

del contexto en la cual el lado derecho de cada regla de reescritura consiste en un terminal seguido por cero o más no terminales (se dice que estas gramáticas tienen la forma normal de Greibach).

36. Muestre que la intersección de un lenguaje regular y un lenguaje independiente del contexto siempre es independiente del contexto.
37. Encuentre una gramática independiente del contexto para el lenguaje de $\{x, y\}$ que consiste en las cadenas en las cuales la relación entre el número de x y el número de y es de tres a dos.

Problemas de programación

1. Realice el algoritmo de análisis sintáctico $LL(1)$ que se muestra en la figura 2.30. Aplique los resultados a las tablas de análisis sintáctico de las figuras 2.29 y 2.31, así como a su respuesta al problema 13 de repaso del capítulo.
2. Realice el algoritmo de análisis sintáctico $LR(1)$ que se muestra en la figura 2.35. Aplique los resultados a las tablas de análisis sintáctico de las figuras 2.34 y 2.38 así como a su respuesta al problema 15 de repaso del capítulo.
3. Escriba un programa para convertir gramáticas independientes del contexto que no generan la cadena vacía en gramáticas con la forma normal de Chomsky.

CAPÍTULO 3

Máquinas de Turing y lenguajes estructurados por frases

- 3.1 **Máquinas de Turing**
Propiedades básicas de las máquinas de Turing
Los orígenes de las máquinas de Turing
- 3.2 **Construcción modular de máquinas de Turing**
Combinación de máquinas de Turing
Bloques de construcción básicos
- 3.3 **Máquinas de Turing como aceptadores de lenguajes**
Procedimientos de evaluación de cadenas
Máquinas de Turing de varias cintas
Máquinas de Turing no deterministas
- 3.4 **Lenguajes aceptados por máquinas de Turing**
Comparación entre lenguajes aceptados por máquinas de Turing y lenguajes estructurados por frases
Alcance de los lenguajes estructurados por frases
- 3.5 **Más allá de los lenguajes estructurados por frases**
Sistema de codificación de máquinas de Turing
Un lenguaje no estructurado por frases
Máquinas de Turing universales
Comparación entre lenguajes aceptables y decidibles
El problema de la parada
- 3.6 **Comentarios finales**

Hasta ahora hemos presentado las clases de autómatas finitos y de pila. En ambos casos nuestro objetivo ha sido comprender el potencial de reconocimiento de lenguajes de estas máquinas teóricas. En este capítulo presentaremos otra clase de autómatas, todavía más generales, conocidos como máquinas de Turing, y estudiaremos el poder computacional de

estas máquinas en el contexto de la solución de problemas de reconocimiento de lenguajes.

Uno de los principales resultados será que las máquinas de Turing son capaces de aceptar exactamente los mismos lenguajes que pueden generar las gramáticas menos restrictivas: las gramáticas estructuradas por frases y así, el poder de procesamiento de lenguajes de esta clase de máquinas está limitado por las mismas fronteras que delimitan los poderes generativos de las gramáticas. La importancia de esta equivalencia aumenta por el hecho de que nadie ha podido definir una clase de máquinas computacionales más poderosas que las máquinas de Turing. De hecho, aprenderemos que en la actualidad los científicos de la computación aceptan de manera general la conjetura de que la clase de máquinas de Turing encierra el poder de cualquier proceso computacional (esta conjetura se conoce como tesis de Turing). Por esto, se cree que la correspondencia entre el potencial de reconocimiento de lenguajes de las máquinas de Turing y el poder generativo de las gramáticas que veremos en este capítulo define los límites máximos de cualquier sistema computacional de reconocimiento de lenguajes.

En resumen, la presentación de las máquinas de Turing en este capítulo servirá como culminación de nuestro estudio de las gramáticas, los lenguajes y los algoritmos de análisis sintáctico. Empero, esto no quiere decir que habrá concluido todo nuestro estudio; más bien, las limitaciones aparentes de los poderes computacionales que descubriremos en este capítulo serán el punto de partida para el estudio de la computabilidad.

3.1 MÁQUINAS DE TURING

La clase de autómatas que ahora se conoce como máquinas de Turing fue propuesta por Alan M. Turing en 1936. (La idea básica de Turing fue estudiar los procesos algorítmicos utilizando un modelo computacional. En ese mismo año, Emil L. Post presentó un enfoque similar, y más tarde se mostró que ambas estrategias son equivalentes en cuanto a poder computacional.) Para nuestros fines es conveniente considerar a las máquinas de Turing como una versión generalizada de los autómatas que hemos visto en capítulos anteriores. Las máquinas de Turing se asemejan a los autómatas finitos en que constan de un mecanismo de control y un flujo de entrada que concebimos como una cinta; la diferencia es que las máquinas de Turing pueden mover sus cabezas de lectura hacia adelante y hacia atrás y pueden leer o escribir en la cinta. Veremos que estas características aumentan en gran medida la capacidad de las máquinas.

Propiedades básicas de las máquinas de Turing

Al igual que las demás máquinas que hemos estudiado, la máquina de Turing contiene un mecanismo de control que en cualquier momento puede encontrarse en uno de entre un número finito de estados. Uno de estos estados

se denomina estado inicial y representa el estado en el cual la máquina comienza los cálculos. Otro de los estados se conoce como estado de parada; una vez que la máquina llega a ese estado, terminan todos los cálculos. De esta manera, el estado de parada de una máquina de Turing difiere de los estados de aceptación de los autómatas finitos y de pila en que éstos pueden continuar sus cálculos después de llegar a un estado de aceptación, mientras que una máquina de Turing debe detenerse en el momento en que llegue a su estado de parada. (Nota: con base en la definición anterior, el estado inicial de una máquina de Turing no puede ser a la vez el estado de parada; por lo tanto, toda máquina de Turing debe tener cuando menos dos estados).

Una diferencia más importante entre una máquina de Turing y los autómatas de los capítulos anteriores es que la máquina de Turing puede leer y escribir en su medio de entrada. Para ser más precisos, la máquina de Turing está equipada con una cabeza que puede emplearse para leer y escribir símbolos en la cinta de la máquina (como sucede con los otros autómatas, esta cinta tiene un extremo izquierdo pero se extiende indefinidamente hacia la derecha). Así, una máquina de Turing puede emplear su cinta como almacenamiento auxiliar, de la misma forma en que otra clase de autómatas utiliza su pila. Sin embargo, con este almacenamiento, una máquina de Turing no se limita a las operaciones de inserción y extracción, sino que puede rastrear los datos de la cinta y modificar las celdas que deseé sin alterar las demás.

Al utilizar la cinta para fines de almacenamiento auxiliar, es conveniente que una máquina de Turing emplee marcas especiales para distinguir porciones de la cinta. Para esto, permitimos que una máquina de Turing lea y escriba símbolos que no aparecen en los datos de entrada; es decir, hacemos una distinción entre el conjunto (finito) de símbolos, llamado alfabeto de la máquina, en el que deben estar codificados los datos de entrada iniciales, y un conjunto, posiblemente mayor (también finito), de símbolos de cinta, que la máquina puede leer y escribir (esta distinción es similar a la que se establece entre el alfabeto de un autómata y sus símbolos de pila). De esta manera, los símbolos de cinta de una máquina de Turing pueden incluir marcas especiales que no sean símbolos del alfabeto de la máquina.

El espacio en blanco es un símbolo que cae en esta categoría; se trata de un símbolo que se supone está en cualquier celda de la cinta que no esté ocupada. Por ejemplo, si una máquina de Turing leyera más allá de los símbolos de entrada de su cinta, encontraría y leería las celdas en blanco que allí se encuentran. Además, puede ser necesario que una máquina de Turing tenga que borrar una celda, escribiendo en ella un espacio en blanco. Por esto, con frecuencia se considera que el espacio en blanco pertenece al conjunto de símbolos de cinta de la máquina de Turing, pero no forma parte del alfabeto de la máquina.

Es fácil que el símbolo de espacio en blanco ocasione confusiones y malas interpretaciones en una página impresa. Por cuestiones de comunicación, adoptamos el símbolo Δ para representar el espacio en blanco. Esta convención elimina la ambigüedad entre las cadenas $x\Delta y$ y $x y \Delta$ que podemos expresar la primera como $x\Delta y$, y la segunda, como $x \Delta \Delta y$.

Las acciones específicas que puede realizar una máquina de Turing consisten en operaciones de escritura y de movimiento. La operación de escritura consiste en reemplazar un símbolo en la cinta con otro símbolo y luego cambiar a un nuevo estado (el cual puede ser el mismo donde se encontraba antes). La operación de movimiento comprende mover la cabeza una celda a la derecha o a la izquierda y luego pasar a un nuevo estado (que, una vez más, puede ser igual al de partida). La acción que se ejecutará en un momento determinado dependerá del símbolo (el símbolo actual) que está en la celda visible en ese momento para la cabeza (la celda actual), así como del estado actual del mecanismo de control de la máquina.

Si representamos con Γ el conjunto de símbolos de cinta de una máquina de Turing, con S , el conjunto de estados y con S' , el conjunto de estados que no son el de parada, entonces es posible representar las transiciones de la máquina por medio de una función, llamada función de transición de la máquina, de la forma $\delta: (S' \times \Gamma) \rightarrow S \times (\Gamma \cup \{L, R\})$, donde suponemos que los símbolos L y R no pertenecen a Γ .

La semántica de esta representación funcional es la siguiente:

- $\delta(p, x) = (q, y)$ significa "si el estado actual es p y el símbolo actual es x , reemplazar la x con el símbolo y y pasar al estado q ".
- $\delta(p, x) = (q, L)$ significa "si el estado actual es p y el símbolo actual es x , mover la cabeza una celda a la izquierda y pasar al estado q ".
- $\delta(p, x) = (q, R)$ significa "si el estado actual es p y el símbolo actual es x , mover la cabeza una celda a la derecha y pasar al estado q ".

Observe que, al describir con una función las transiciones de una máquina de Turing, la máquina es determinista. Para ser más precisos, existe una, y sólo una, transición asociada a cada par estado-símbolo donde el estado no es el de detención.

Durante la operación normal, una máquina de Turing ejecuta transiciones repetidamente hasta llegar al estado de parada (esto quiere decir que en ciertas condiciones es posible que nunca se detengan los cálculos de una máquina de Turing, ya que su programa interno puede quedar atrapado en un ciclo sin fin). Existe, sin embargo, una anomalía que puede ocurrir durante este proceso: la cabeza de la máquina puede "rebasar" el extremo izquierdo de la cinta. En este caso, la máquina abandonará los cálculos y decimos que la ejecución de la máquina sufrió una terminación anormal.

Como sucede con otros autómatas, es útil representar visualmente una máquina de Turing, como se hace en la figura 3.1. Allí, el mecanismo de control de la máquina está representado por un rectángulo con una especie de carátula de reloj que indica el estado actual de la máquina. Encima de este rectángulo se encuentra la cinta de la máquina; la posición de la cabeza de la máquina está indicada con un apuntador, como se hizo en los autómatas finitos y de pila.

