq \#$

Al terminar este proceso, si no se ha detenido el proceso por fallas, leimos una secuencia de a's, la seguida del mismo número de secuencia de b's, cambiando las a's por blancos y las b's por x's. Quedamos entonces en la siguiente configuración:

$$(B, \#^{n+1}\underline{(}x)x^{n-1}\omega_2)$$

Donde para que la cadena sea aceptable:

$$\omega_2 = c^n$$

Tenemos, entonces las siguientes transiciones. En este caso se marcarán las c's de derecha a izquierda con blancos!

- Si está en el estado B al haber buscado una d[7] y encontrado una x o devolviéndose de marcar la c correspondiente.
  - $\delta(B, x) = (FindC, (\rightarrow, \#))$
  - $\delta(B, \#) = (\bigcirc, (NOP, \boxtimes))$
  - $\delta(B, \sigma) = (\bigcirc, (NOP, \boxtimes))$ para $\sigma \notin \{x, \#\}$

Ejemplo 1.2. Al igual que en los autómatas podemos definir las máquinas usando fórmulas en lugar de dibujos.
Suponga que queremos definir una máquina de Turing que comienza en el extremo izquierdo de la cinta donde aparece una cadena de dígitos y queremos escribir el resultado de dividir el número que aparece en la cinta por cuatro (ver el ejemplo del trasductor del capítulo 1), esta máquina se describe así:

- $Q = 0, 1, 2, 3, fin$
- $\Sigma = 0, 1, 2, 3, R, \#$
- $q_I = 0$
- La función de transición la escribimos así:
  - $\delta(s, d) = ((10 \cdot s) + d) mod 4, (((10 \cdot s) + d) \div 4), \rightarrow), \ d \in 1, 2, 3, 4$
  - $\delta(0, \#) = (\bigcirc, (\#, NOP))$
  - $\delta(d, \#) = (d, (R, \Rightarrow))$
  - $\delta(d, R) = (\bigcirc, (d, \Rightarrow))$

1.7. Máquinas Sencillas

1.7.1. Moverse a la derecha

$$R = (\{q\}, \Sigma, q, \delta_R)$$

Donde

$$\delta_R(q, \lambda) = (\bigcirc, \lambda, \Rightarrow)$$

1.7.2. Moverse a la izquierda

$$L = (\{q\}, \Sigma, q, \delta_L)$$

Donde

$$\delta_L(q, \lambda) = (\bigcirc, \lambda, \Leftarrow)$$

1.7.3. Escribir un símbolo: σ

$$W_\sigma = (\{q\}, \Sigma, q, \delta_{W_\sigma})$$

Donde

$$\delta_{W_\sigma}(q, \lambda) = (\bigcirc, \sigma, NOP)$$

1.7.4. Moverse a la derecha hasta encontrar un símbolo dado, σ

Esta máquina hace las siguiente secuencia de operaciones:

1. Se mueve a la derecha
2. Si está leyendo el símbolo σ, para de lo contrario vuelve al paso 1

$$R_{=\sigma} = (\{q, p\}, \Sigma, q, \delta_R)$$

Donde

$$\delta_{R_{=\sigma}}(q, \lambda) = (p, \lambda, R)$$

$$\delta_{R_{=\sigma}}(p, \sigma) = (\bigcirc, \sigma, NOP)$$

Para $\rho \neq \sigma$:

$$\delta_{R_{=\sigma}}(p, \rho) = (p, \rho, \Rightarrow)$$

1.7.5. Moverse a la izquierda hasta encontrar un símbolo, σ

Esta máquina hace las siguiente secuencia de operaciones:

1. Se mueve a la izquierda
2. Si está leyendo el símbolo σ, para de lo contrario vuelve al paso 1

$$L_{=\sigma} = (\{q, p\}, \Sigma, q, \delta_L)$$

Donde

$$\delta_{L_{=\sigma}}(q, \lambda) = (p, \lambda, R)$$

$$\delta_{L_{=\sigma}}(p, \sigma) = (\bigcirc, \sigma, NOP)$$

Para $\rho \neq \sigma$:

$$\delta_{L_{=\sigma}}(p, \rho) = (p, \rho, \Leftarrow)$$

1.7.6. Moverse a la derecha hasta no estar leyendo un símbolo dado, σ

Esta máquina hace las siguiente secuencia de operaciones:

1. Se mueve a la derecha
2. Si está no está leyendo el símbolo σ, para de lo contrario vuelve al paso 1

$$R_{\neq\sigma} = (\{q, p\}, \Sigma, q, \delta_{R_{\neq\sigma}})$$

Donde

$$\delta_{R_{\neq\sigma}}(q, \lambda) = (p, \lambda, R)$$

$$\delta_{R_{\neq\sigma}}(p, \sigma) = (p, \sigma, NOP)$$

Para $\rho \neq \sigma$:

$$\delta_{R_{\neq\sigma}}(p, \rho) = (\bigcirc, \rho, \Rightarrow)$$

1.7.7. Moverse a la derecha hasta no estar leyendo un símbolo dado, σ

Esta máquina hace las siguiente secuencia de operaciones:

1. Se mueve a la derecha
2. Si está no está leyendo el símbolo σ, para de lo contrario vuelve al paso 1

$$R_{\neq\sigma} = (\{q, p\}, \Sigma, q, \delta_{R_{\neq\sigma}})$$

Donde

$$\delta_{R_{\neq\sigma}}(q, \lambda) = (p, \lambda, R)$$

$$\delta_{R_{\neq\sigma}}(p, \sigma) = (p, \sigma, NOP)$$

Para $\rho \neq \sigma$:

$$\delta_{R_{\neq\sigma}}(p, \rho) = (\bigcirc, \rho, \Rightarrow)$$

1.8. Cambiando la notación

La notación para describir máquinas de Turing puede resultar un poco engorrosa. En los ejemplos mostrados arriba, veíamos que acciones como buscar un símbolo, requieren muchos estados. Muchos autores, introducen una notación en la que priman las operaciones. Se tienen algunas operaciones básicas, e inclusive se pueden definir máquinas más complejas y conectarlas. En este caso, los nodos son operaciones y los arcos se etiquetan con símbolos que permiten el paso de una operación a otra.

Las operaciones básicas se describen a continuación.

- NOP
- Moverse a la derecha sin escribir. Es lo mismo que tener la siguiente transición.
- Moverse a la izquierda sin escribir
- Moverse a la derecha hasta encontrar un símbolo dado. Note que se mueve a la derecha. Si es el símbolo dado entonces ya termina. se vuelve a mover a la derecha.
- Moverse a la derecha hasta que no esté leyendo un símbolo dado.
- Moverse a la izquierda hasta encontrar un símbolo dado
- Moverse a la izquierda hasta que no esté leyendo un símbolo dado
- Escribir un símbolo

Estas corresponden a las máquinas descritas en la sección anterior. Podemos, además extender el formalismo para abreviar máquinas. Por ejemplo, la siguiente máquina copia el primer símbolo al final de la cadena. Suponiendo que el alfabeto es $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Con el formalismo de operaciones lo escribiríamos así:

Con el sistema de abreviación así:

Si además lo escribimos así, nos sirve para cualquier símbolo que no sea blanco. Acá se supone que el alfabeto son todos los caracteres imprimibles.

Ejemplo 1.3. Se muestra la definición de una máquina de Turing que corre a la izquierda la cadena a la izquierda de la cabeza lectora, borrando el primer símbolo. Es decir, si comienza en esta configuración:

$$\underline{\#}\sigma_1 \dots \sigma_n$$

terminaría en esta.

$$\underline{\#}\sigma_2 \dots \sigma_n$$

Podría haber algo a la izquierda del primer #, y no afectaría para nada su comportamiento.

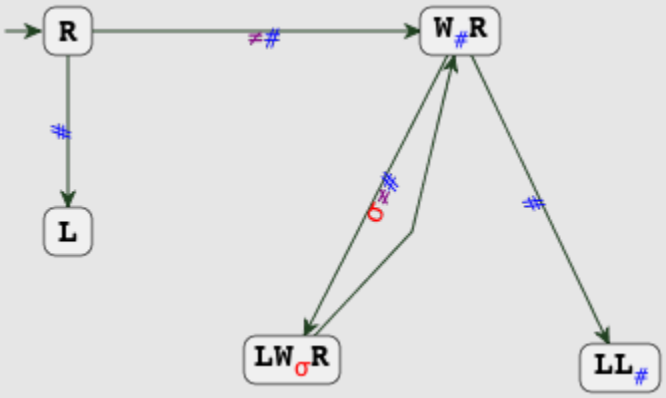

Figura 1.2: Una máquina para correr a la izquierda (llamada SL)

1.9. Combinación de máquinas

Las máquinas de Turing pueden combinarse. Para esto agregamos una nueva operación que es llamar una máquina. Si la máquina invocada termina la ejecución con la operación STOP, la ejecución termina. De lo contrario, ejecuta hasta que no haya transición aplicable y luego se retorna el control a la máquina invocante.