Como también sucede con los otros autómatas, la colección de transiciones de una máquina de Turing se puede representar de manera conveniente por medio de un diagrama de transiciones en el cual se ilustran con pequeños

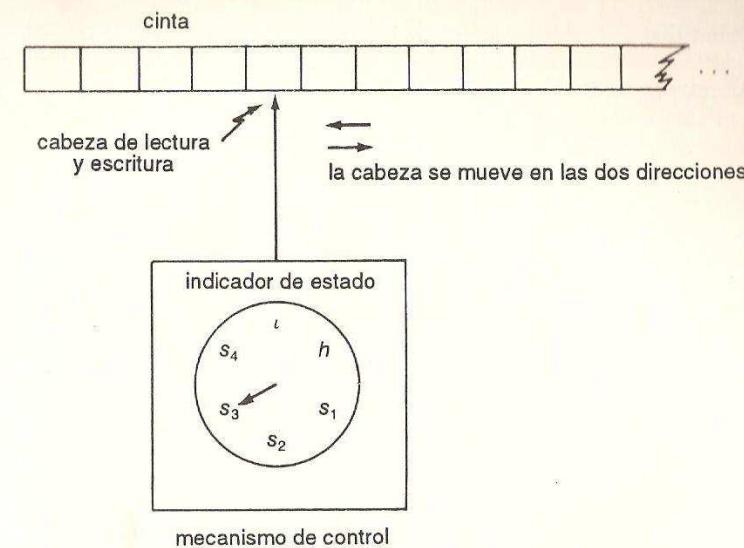


Figura 3.1 Representación de una máquina de Turing

círculos los estados de la máquina (identificando los estados inicial y de parada con un apuntador y un doble círculo respectivamente) conectados por arcos que representan las transiciones posibles. Cada uno de los arcos del diagrama de transiciones de una máquina de Turing se etiqueta con un par de símbolos separados por una diagonal. El primer símbolo del par representa el símbolo de la cinta que debe existir en la celda actual para que la transición sea aplicable; el segundo símbolo es el que se escribirá en la celda actual (si la transición es una operación de escritura) o de lo contrario uno de los símbolos, L o R (si la transición es una operación de movimiento). De esta manera, la transición $\delta(p, x) = (q, y)$ estaría representada por un arco de p a q con etiqueta x/y ; o $\delta(p, x) = (q, L)$ aparecería como un arco de p a q con etiqueta x/L . Como muestra, la figura 3.2 es un diagrama de transiciones completo para una máquina de Turing con símbolos de cinta $\{a, b, \Delta\}$ que mueve su cabeza hacia la derecha hasta encontrar un espacio en blanco.

Con base en nuestras experiencias con otros autómatas, no debe sorprender que muchas veces resulta difícil manejar y comprender un diagrama de transiciones completo para una máquina de Turing. Por esto, en ocasiones dibujamos esqueletos de los diagramas, que sólo contienen los arcos pertinentes para el análisis, dándose por sentado que, de ser necesario, podrían añadirse los demás arcos.

Como pronto veremos, la "máquina" computacional original de Turing era una computación humana con lápiz y papel. Con la llegada de los equipos de cálculo electrónico, este modelo dio lugar al concepto de máquina electromecánica que lee y escribe en una cinta magnética. Lo importante es que

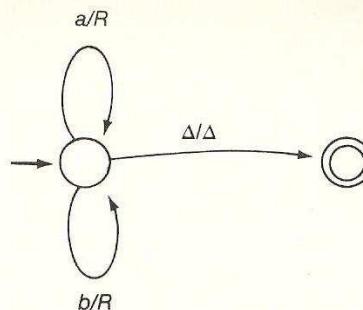


Figura 3.2 Diagrama de transiciones para una máquina de Turing

las capacidades computacionales teóricas del sistema son las mismas, sin importar la tecnología con la cual se implanten sus componentes. De hecho, el concepto de máquina de Turing, al igual que los otros autómatas que hemos estudiado, es independiente de su implantación. Entonces, aislaremos las características definitivas de una máquina de Turing en la siguiente definición formal.

- Una máquina de Turing es una sextupla de la forma $(S, \Sigma, \Gamma, \delta, i, h)$, donde:
- S es una colección finita de estados.
 - Σ es un conjunto finito de símbolos distintos de espacio en blanco, llamado alfabeto de la máquina.
 - Γ es un conjunto finito de símbolos, incluidos los de Σ , que se conocen como símbolos de la cinta de la máquina.
 - δ es la función de transición de la máquina.
 - i es un elemento de S llamado estado inicial.
 - h es un elemento de S llamado estado de parada.

En ocasiones es conveniente contar con una notación concisa que represente la configuración de la cinta de una máquina de Turing, incluyendo el contenido de sus celdas y la posición de la cabeza. En estos casos presentaremos la lista del contenido de las celdas de la cinta, subrayando la posición de la cabeza. De esta manera, $\Delta\bar{x}\bar{y}\bar{z}\Delta\Delta\dots$ representará una cinta que contiene un espacio en blanco, seguido por los símbolos x, y y z , seguidos a su vez por espacios en blanco, con la cabeza sobre la celda que contiene la z .

Los orígenes de las máquinas de Turing

El propósito para el cual se desarrollaron las máquinas de Turing es distinto del de los otros autómatas que hemos estudiado, ya que se diseñaron para contener todo el poder de los procesos computacionales. En otras palabras, la

intención de Turing fue desarrollar un sistema en el cual fuera posible modelar cualquier proceso que pudiera considerarse como un cálculo.

Recuerde que esto sucedió mucho antes del desarrollo de la maquinaria computacional que existe en la actualidad. Turing pensó en un cálculo realizado por un ser humano con lápiz y papel. Bajo este contexto, Turing razonó que, en un momento dado las personas sólo podrían concentrarse en una porción restringida del papel y que, a su vez, la colección de marcas en este trozo de papel se podía considerar como un solo símbolo. Por esto, Turing consideró dividir el papel en secciones, cada una de las cuales constituía la cantidad de papel requerida para registrar un solo símbolo. Turing también llegó a la conclusión de que cualquier proceso computacional sólo podía implicar un número finito de símbolos. De hecho, puesto que debe ser posible registrar cada símbolo en una cantidad fija de papel, la existencia de una cantidad cada vez mayor de símbolos dictaminaría que los símbolos deberían tener características distintivas arbitrariamente pequeñas. Entonces, se llegaría a un punto donde comenzaría a fallar la capacidad humana para distinguir los símbolos, lo que daría como resultado un número limitado de símbolos efectivos.

Turing planteó que al considerar una sección específica del papel, una persona podría alterar dicha sección o pasar a otra. La acción por emprender y sus detalles dependerían del símbolo existente en la sección y del estado de la mente de la persona. Turing concluyó que, al igual que con el número de símbolos, los seres humanos poseían, sólo una cantidad finita de estados de la mente distinguibles. Turing consideró también que la persona se hallaría en un estado especial inicial al comenzar los cálculos y en un estado designado como de parada al completar los cálculos.

Para evitar que la disponibilidad de papel restringiera el poder del modelo, Turing propuso que la cantidad de papel disponible para los cálculos fuera limitada.

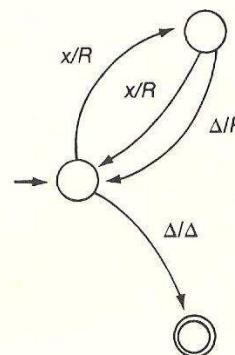
Es muy significativo que nadie haya podido producir un modelo computacional ampliamente aceptado que supere el poder del modelo de Turing. Este modelo es incluso más general que los computadores actuales, ya que una máquina de Turing nunca se ve restringida por la carencia de espacio de almacenamiento, algo que finalmente debe ocurrir en una máquina real. Por esto, la mayoría de los científicos de la computación aceptan la tesis de Turing: *el poder computacional de una máquina de Turing es tan grande como el de cualquier sistema computacional posible*. Además, esta tesis y los resultados que la apoyan hoy en día tienen importantes implicaciones con respecto a las dudas sobre la computabilidad. Para resolver cualquier problema con un computador, se requiere desarrollar un proceso computacional (o un algoritmo) que resuelva el problema. Entonces, las respuestas a las preguntas que se planteaban en la época de Turing, como “¿qué puede hacerse con un proceso computacional?” nos ayudan a responder las preguntas actuales, como “¿qué puede hacer un computador moderno?”.

En el capítulo 4 volveremos a hablar de la función de las máquinas de Turing en el estudio de los procesos computacionales, además de investigar

la relación entre estas nociones abstractas y el diseño de lenguajes de programación. Sin embargo, en este capítulo continuaremos con la presentación de una técnica útil para construir máquinas de Turing y luego veremos la relación entre máquinas de Turing, gramáticas y lenguajes.

Ejercicios

1. ¿Con qué configuración de cinta se detendrá la máquina de Turing que se presenta a continuación si inicia con la cinta configurada como $\underline{x}x\Delta\Delta\Delta\dots$?



2. Diseñe una máquina de Turing con símbolos de cinta x , y y Δ que busque en la cinta el patrón $xyxy$ y se detenga si y sólo si encuentra ese patrón.
3. Diseñe una máquina de Turing que, al iniciar con la cabeza sobre la celda del extremo izquierdo de la cinta, tenga una terminación anormal si y sólo si hay una x registrada en algún lugar de la cinta. Si aplicara su máquina a una cinta que no contiene una x , ¿detectaría la máquina esta situación?
4. Muestre que las capacidades computacionales de una máquina de Turing no aumentarían si se permitiera que tuvieran más de un estado de parada, ya que esta máquina se podría simular con una máquina de Turing con un solo estado de parada.

3.2 CONSTRUCCIÓN MODULAR DE MÁQUINAS DE TURING

Aunque el propósito principal de este capítulo es estudiar la capacidad de aceptación de lenguajes por parte de las máquinas de Turing, un efecto

secundario importante es la presentación de los fundamentos de las máquinas de Turing, ya que estas máquinas servirán como herramientas en los capítulos subsecuentes. Por esto investigaremos las máquinas de Turing con mayor detenimiento antes de continuar con nuestro estudio de los lenguajes. En esta sección, nuestro objetivo es desarrollar técnicas por medio de las cuales puedan construirse máquinas de Turing complejas a partir de bloques elementales. Este enfoque facilitará la construcción y la comprensión de las máquinas que veremos más adelante.

Combinación de máquinas de Turing

Aunque hablaremos en términos de la combinación de máquinas de Turing para formar otras máquinas de Turing mayores, nuestra estrategia será en realidad combinar los programas de las máquinas de Turing en una forma muy semejante a como se combinan módulos de programas para desarrollar grandes sistemas de software. Para esto, es útil representar los programas de las máquinas de Turing por medio de diagramas de transiciones y combinarlos de manera parecida a lo que hicimos para formar la unión y la concatenación de autómatas finitos.

Suponga que tenemos dos máquinas de Turing, M_1 y M_2 , con diagramas de transiciones T_1 y T_2 , respectivamente, y símbolos de cinta del conjunto Γ . Si queremos desarrollar un diagrama de transiciones para otra máquina que simule las actividades de M_1 seguidas por las de M_2 , bastaría con eliminar la designación de parada del estado de parada de T_1 y la característica de inicio del estado inicial de T_2 , y luego dibujar una arco con etiqueta x/x para cada x en Γ , del antiguo estado de parada de T_1 al antiguo estado inicial de T_2 .

Obviamente, el resultado se comportaría de la manera esperada, pero este sencillo enfoque tiene varias desventajas. Por ejemplo, en potencia podría introducirse un gran número de transiciones que fueran simples reescrituras del contenido de la celda (esta deficiencia presentaría obstáculos para nuestro análisis de la complejidad en el capítulo 5). Además, suponga que queremos que la máquina compuesta se detenga después de simular M_1 , a menos que el símbolo actual, al llegar al estado de parada, sea z , en cuyo caso nos gustaría que la nueva máquina simulara las acciones de M_2 ; o suponga que necesitamos combinar tres diagramas y controlar el paso de uno a otro dependiendo del valor del símbolo actual al llegar a cada uno de los antiguos estados de parada. En estos casos se requiere un enlace un poco más complicado.

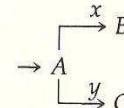
Suponga entonces que necesitamos combinar los diagramas de transiciones de varias máquinas de Turing para obtener una máquina que simule alguna combinación de las máquinas originales. Lo hacemos de la siguiente manera:

1. Elimine la característica de inicio de los estados iniciales de todas las máquinas, excepto la de aquél donde iniciará la máquina compuesta.

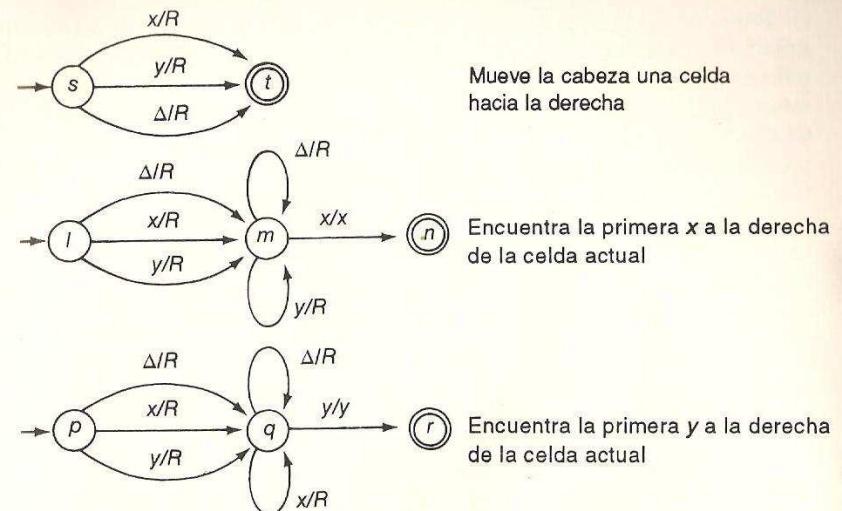
2. Elimine la característica de detención de los estados de parada de todas las máquinas e introduzca un nuevo estado de parada que no se encuentre en ninguno de los diagramas que se combinan.
 3. Para cada uno de los antiguos estados de parada p y cada $x \in \Gamma$, dibuje un arco de la siguiente forma:
 - a. Si la máquina compuesta debe detenerse al llegar a p con el símbolo actual x , dibuje un arco con etiqueta x/x de p al nuevo estado de parada.
 - b. Si al llegar al estado p con el símbolo actual x , la máquina compuesta debe transferir el control a la máquina $M = (S, \Sigma, \Gamma, \delta, i, h)$, dibuje entonces un arco con etiqueta x/z de p al estado q de M , donde $\delta(i, x) = (q, z)$.

La figura 3.3 proporciona un ejemplo de esta construcción. Aquí hemos comenzado con los diagramas de transiciones para tres máquinas distintas, cada una con los símbolos de cinta x , y y Δ . Una mueve su cabeza una celda a la derecha, otra encuentra la primera x a la derecha de la celda actual, y la tercera encuentra la primera y a la derecha de la celda actual. Con el proceso de composición antes mencionado y empleando estas máquinas como bloques de construcción, hemos construido una máquina compuesta que encuentra la segunda ocurrencia del símbolo distinto de espacio en blanco que se halla a la derecha de la posición inicial de la cabeza.

Es conveniente evitar los detalles interiores de los bloques de construcción a partir de los cuales construimos máquinas de Turing más complejas, como sucede en el caso de grandes sistemas de software. Si sabemos cuál es la tarea que realiza cada una de las máquinas más pequeñas, entonces nuestra preocupación será el adecuado enlace de estas máquinas y no cómo cada una lleva a cabo su tarea específica. Por esto, evitaremos el uso de diagramas de transiciones detallados como los que se muestran en la figura 3.3, y en vez de esto utilizaremos diagramas compuestos. En estos diagramas cada uno de los bloques de construcción se representa como un nodo, con flechas entre los nodos para indicar las transiciones entre bloques. Estas flechas se etiquetan de acuerdo con el valor que debe aparecer en la celda actual para que se recorra la flecha. Es decir, si una flecha del nodo A al nodo B tiene una etiqueta x , entonces la ejecución se transferirá a la máquina B si A llega a su antiguo estado de parada con una x en la celda actual. Como hicimos con los diagramas de transiciones, debe indicarse con un apuntador el nodo del diagrama compuesto donde debe comenzar la ejecución. Por ejemplo, la máquina compuesta presentada en la figura 3.3 podría resumirse con el diagrama compuesto



donde el nodo A representa la máquina que mueve su cabeza una celda a la derecha, B la máquina que busca una x y C la máquina que busca una y .



Encuentra la segunda ocurrencia del símbolo distinto de espacio en blanco que está a la derecha de la posición inicial de la cabeza

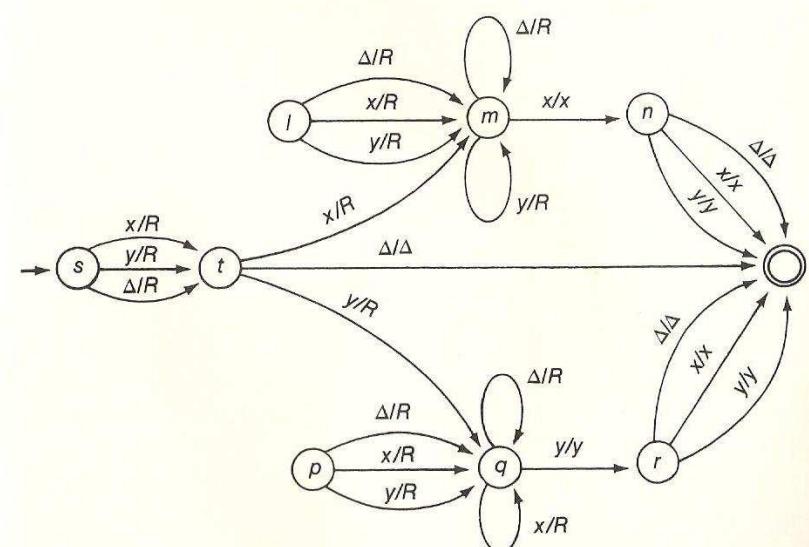


Figura 3.3 Construcción de una máquina de Turing compuesta a partir de máquinas más pequeñas

Por cuestiones de conveniencia, existen varias abreviaturas en cuanto a la notación que se emplea habitualmente al dibujar diagramas compuestos para máquinas de Turing. Uno es reemplazar varias flechas que tienen la misma fuente y el mismo destino por una sola flecha rotulada con una lista de símbolos, o quizás con $\neg x$ (léase "no x "), si hay que recorrer la flecha para todos los símbolos actuales distintos de x . Otra abreviatura es utilizar una flecha sin etiqueta para representar una transición que debe recorrerse sin importar el valor de la celda actual. Incluso esta situación se llega a abreviar más, eliminando la flecha y colocando los nodos uno al lado del otro. Por ejemplo, la secuencia $\rightarrow A \rightarrow B \rightarrow C$ podría simplificarse a $\rightarrow ABC$.

Como ejemplos, la figura 3.4a muestra un diagrama compuesto de una máquina construida a partir de M_1 y M_2 . La máquina compuesta simula las acciones de M_1 hasta el punto donde normalmente se detendría y luego simula las acciones de M_2 . La figura 3.4b representa una máquina que simula M_1 y luego hace una transferencia a M_2 sólo si la celda actual contiene una x ; de lo contrario, se detiene al concluir las acciones de M_1 . Por último, la figura 3.4c representa la máquina compuesta que comienza con la simulación de M_1 y luego simula M_2 o M_3 dependiendo de si el símbolo actual es x o no.

Otra abreviatura que utilizaremos consiste en aplicar la notación

$$\xrightarrow{x, y, z} \left\{ \omega \right\}$$

para indicar que cuando el símbolo actual es x , y o z , la máquina deberá proseguir en esta dirección, donde ω representa el símbolo que en realidad está presente. Esta notación evita saturar el diagrama con rutinas similares, aunque separadas, para cada uno de los símbolos x , y y z . En vez de esto, podemos presentar una sola rutina genérica que trata con el símbolo ω , de manera parecida a como un subprograma presenta una rutina genérica en términos de parámetros formales. Así, si la máquina

$$\rightarrow M_1 \xrightarrow{x, y} \left\{ \omega \right\} \xrightarrow{\omega} M_2 \xrightarrow{\omega}$$

fuerá a llegar al antiguo estado de parada de M_1 con el símbolo actual x o y , continuaría con la ejecución de M_2 ; esto llevaría al antiguo estado de parada de M_2 con el mismo símbolo actual que tenía al entrar a M_2 , y luego la ejecución regresaría a M_1 .

Bloques de construcción básicos

Una vez establecido este sistema de notación, consideremos ahora las máquinas de Turing elementales que emplearemos como bloques de construcción en análisis posteriores. Obtendremos una pista de cuáles deberán ser estos

bloques de construcción si recordamos que las únicas actividades disponibles para una máquina de Turing son mover la cabeza una celda a la derecha, moverla una celda a la izquierda y escribir un símbolo en la celda actual. Por consiguiente, si construimos máquinas individuales que realizan estas sencillas tareas, entonces cualquier otra máquina de Turing debe ser una composición de estos bloques.

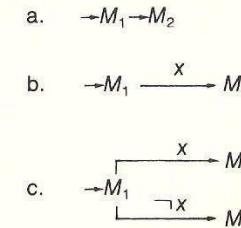


Figura 3.4 Ejemplos de máquinas de Turing compuestas

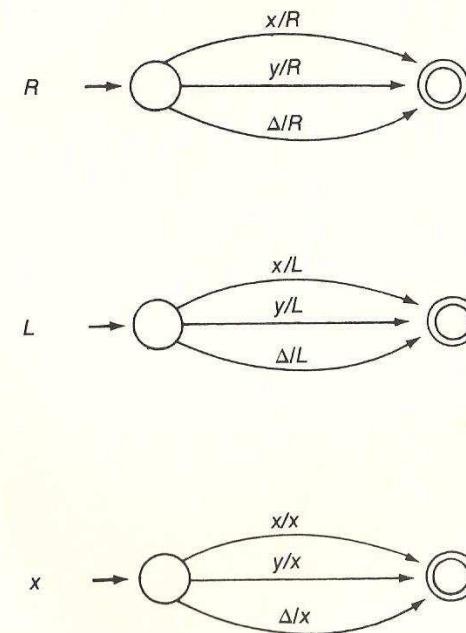


Figura 3.5 Máquinas R , L y x

La figura 3.5 muestra los diagramas de transiciones de las máquinas de Turing que efectúan cada una de estas actividades rudimentarias, suponiendo que los símbolos de la cinta son x , y y Δ (las máquinas para otros

símbolos no son más que simples generalizaciones de estas máquinas). Representamos con R la máquina que mueve su cabeza una celda a la derecha, con L la que la mueve una celda a la izquierda y con x la máquina que escribe un símbolo en la celda actual. De esta manera, una máquina de Turing que se mueve una celda a la derecha, escribe el símbolo y , se mueve una celda hacia la izquierda y se detiene, podría representarse con el diagrama compuesto

$$\rightarrow R \rightarrow y \rightarrow L$$

o, en forma condensada, como

$$\rightarrow RyL$$

Nuestro siguiente paso es establecer un repertorio de máquinas ligeramente más complejas. Un grupo de estas máquinas, presentado en la figura 3.6, lleva a cabo búsquedas sencillas. De manera más precisa, para cualquier símbolo x la máquina representada por R_x recorre la cinta a la derecha de su posición inicial en busca de una celda que contenga el símbolo x . Si encuentra ese símbolo, la máquina se detiene y esa celda será la actual; de no ser así, la máquina continuará su búsqueda eternamente.