A continuación mostramos unos ejemplos que usan el llamado a otras máquinas para lograr un objetivo.

Ejemplo 1.4. Definimos una máquina para borrar el símbolo que está siendo leído por la cabeza lectora, terminando en la misma posición. La cabeza lectora NO puede estar en la primera posición de la cinta.
si comienza en esta configuración:

$$\sigma_0 \underline{\sigma_1} \sigma_2 \dots \sigma_n$$

terminaría en esta.

$$\sigma_0 \underline{\sigma_2} \dots \sigma_n$$

Esto se logra así:

- Se mueve a la izquierda
- Si es un blanco, simplemente, llama a la máquina del Ejemplo 1.2 y se vuelve a mover a la derecha.
- Si es un símbolo, α que no es blanco, escribe un blanco, llama a la máquina del Ejemplo 1.2, escribe α y se vuelve a mover a la derecha.

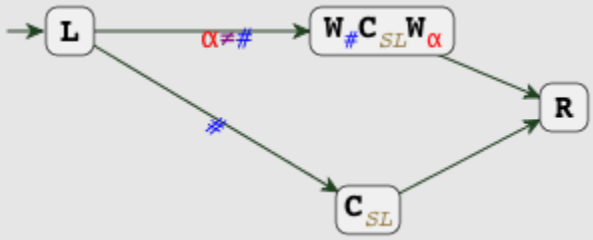

Figura 1.3: Una máquina para eliminar el símbolo actual (llamada Del)

Ejemplo 1.5. Este ejemplo muestra una máquina que comienza en una secuencia de a's y quita la mitad. Si la cantidad de a's es impar deja $(n \div 2)$, seguidas por una x.
Si comienza

$$\underline{\#}\sigma_1 \dots \sigma_{2n}$$

terminaría en esta.

$$\#\sigma_1 \dots \underline{\sigma_n}$$

Si comienza

$$\underline{\#}\sigma_1 \dots \sigma_{2n+1}$$

terminaría en esta.

$$\#\sigma_1 \dots \sigma_n \underline{x}$$

La idea es la siguiente:

1. Se mueve a la derecha, debe haber una `a`
2. Se vuelve a mover a la derecha,
3. si hay un expacio es por que había un número impar de `a`'s (en el caso inicial) una sóla `a`. Se mueve a la izquierda, escribe `x`
4. Si hay una `a`, la elimina con la máquina `DEL`
5. Si al eliminar la segunda `a` queda en una `a` vuelve al paso 3

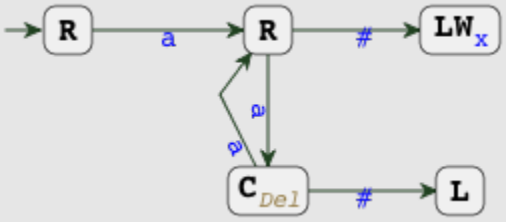

Figura 1.4: Dividir por 2

Finalmente, vamos a usar esta última máquina para verificar, que se está leyendo una cadena de a's donde el número de a's es una potencia de 2. La idea es dividir por 2 hasta llegar a leer x. Si x es lo único que hay en la cinta, es que se llegó a 1, que sí es potencia de dos!

Ejemplo 1.6. La idea acá es dividir por 2, hasta llegar a una división que deja una x al final. Si esa x es el único símbolo es porque se dividió a por 2 dando como resultado x. En este caso escribe Y, aceptando la cadena. En caso contrario, escribe N.

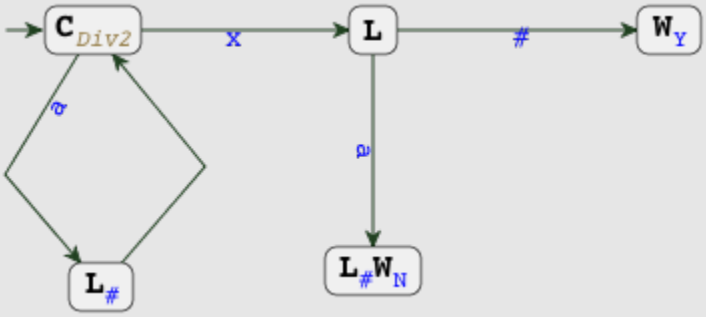

Figura 1.5: Una máquina para verificar que el número de a's es potencia de 2