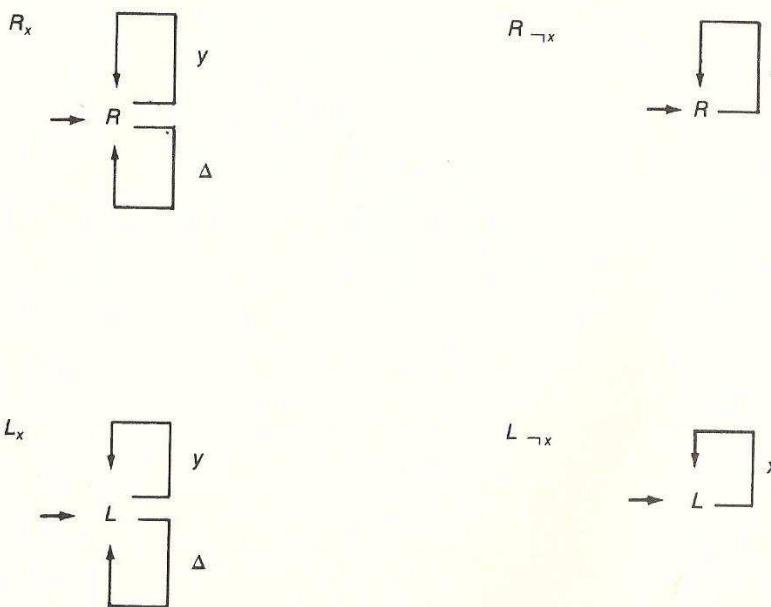


Figura 3.6 Máquinas R_x , R_{-x} , L_x y L_{-x}

De modo similar, la máquina R_{-x} buscará a la derecha de la posición inicial cualquier símbolo que no sea x . Las máquinas L_x y L_{-x} efectúan estas mismas búsquedas hacia la izquierda de la posición inicial (observe que, a diferencia de la búsqueda hacia la derecha desde la posición inicial, la búsqueda hacia la izquierda puede ocasionar una terminación anormal si llega al extremo izquierdo de la cinta sin encontrar el objetivo de la búsqueda). Las máquinas R_Δ , $R_{-\Delta}$, L_Δ y $L_{-\Delta}$ tendrán una utilidad especial para nosotros, pues pueden usarse para construir máquinas compuestas que busquen celdas en blanco o que no estén en blanco.

Otra colección de máquinas que será útil es la que lleva a cabo operaciones de desplazamiento; en la figura 3.7 se presentan dos de estas máquinas. La máquina S_R desplaza una celda hacia la derecha la cadena de símbolos que no están en blanco y que se encuentran a la izquierda de la celda actual. Así, si la cinta de S_R tuviera la configuración inicial $\Delta xyyxx\Delta\Delta\Delta\Delta\cdots$, entonces S_R se detendría con la cinta configurada como $\Delta\Delta xyyx\Delta\Delta\Delta\cdots$; o, si se aplicara a $\Delta yxy\Delta\Delta xxy\Delta\Delta\cdots$, S_R produciría $\Delta\Delta yxy\Delta xxy\Delta\Delta\cdots$. Como un caso algo especial, si la celda a la izquierda de la actual contiene un espacio en blanco, S_R únicamente borraría la celda actual. Así, S_R convertiría la configuración $xy\Delta yx\Delta\Delta\Delta\cdots$ de la cinta a $xy\Delta\Delta x\Delta\Delta\Delta\cdots$.

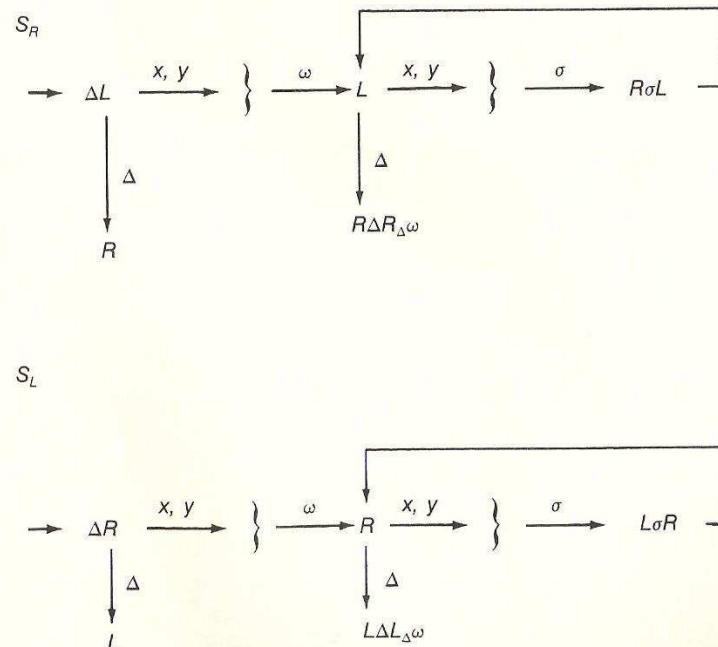


Figura 3.7 Máquinas S_R y S_L

En forma parecida, S_L realiza un desplazamiento hacia la izquierda. Si la cinta de S_L tuviera la configuración inicial $\Delta xyx\Delta\Delta\Delta\dots$, entonces S_L se detendría con su cinta configurada como $\Delta yyx\Delta\Delta\Delta\dots$; osi se aplicara a $\Delta yxy\Delta\Delta xxy\Delta\Delta\dots$, S_L produciría $\Delta yxy\Delta xxy\Delta\Delta\Delta\dots$.

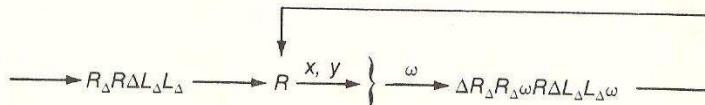


Figura 3.8 Máquina copiadora que transforma un patrón de la forma $\Delta w\Delta$
 $\Delta w\Delta w\Delta$

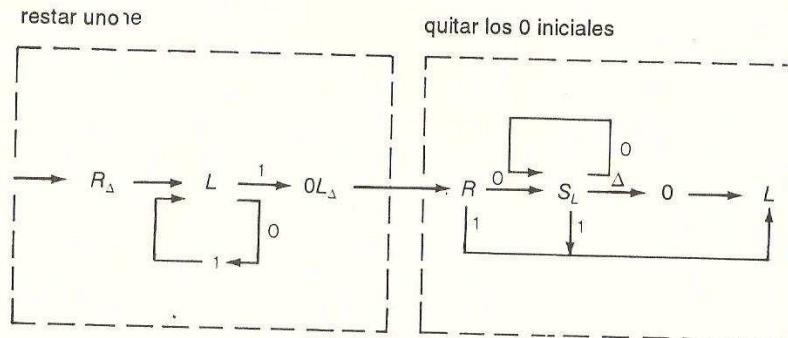


Figura 3.9 Máquina de Turing para reducir en uno una representación binaria

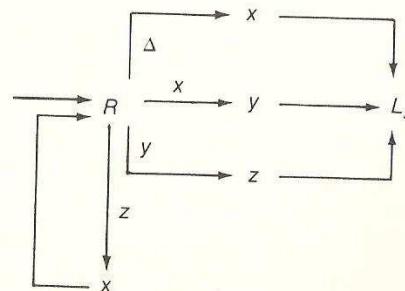


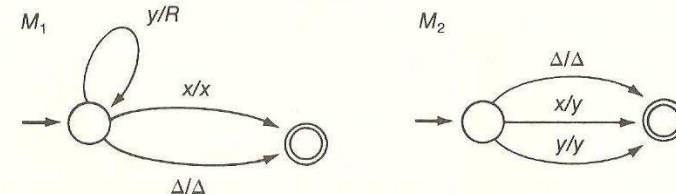
Figura 3.10 Máquina de Turing compuesta que produce la siguiente cadena en la secuencia $\lambda, x, y, z, xx, yx, zx, xy, yy, zy, xz, yz, zz, xxx, yxx, \dots$

Concluimos con varios ejemplos de cómo pueden combinarse las máquinas elementales presentadas hasta ahora para formar máquinas compuestas más complejas. La figura 3.8 define una máquina copiadora que transforma un patrón de la forma $\Delta w\Delta$ a la forma $\Delta w\Delta w\Delta$, donde w es cualquier cadena (posiblemente de longitud cero) de símbolos que no son espacios en blanco. La máquina que se presenta en la figura 3.9 supone que su entrada representa un entero positivo en notación binaria y reduce en uno el valor representado. Por último, la máquina de la figura 3.10 modifica cadenas del alfabeto $\{x, y, z\}$. Específicamente, si iniciamos la máquina con su cinta configurada como $\Delta w_1\Delta\Delta\Delta\dots$, donde w_1 es una cadena en $\{x, y, z\}^*$, entonces la máquina se detendrá con la cinta configurada como $\Delta w_2\Delta\Delta\Delta\dots$, donde w_2 es la cadena que sigue a w_1 en la secuencia $\lambda, x, y, z, xx, yx, zx, xy, yy, zy, xz, yz, zz, xxx, yxx, \dots$. Más adelante veremos la utilidad de esta máquina como bloque de construcción.

Ejercicios

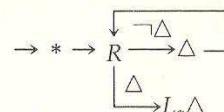
- Combine los diagramas de transiciones de las máquinas M_1 y M_2 que se muestran a continuación para formar el diagrama de transiciones de la máquina compuesta

$\rightarrow M_1 \rightarrow M_2$. ¿Qué hace la máquina compuesta?



- Utilizando los bloques de construcción presentados en esta sección, construya una máquina de Turing que reemplace la cadena de ceros y unos inmediatamente a la derecha de la cabeza por el complemento de la cadena y luego regrese la cabeza a su posición original. Suponga que el extremo derecho de la cadena está marcado con un espacio en blanco (el complemento de una cadena de ceros y unos es la cadena que se forma al sustituir los ceros originales con unos, y los unos originales, con ceros).
- ¿En qué condiciones se presentaría una terminación anormal durante la ejecución de la máquina S_R ?

4. Describa los cálculos efectuados por la siguiente máquina de Turing.



3.3 MÁQUINAS DE TURING COMO ACEPTADORES DE LENGUAJES

Hemos presentado a los autómatas de los capítulos anteriores como aceptadores de lenguajes y los hemos utilizado para evaluar cadenas y determinar su pertenencia a un lenguaje específico. Por lo tanto, es natural que estudiemos a las máquinas de Turing desde la misma perspectiva. En esta sección analizaremos los aspectos de la aceptación de cadenas realizada por máquinas de Turing; empezaremos considerando el contexto en el cual una máquina de Turing acepta una cadena de entrada.

Procedimientos de evaluación de cadenas

Para evaluar una cadena de algún alfabeto Σ con una máquina de Turing, registramos la cadena en la cinta (que de otra manera estaría en blanco) de la máquina, comenzando por la segunda celda (si fuéramos a evaluar la cadena $xxyy$, la cinta aparecería como $\Delta xxyy\Delta\Delta\Delta\cdots$). Luego colocamos la cabeza de la máquina en la celda del extremo izquierdo de la cinta y ponemos en marcha la máquina a partir de su estado inicial (véase Fig. 3.11). Decimos que la máquina acepta la cadena si, a partir de esta configuración inicial, encuentra su camino hasta su estado de parada.

Como ejemplo, en la figura 3.12 se muestra un diagrama compuesto para una máquina que acepta las cadenas que tengan precisamente la forma $x^n y^n z^n$, donde $n \in \mathbb{N}$. Esta máquina interroga la entrada reduciendo repetidamente la longitud de una cadena no vacía de la forma $x^n y^n z^n$ a través de la secuencia $x^{n-1} y^{n-1} z^n, x^{n-1} y^{n-1} z^{n-1}$ y $x^{n-1} y^{n-1} z^{n-1}$. La máquina se detiene si y sólo si este proceso de reducción produce una cadena vacía como resultado de la eliminación del mismo número de x , y y z (se emplea $\#$ como símbolo de cinta para ayudar a encontrar el extremo izquierdo de la cinta después de completar cada secuencia de reducciones).

Al igual que con los demás autómatas, la colección de cadenas aceptadas por una máquina de Turing M se llama lenguaje aceptado por la máquina y se representa con $L(M)$. Se dice que un lenguaje L es un lenguaje aceptado por una máquina de Turing si existe una máquina de Turing M tal que $L = L(M)$.

La figura 3.12 tiene una relevancia especial para nuestros fines, pues muestra que las máquinas de Turing son capaces de aceptar lenguajes que no

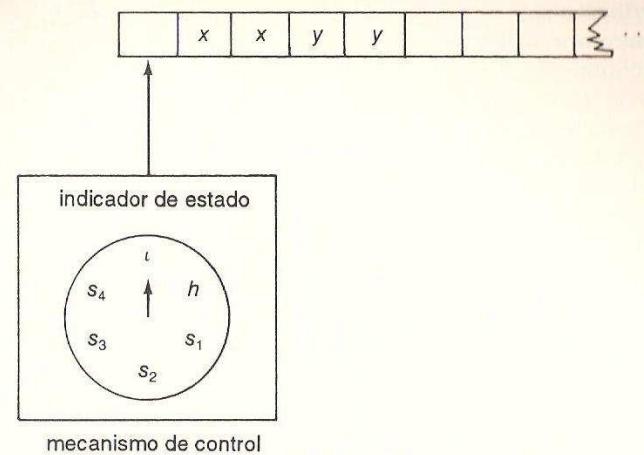


Figura 3.11 Configuración inicial de una máquina de Turing al evaluar la cadena $xxyy$.

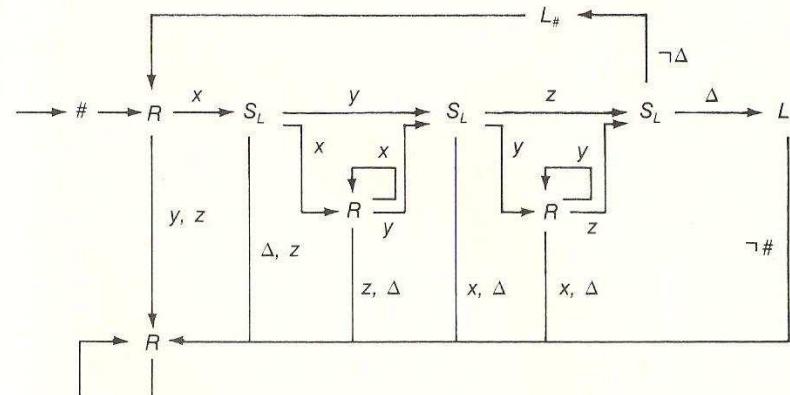


Figura 3.12 Máquina de Turing M para la cual $L(M) = \{x^n y^n z^n: n \in \mathbb{N}\}$

pueden aceptar los autómatas de pila (recuerde que $\{x^n y^n z^n: n \in \mathbb{N}\}$ no es un lenguaje independiente del contexto). De hecho, encontraremos que la clase de los lenguajes aceptados por máquinas de Turing contiene propiamente todos los lenguajes independientes del contexto, aunque requeriremos un poco de trabajo preliminar para demostrarlo.

Hemos definido la aceptación de cadenas de las máquinas de Turing de una manera que se ajusta a la de los demás autómatas. Sin embargo, en ocasiones es conveniente requerir que una máquina de Turing escriba un

mensaje de aceptación en su cinta antes de detenerse. Por ejemplo, podríamos desear que una máquina de Turing acepte una cadena si se detiene con su cinta en la configuración $\Delta Y \Delta \Delta \Delta \dots$, donde el símbolo Y se usa para representar la respuesta afirmativa (en estas circunstancias, no se considera una aceptación si la máquina se detiene con cualquier otra configuración de la cinta).

Por fortuna, este criterio de aceptación adicional no afecta la clase de lenguajes que pueden aceptar las máquinas de Turing. Dada una máquina de Turing M que acepte las cadenas con sólo detenerse, existe otra máquina de Turing M' que acepta las mismas cadenas si se detiene con la cinta configurada como $\Delta Y \Delta \Delta \Delta \dots$, y viceversa. Para justificar esta afirmación, consideraremos primero una máquina de Turing M que acepte cadenas con sólo detenerse y luego mostremos que podemos modificarla para que acepte las mismas cadenas si se detiene con una configuración de cinta $\Delta Y \Delta \Delta \Delta \dots$.

Básicamente, todo lo que tenemos que hacer es modificar M para que lleve el control de la porción de la cinta que altera durante la ejecución. Luego, después de terminar sus cálculos normales, podrá borrar esa porción de la cinta y escribir el mensaje adecuado en la cinta en blanco antes de detenerse. Para llevar este control de la porción alterada de la cinta, utilizamos los símbolos de cinta especiales $\#$ y $*$. El símbolo $\#$ se usará para marcar el extremo izquierdo de la cinta y el símbolo $*$ para marcar el extremo derecho de la porción alterada. Así, la máquina modificada deberá comenzar cualquier cálculo con

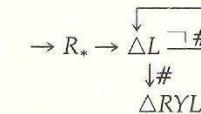
$$\rightarrow R_{\Delta} S_R R * L_{\Delta} L \# R$$

Es decir, debe desplazarse hacia el extremo derecho de la cinta de entrada y desplazar la cadena una celda a la derecha. Luego, debe marcar la primera celda a la derecha de la entrada con el símbolo $*$, regresar a la celda del extremo izquierdo de la cinta, escribir el símbolo $\#$ y ubicar su cabeza sobre el espacio en blanco en el extremo izquierdo de la cadena de entrada. En resumen, dada la configuración inicial $\Delta w \Delta \Delta \dots$, donde w es la cadena de entrada, la máquina produce la configuración $\# \Delta w * \Delta \Delta \dots$.

A partir de esta configuración, nuestra máquina modificada debe continuar con la simulación de las acciones de M . Sin embargo, al efectuar la simulación, la máquina debe estar pendiente de la ocurrencia de dos circunstancias especiales. La primera ocurre si el símbolo $\#$ se convierte en el símbolo actual. Observe que, como toda la entrada se ha desplazado una celda a la derecha, no se leerá la celda que contiene $\#$ durante la simulación, a menos que el proceso simulado sufra una terminación anormal. Entonces, si el símbolo actual fuera $\#$, nuestra máquina modificada debería mover su cabeza una celda más hacia la izquierda para que también experimentara una terminación anormal.

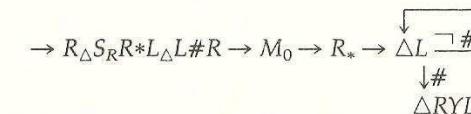
La otra circunstancia especial es la ocurrencia del símbolo $*$ como símbolo actual. Esta situación indica que el cálculo simulado ha movido su cabeza a la derecha, hacia una celda de la cinta que no se había alterado antes. En este caso, nuestra máquina modificada debe empujar la marca $*$ una celda a la derecha ejecutando $\rightarrow R^* L \Delta$ antes de continuar con la simulación.

Una vez que se han simulado los cálculos de M hasta el punto donde M ha llegado a su estado de parada, nuestra máquina deberá borrar su cinta y escribir el mensaje Y antes de detenerse. Esto se logra añadiendo el bloque



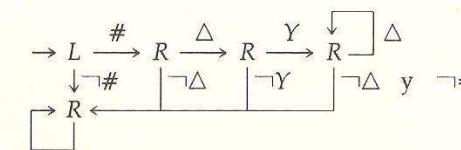
al final de nuestra máquina modificada.

En resumen, la máquina final es



donde M_0 es la máquina que simula M , excepto por las dos circunstancias especiales en los cuales surge $\#$ o $*$ como el símbolo actual.

A la inversa, suponga que comenzamos con una máquina de Turing M' que acepta cadenas deteniéndose con la cinta configurada como $\Delta Y \Delta \Delta \Delta \dots$. Entonces podríamos alterar esta máquina para obtener otra que no se detuviera en ningún otro caso. Lo único que se requiere es insistir en que la máquina revise su cinta en busca de la configuración $\Delta Y \Delta \Delta \Delta \dots$ antes de detenerse. Para esto, construimos otra máquina que marque los extremos izquierdo y derecho de la cinta en la forma antes descrita; simule las acciones de M' tomando en cuenta la ocurrencia de $\#$ o $*$ como símbolo actual; y, por último, si la simulación llega hasta el estado de parada original, confirme que la porción de la cinta limitada por las marcas $\#$ y $*$ esté configurada como $\# \Delta Y \Delta \Delta \dots \Delta \Delta *$ antes de detenerse. Este último paso podría lograrse con una rutina como



Concluimos que se pueden establecer varias maneras para que una máquina de Turing acepte sus cadenas de entrada. Podemos permitir que simplemente se detengan o insistir en que respondan con un mensaje de aceptación. Ambas estrategias tienen sus ventajas y desventajas pero, ya que cuentan con el mismo poder, tenemos la libertad de elegir el método que más convenga para la aplicación. Sin embargo, supondremos que una máquina de Turing acepta sus cadenas con sólo detenerse, a menos que se especifique lo contrario.

Máquinas de Turing de varias cintas

Ahora consideraremos las máquinas de Turing que tienen más de una cinta (este tipo de máquinas se conoce como máquinas de Turing de k cintas, donde k representa un entero positivo, en aquellos casos donde tenga relevancia el número de cintas). Cada una de estas cintas tiene un extremo izquierdo, se extiende indefinidamente hacia la derecha, y el acceso a cada una se logra a través de una cabeza separada de lectura y escritura. La transición que se ejecutará en un instante determinado dependerá de la colección de símbolos presentes para estas cabezas, junto con el estado actual de la máquina. La acción de una sola transición afecta sólo a una de las cintas de la máquina; esta acción puede ser escribir en la celda actual de esa cinta, mover la cabeza correspondiente una celda a la izquierda o moverla una celda a la derecha.

Para evaluar la aceptación de una cadena de símbolos utilizando una máquina de Turing de varias cintas, comenzamos con la máquina en su estado inicial y con la cadena de entrada registrada en la primera cinta (en el mismo formato que tendría en la cinta única de una máquina convencional), a la vez que las otras cintas están en blanco y todas las cabezas se encuentran en la celda del extremo izquierdo de sus cintas. La cadena es aceptada si, a partir de esta configuración inicial, la máquina se detiene.

Uno podría esperar que las máquinas de Turing de varias cintas tuvieran un potencial de procesamiento de lenguajes mayor que sus homólogas de una sola cinta; no obstante, el siguiente teorema muestra que esto no es cierto. Aunque en ocasiones conviene emplear una máquina con varias cintas, estas máquinas no pueden aceptar más lenguajes que las máquinas de Turing convencionales.

TEOREMA 3.1

Para cada máquina de Turing de varias cintas M , hay una máquina de Turing tradicional (de una cinta) M' tal que $L(M) = L(M')$.

DEMOSTRACIÓN

Suponga que M es una máquina de Turing de k cintas que acepta el lenguaje L . Nuestra estrategia es mostrar la posibilidad de representar el contenido de las k cintas en una sola, de modo que las acciones de M puedan ser simuladas por una máquina M' de cinta única. Visualizamos las cintas de M colocadas paralelas entre sí, con los extremos izquierdos alineados, como se presenta en la figura 3.13a, donde un apuntador bajo cada cinta indica la posición de la cabeza. Con base en esta disposición, es posible representar el contenido de las k cintas y las posiciones de sus cabezas en una tabla que contiene $2k$ filas y un número ilimitado de columnas que se extienden hacia la derecha, como se muestra en la figura 3.13b. Las filas impares de esta tabla representan las cintas; las pares se usan para indicar la posición

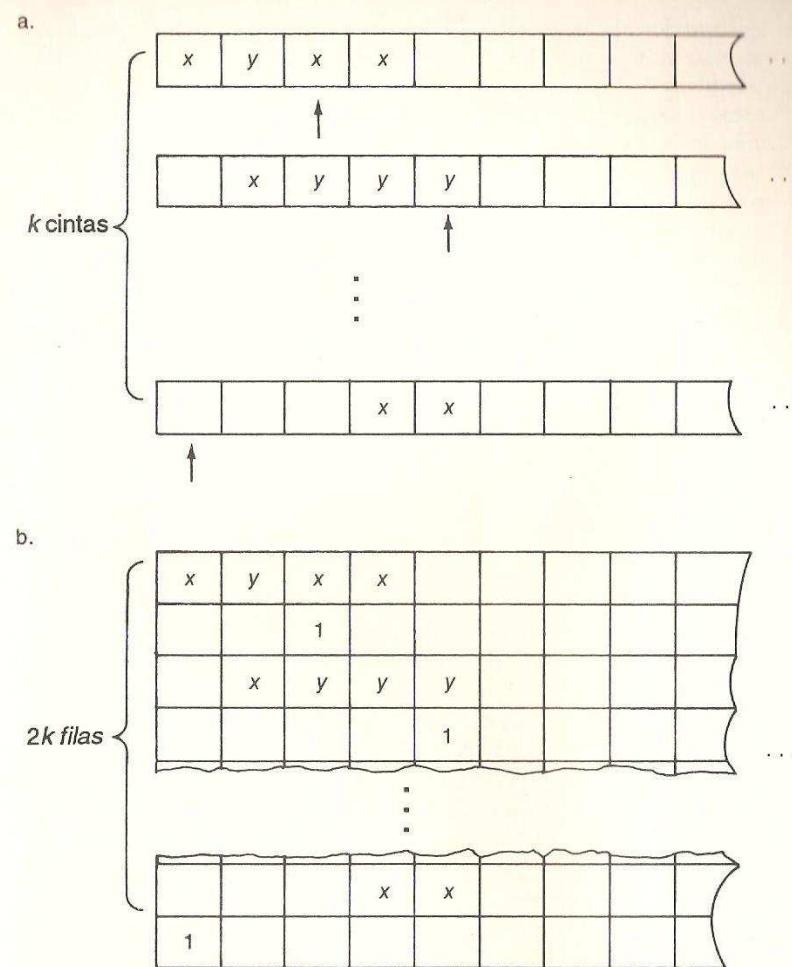


Figura 3.13 Representación de las cintas de una máquina de Turing de k cintas con un solo arreglo

de la cabeza de la cinta de la fila anterior, lo cual se logra colocando un 1 en la columna asociada a la celda actual de la fila superior y dejando en blanco todas las demás celdas.

Observe que cada una de las columnas de la tabla que acabamos de describir es en esencia una $2k$ -tupla, siendo los componentes impares símbolos de cinta de M y los pares elementos de $\{\triangle, 1\}$. Por lo tanto, existe sólo un número finito de combinaciones distintas de símbolos

que pueden aparecer en las columnas. Esto quiere decir que podemos asignar un nuevo símbolo de cinta, único, a cada una de las $2k$ -tuplas posibles y luego representar toda la tabla como una cadena infinita de estos nuevos símbolos. De esta manera podemos almacenar en una sola cinta toda la información almacenada en las k cintas de la máquina. Aunque cada una de las celdas de la cinta única contiene sólo un símbolo, éste representa una $2k$ -tupla. Por lo tanto, es conveniente expresarlo como si la cinta contuviera directamente la tupla, en vez del símbolo que la representa. Nosotros adoptaremos esto de manera informal.

Ahora estamos preparados para describir una máquina de Turing de cinta única M' que acepte el mismo lenguaje que la máquina de k cintas M . Los alfabetos de M y M' son los mismos; empero, la colección de símbolos de cinta de M' consiste en los símbolos de cinta de M , un símbolo para cada $2k$ -tupla posible y un símbolo $\#$ que se usa como marca especial. Para evaluar una cadena, la máquina M' comienza con su cinta como un duplicado exacto de la cinta 1 de la máquina M de k cintas. La primera tarea de M' es traducir el contenido de su cinta a un formato que represente las k cintas de M . Para lograrlo, M' ejecuta los pasos siguientes:

- A1. Desplaza el contenido de la cinta una celda a la derecha por medio de $\rightarrow R_{\Delta} S_R L_{\Delta}$ (la cabeza quedará sobre la segunda celda de la cinta).
 - A2. Mueve la cabeza una celda a la izquierda (es decir, hacia la celda del extremo izquierdo de la cinta), escribe la marca especial # y luego mueve la cabeza una celda a la derecha (aquí se supone que # no es un símbolo de cinta de M , sino que se emplea para marcar el final de la cinta. Si se lee durante la simulación de M , sabemos que M sobrepasó el extremo de su cinta).
 - A3. Repite los pasos siguientes hasta que el símbolo reemplazado en el paso b sea un espacio en blanco.
 - a. Mueve la cabeza de la cinta una celda a la derecha.
 - b. Reemplaza el símbolo actual, digamos x , con el símbolo de cinta que representa a la tupla $(x, \Delta, \Delta, \Delta, \dots, \Delta, \Delta)$.
 - A4. Ejecuta L_{Δ} , lo cual moverá la cabeza de regreso a la segunda celda de la cinta. Sustituye el espacio en blanco de esta celda con el símbolo de cinta que representa a la tupla $(\Delta, 1, \Delta, 1, \dots, \Delta, 1)$.

Después de ejecutar estos pasos, M' habrá traducido su cinta a un formato de cintas múltiples, marcado el extremo izquierdo de la cinta con el símbolo $\#$ y devuelto su cabeza a la tupla que representa las celdas del extremo izquierdo de las k cintas de M (véase Fig 3.14).

La tarea de M' sería ahora simular las acciones de M hasta que M se detenga o termine anormalmente. Por supuesto, para simular una

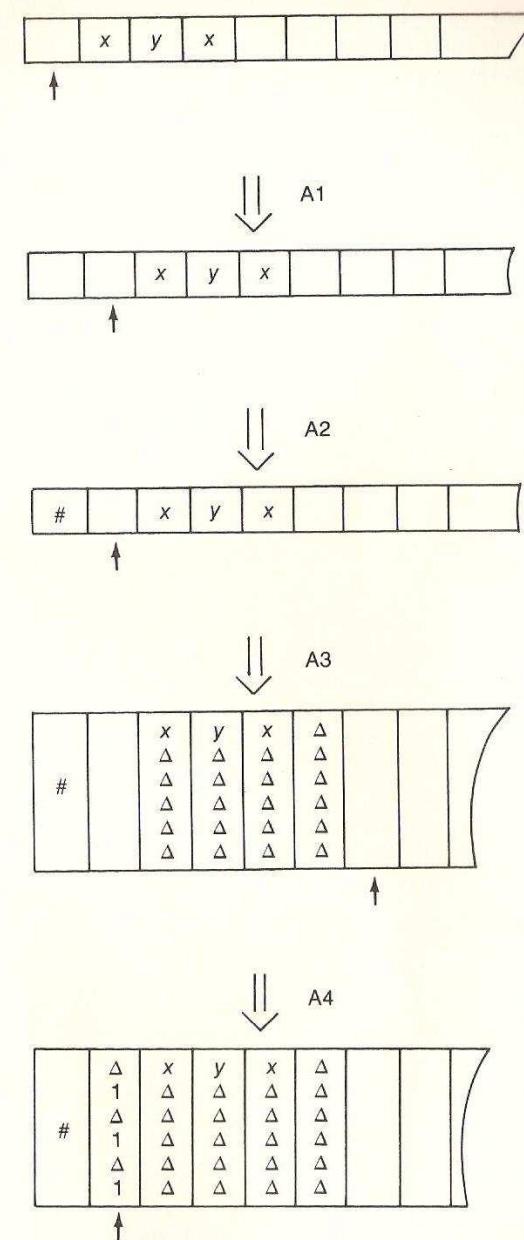


Figura 3.14 Conversión de una cinta única a un formato de tres cintas

transición de M se requiere una secuencia de pasos en M' . Diseñamos M' de manera que cada secuencia comience en un estado especial, llamado estado compuesto, que refleje el estado actual de M así como la colección de k símbolos de las celdas actuales de las cintas de M . Así, cada estado compuesto es conceptualmente una $K+1$ -tuple, donde el primer componente es un estado de M y los restantes son símbolos del alfabeto de M .

Construimos M' de forma tal que hallarse en el estado compuesto $(p, x_1, x_2, \dots, x_k)$ corresponda a que M se encuentre en el estado p con los símbolos actuales x_1, x_2, \dots, x_k . A partir de dicho estado, M' ejecutará una secuencia de pasos diseñados para simular cada transición que ejecutaría M en la situación correspondiente (observe que esta transición es determinada en forma única por el estado compuesto. De esta manera, la decisión de cuál será la siguiente transición que se ejecute no se realiza dinámicamente durante el proceso de simulación, sino que se determina antes de la ejecución, durante la construcción de la máquina). Una vez que se ha ejecutado la secuencia de simulación de la transición, M' se desplazará al estado compuesto que corresponda a la situación donde se encontraría M después de ejecutar la transición individual.

Para establecer la base de este proceso de simulación, refinamos el paso A4 anterior para que deje a M' en el estado compuesto que representa a la $k+1$ -tuple $(i, \triangle, \triangle, \dots, \triangle)$, donde i es el estado inicial de M . Así, después de terminar con el paso A4, M' se hallará en el estado compuesto asociado al estado inicial y a los símbolos actuales de M , dispuesta a iniciar el proceso de reconocimiento de la cadena.

Para simular la ejecución de la transición τ de M , M' ejecuta los pasos B1 a B3, donde empleamos la notación j_τ para representar el número de la cinta que se vería afectada por la transición τ (recuerde que una transición de la máquina de Turing de varias cintas afecta sólo a una de las cintas de la máquina).

- B1. Mueve la cabeza a la derecha hasta que el componente $2j_\tau$ de la tupla en la celda actual sea 1 (es decir, encuentra la posición de la cabeza asociada con la cinta j_τ).
- B2. a. Si la transición τ es una operación de escritura, modifica el componente $2j_\tau - 1$ según se indique.
b. Si la transición τ es un movimiento hacia la derecha, reemplaza con un espacio en blanco el 1 del componente $2j_\tau$ de la celda actual, mueve la cabeza una celda a la derecha y cambia de espacio en blanco a 1 el componente $2j_\tau$ de la tupla que allí se encuentra (si en lugar de una tupla detecta un espacio en blanco al moverse a la derecha, reemplaza éste con la $2K$ -tuple $(\triangle, \triangle, \triangle, \triangle, \dots, \triangle, \triangle)$ antes de escribir el 1).

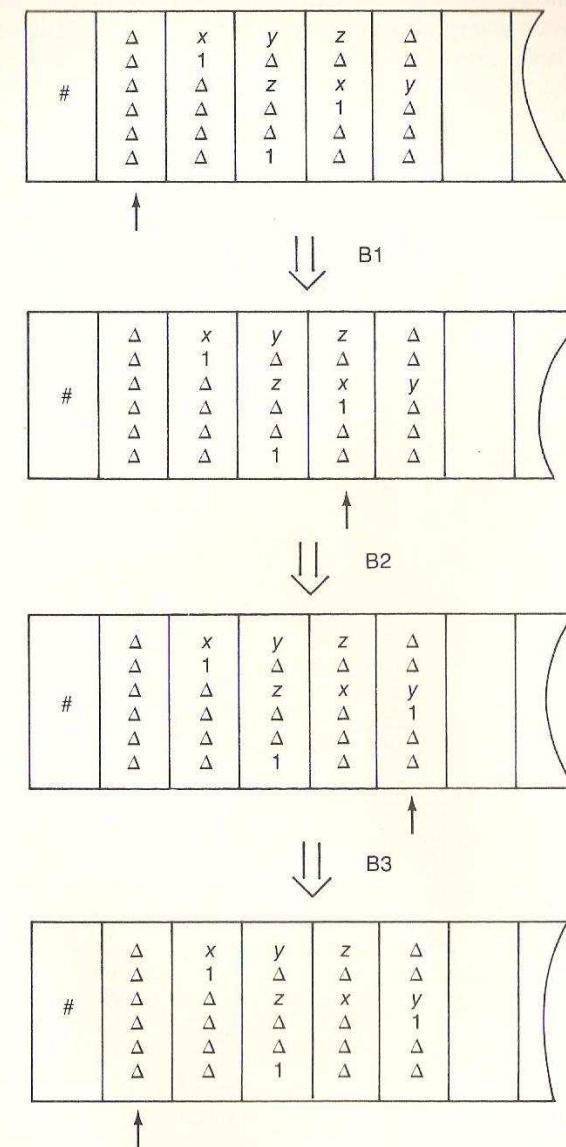


Figura 3.15 Simulación con una sola cinta de una máquina de Turing de tres cintas que ejecuta una operación de movimiento hacia la derecha en su segunda cinta

- c. Si la transición τ es un movimiento hacia la izquierda, reemplaza con un espacio en blanco el 1 del componente $2j_\tau$ de la celda actual, se mueve una celda a la izquierda y cambia de espacio en blanco a 1 el componente $2j_\tau$ de la tupla que allí se encuentra (si en lugar de una tupla detecta el símbolo # al moverse hacia la izquierda, mueve la cabeza de nuevo hacia la izquierda; esto occasionará que M' abandone los cálculos, como lo haría M).
- B3. Utilizando como guía la marca especial # del extremo izquierdo de la cinta, devuelve la cabeza a la segunda celda de la cinta y pasa al estado compuesto de M' que refleje la configuración de M después de ejecutar la transición τ .

Como ejemplo, la figura 3.15 presenta la simulación de una máquina de tres cintas que mueve la cabeza de su segunda cinta una celda hacia la derecha.

Si la ejecución de B1 a B3 conduce a un estado compuesto cuyo primer componente es el estado de parada de M , se han detenido los cálculos que se simulan. Por esto diseñamos M' de manera que también se detenga.

Al construirse de esta manera, M' únicamente simula las actividades de M y, por lo tanto, acepta una cadena si y sólo si M la hubiera aceptado. Por consiguiente, la máquina M' de cinta única acepta exactamente el mismo lenguaje que la máquina M de varias cintas.



Entre otras cosas, el teorema 3.1 refuerza nuestra confianza en la tesis de Turing. Al ampliar el potencial de los autómatas finitos con la adición de memoria para obtener la clase de los autómatas de pila, podemos vernos tentados a ampliar las máquinas de Turing de manera parecida. Sin embargo, el teorema 3.1 indica que esto no produciría un aceptador de lenguajes más poderoso. Podríamos modelar cualquier dispositivo de memoria que se quisiera agregar a una máquina de Turing mediante cintas adicionales, y por el teorema 3.1 sabemos que estas cintas adicionales no mejorarían el potencial de reconocimiento de lenguajes de la máquina. Por lo tanto, no podemos contradecir la tesis de Turing añadiendo memoria a las máquinas de Turing.

Otro fenómeno que apoya la tesis de Turing es que no se obtiene ningún poder de reconocimiento adicional si se introduce el no determinismo en el tratamiento de las máquinas de Turing. Nuestro primer paso para apoyar esta afirmación es presentar el concepto de una máquina de Turing no determinista.

Máquinas de Turing no deterministas

Una máquina de Turing no determinista es similar a una máquina de Turing tradicional; la diferencia es que quizás una máquina no determinista no se

encuentre completamente definida o, lo que es más importante, puede que ofrezca más de una transición aplicable a un par estado-símbolo. Si una máquina de Turing no determinista llegara al par estado-símbolo actual sin que existiera una transición aplicable, la máquina abandonaría los cálculos. Si una máquina de Turing no determinista llegara al par estado-símbolo actual donde puede aplicarse más de una transición, la máquina llevaría a cabo una elección no determinista y proseguiría con sus cálculos mediante la ejecución de una de las opciones aplicables.

En resumen, una máquina de Turing no determinista se puede definir como una sexteta $(S, \Sigma, \Gamma, \pi, \iota, h)$, lo mismo que una máquina de Turing determinista, excepto que el cuarto componente es un subconjunto de $((S - \{h\}) \times \Gamma) \times (S \times (\Gamma \cup \{L, R\}))$ en vez de una función de $(S - \{h\}) \times \Gamma$ a $S \times (\Gamma \cup \{L, R\})$. En este contexto, es evidente que las máquinas de Turing no deterministas forman una clase de máquinas que contiene propiamente a las máquinas de Turing tradicionales (deterministas) que hemos analizado hasta ahora.

Decimos que una máquina de Turing no determinista M acepta una cadena w si es posible que M llegue a su estado de parada después de iniciar sus cálculos con la entrada w . Decimos *posible* ya que reconocemos que en el caso de una máquina no determinista, no alcanzar el estado de parada en un intento particular podría ser el resultado de una mala decisión de la máquina, más que de una cadena de entrada impropia. Definimos como lenguaje $L(M)$ a la colección de todas las cadenas aceptadas por la máquina de Turing no determinista M . Así, una cadena w se encuentra en $L(M)$ si y sólo si M dispone de opciones que le permitan alcanzar el estado de parada al recibir la entrada w .

Puesto que la máquina de Turing no determinista es una generalización de la máquina de Turing tradicional, puede aceptar todo lenguaje que acepta una máquina tradicional. Lo que resulta más interesante es que, como lo muestra el teorema siguiente, las máquinas de Turing no deterministas son incapaces de aceptar más lenguajes que las deterministas. Por lo tanto, como sucede con los autómatas finitos, la introducción del no determinismo no aumenta el potencial de reconocimiento de lenguajes de las máquinas de Turing.

TEOREMA 3.2

Para cada máquina de Turing no determinista M , existe una máquina de Turing determinista D tal que $L(M) = L(D)$.

DEMOSTRACIÓN

Suponga que M es una máquina de Turing no determinista que acepta el lenguaje $L(M)$. Debemos mostrar la existencia de una máquina de Turing determinista que acepte el mismo lenguaje. Esto se logra de manera indirecta, mostrando que existe una máquina de Turing determinista de tres cintas M' tal que $L(M) = L(M')$ y, por el teorema

3.1. debe existir una máquina de Turing tradicional (determinista, de cinta única) que acepte $L(M)$.

Diseñamos M' para que pruebe de manera sistemática todas las opciones posibles para la máquina determinista M , con el objetivo de encontrar una combinación que lleve a la aceptación de la cadena de entrada. Se emplean las tres cintas de M' de la manera siguiente: la primera cinta contiene la cadena de entrada que se evalúa; la segunda cinta se emplea como "cinta de trabajo" en donde repetidamente M' copia la versión intacta de la cadena de entrada y luego, usando esta copia, simula una secuencia de transiciones de M ; la tercera cinta sirve para llevar el control de la secuencia de transiciones (de M) que se aplica, así como las secuencias que ya se han simulado.

El proceso de copiado de la cadena de entrada de la cinta 1 en la cinta 2 implica dos aspectos sutiles, pero importantes. En primer lugar, M' debe desplazar la cadena una celda hacia la derecha durante el proceso de copiado. Es decir, hay que colocar el contenido de la celda uno de la cinta 1 en la celda dos de la cinta 2, la celda dos de la cinta 1 va a la celda tres de la cinta 2, etcétera. Esto permite que M' coloque un símbolo especial en la celda del extremo izquierdo de su segunda cinta, lo que a su vez permite que M' detecte una terminación anormal de M sin abortar sus propios cálculos. Si la secuencia de transiciones que se simula ocasiona que M rebase el extremo de la cinta, M' detectará esta marca especial, notará que la secuencia actual no es productiva y comenzará el proceso de evaluación de otra secuencia.

El segundo aspecto que está presente en el proceso de copiado es que M' debe colocar una marca especial en el extremo derecho de la cadena de su segunda cinta. Si esta marca se detecta al simular las transiciones de M , se desplaza hacia la derecha. Así, cuando M' necesita borrar esta cinta antes de comenzar otra simulación, sólo tiene que borrar hasta esta marca especial.

Por último, debemos indicar cómo M' lleva el control de la simulación de las secuencias de transiciones de M . La idea es bastante sencilla. En primer lugar, rotule cada uno de los arcos del diagrama de transiciones de M con un símbolo único. Si existieran sólo cinco arcos en el diagrama, se podrían emplear los dígitos 1, 2, 3, 4 y 5. Luego, construya un componente en M' que genere todas las cadenas de estos símbolos en forma sistemática utilizando la cinta 3 (como modelo, véase Fig 3.10). Cada una de estas cadenas representa una secuencia de transiciones y, a final de cuentas, estará representada cada una de las transiciones posibles.

Al utilizar este generador de secuencias interno, las actividades de M' se llevan a cabo de la siguiente manera:

1. Copia la cadena de entrada de la cinta 1 a la cinta 2, en la forma antes descrita.
2. Genera la siguiente secuencia de transiciones en la cinta 3.
3. Simula esta secuencia con la cinta 2.
4. Si esta simulación conduce a un estado de parada de M , se detiene. De lo contrario, borra la cinta 2 y regresa al paso 1.

En resumen, no podemos ampliar el potencial de reconocimiento de lenguajes de las máquinas de Turing añadiendo cintas o introduciendo un comportamiento no determinista, observación que apoya la tesis de Turing. Esto nos puede llevar a la conjectura de que la clase de lenguajes aceptados por máquinas de Turing representa el final de nuestra jerarquía de lenguajes que las máquinas pueden reconocer y, por lo tanto, son de importancia para nuestro estudio. Por esto, en las secciones siguientes nos dedicaremos a aprender más acerca de esta clase de lenguajes.

Ejercicios

1. a. Diseñe una máquina de Turing que acepte el lenguaje Σ^* , donde $\Sigma = \{x, y\}$.
b. Diseñe una máquina de Turing que acepte el lenguaje \emptyset .
2. Muestre que una máquina de Turing se puede modificar para que evite una terminación anormal pero a la vez acepte las mismas cadenas que antes.
3. Muestre que si se permite en una máquina de Turing de varias cintas que las transiciones individuales afecten a más de una cinta, entonces no aumenta el potencial de la máquina. En otras palabras, muestre que cualquier transición que opera sobre más de una cinta se puede simular con una secuencia de transiciones que operan cada una en una sola cinta.
4. Muestre que cualquier cálculo de una "máquina de Turing" cuya cinta se extiende infinitamente hacia la izquierda y hacia la derecha se puede simular con una máquina de Turing de dos cintas y, por lo tanto, con una máquina de Turing tradicional.

3.4 LENGUAJES ACEPTADOS POR MÁQUINAS DE TURING

En la sección anterior analizamos los rudimentos de las máquinas de Turing como aceptadores de lenguajes, y denominamos lenguajes aceptados por máquinas de Turing a los lenguajes que estas máquinas aceptan. En esta sección describiremos con mayor detenimiento esta clase de lenguajes.

Comparación entre lenguajes aceptados por máquinas de Turing y lenguajes estructurados por frases

En el capítulo 1 presentamos las gramáticas estructuradas por frases y analizamos cómo una gramática de este tipo define un lenguaje que consiste en las cadenas de terminales generados por la gramática. Al restringir las formas de las reglas de reescritura disponibles, hemos podido identificar clases de gramáticas que generan lenguajes regulares e independientes del contexto. Ahora queremos considerar las gramáticas estructuradas por frases que no tienen restricción alguna en cuanto a sus reglas de reescritura. Así, tanto el lado izquierdo como el derecho de las reglas de reescritura pueden consistir en cualquier cadena finita de terminales y no terminales, siempre y cuando exista por lo menos un no terminal en el lado izquierdo.

Los lenguajes que generan estas gramáticas se conocen como **lenguajes estructurados por frases**. Éstos son los lenguajes que pueden definirse “gramaticalmente”, en el sentido de que sus estructuras de cadenas se pueden analizar empleando una jerarquía de estructuras de frases. Puesto que las gramáticas regulares y las independientes del contexto son casos especiales de las gramáticas sin restricciones, los lenguajes que las primeras generan están contenidos en la clase de los lenguajes estructurados por frases. Además, la figura 3.16 muestra una gramática sin restricciones que genera el lenguaje $\{x^n y^n z^n : n \in \mathbb{N}\}$, el cual, como ya sabemos, no es independiente del contexto. Por ello, los lenguajes estructurados por frases constituyen una clase de lenguajes más extensa que la de los independientes del contexto.

En esta sección, nuestro objetivo es caracterizar a los lenguajes estructurados por frases como aquellos que las máquinas de Turing pueden aceptar. Es decir, los lenguajes estructurados por frases son precisamente los lenguajes aceptados por máquinas de Turing. Esto se demostrará en dos etapas. Primero, en el teorema 3.3 mostramos que todo lenguaje aceptado por máquinas de Turing es un lenguaje estructurado por frases; luego, en el teorema 3.4 mostramos que todo lenguaje estructurado por frases es aceptado por máquinas de Turing.

La demostración que daremos para el teorema 3.3 se basa en la construcción de una gramática que genera el mismo lenguaje que el aceptado por una máquina de Turing determinada. Esta construcción requiere un sistema de notación para representar la configuración total de una máquina de Turing en cualquier etapa de sus cálculos. Al emplear este sistema, el contenido de la

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

Figura 3.16 Gramática que genera el lenguaje $\{x^n y^n z^n : n \in \mathbb{N}\}$

cinta de la máquina se representa como una cadena de símbolos de cinta encerrados entre corchetes. Primero representamos con [el extremo izquierdo de la cinta, luego presentamos la cadena de símbolos que se encuentran en la cinta, comenzando por la celda del extremo izquierdo e incluyendo por lo menos un espacio en blanco después del último símbolo distinto de un espacio, y por último cerramos la representación con]. De esta manera, una cinta que contiene $\triangle xxyx\triangle\triangle\triangle\dots$ se podría representar como $[\triangle xxyx\triangle]$ o quizás como $[\triangle xxyx\triangle\triangle\triangle\triangle\triangle]$, y a una cinta que contiene $\triangle\triangle\triangle x\triangle yx\triangle x\triangle\triangle\dots$ como $[\triangle\triangle\triangle x\triangle yx\triangle x\triangle\triangle]$.

Para completar la representación de la configuración de la máquina, insertamos en nuestra representación de la cinta el estado actual justo a la izquierda del símbolo actual. Así, si p fuera un estado de la máquina, entonces $[\triangle xpxyxx\triangle]$ representaría la configuración en la cual p es el estado actual y la configuración de la cinta es $\triangle xxyxx\triangle\triangle\triangle\dots$. De manera parecida, $[p\triangle xy\triangle]$ representaría a la máquina en el estado p con la configuración $\triangle xy\triangle\triangle\triangle\dots$ (Podemos suponer que los símbolos empleados para representar los estados de la máquina son distintos de los símbolos de cinta de la máquina.)

La importancia que para nosotros tiene esta notación es que nos ofrece un medio para expresar los cálculos de una máquina de Turing como secuencia de cadenas de símbolos, donde cada cadena representa la configuración de la máquina en un instante determinado de los cálculos. Así, si la máquina acepta cadenas deteniéndose con la cinta configurada como $\triangle Y\triangle\triangle\triangle\dots$, entonces el

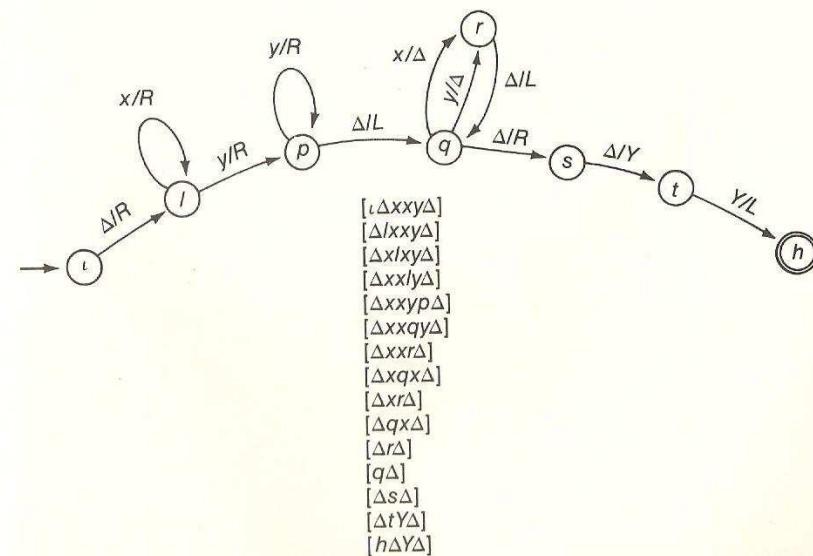


Figura 3.17 Diagrama de transiciones parcial para una máquina de Turing y la secuencia de configuraciones que produce al procesar la cadena xxy

proceso de aceptación de la cadena w se podría resumir como la secuencia de configuraciones de la máquina que comienza con $[i\Delta w\Delta]$ y termina con $[h\Delta Y\Delta]$, donde i y h representan los estados de inicio y parada de la máquina, respectivamente.

La figura 3.17 muestra una máquina de Turing que acepta el lenguaje $\{x^n y^n : m \in \mathbb{N}, n \in \mathbb{N}^+\}$ y la secuencia de configuraciones que produce al procesar la cadena xyy . Esta secuencia comienza con una configuración que contiene la cadena de entrada y termina con la configuración $[h\Delta Y\Delta]$. Así, al leerla de atrás hacia adelante obtenemos la secuencia que conduce de la configuración $[h\Delta Y\Delta]$ a una configuración que contiene la cadena de entrada; esta secuencia es bastante similar a una derivación de la cadena. De hecho, si construyéramos una gramática en la cual toda derivación tuviera que comenzar con el patrón $[h\Delta Y\Delta]$ y cuyas reglas de reescritura simularon la acción inversa de una transición de la máquina, entonces estaríamos en camino de construir una gramática que generara las cadenas aceptadas por la máquina. Éste es el enfoque que usaremos para demostrar el teorema 3.3.

TEOREMA 3.3

Todo lenguaje aceptado por máquinas de Turing es un lenguaje estructurado por frases.

DEMOSTRACIÓN

Nuestra tarea es mostrar que para cualquier lenguaje L aceptados por máquinas de Turing existe una gramática estructurada por frases que genera L . Para esto seleccionamos una máquina de Turing M que acepte cadenas sólo deteniéndose con su cinta configurada como $\underline{\underline{Y}}\Delta\Delta\Delta\dots$ y para la cual $L(M) = L$.

A partir de esta máquina definimos una gramática G de la manera siguiente: los no terminales de G se definen como S (el símbolo inicial de la gramática), $[,]$, los símbolos que representan los estados de M , y los símbolos de cinta de M , incluidos Δ y Y . Los terminales de G son los símbolos del alfabeto de M .

La única regla de reescritura que contiene el símbolo inicial es la regla

$$S \rightarrow [h\Delta Y\Delta]$$

Esta regla garantiza que cualquier derivación basada en esta gramática comenzará al final de alguna secuencia de configuraciones de la máquina. También introducimos la regla

$$\Delta] \rightarrow \Delta\Delta]$$

que permite que una derivación amplíe la cadena $[h\Delta Y\Delta]$ a cualquier longitud que se desee.

A continuación, introducimos reglas de reescritura que simulan transiciones a la inversa. Para cada transición de la forma $\delta(p, x) = (q, y)$ introducimos la regla de reescritura

$$qy \rightarrow px$$

(De esta manera, $[\Delta zqy\Delta]$ puede reescribirse como $[\Delta zpx\Delta]$, lo cual refleja que si la configuración de M fuera $[\Delta zpx\Delta]$, la aplicación de $\delta(p, x) = (q, y)$ desplazaría M hacia $[\Delta zqy\Delta]$.)

Para cada transición de la forma $\delta(p, x) = (q, R)$, introducimos la regla

$$xq \rightarrow px$$

(P. ej., $[\Delta xqyz\Delta]$ se podría reescribir como $[\Delta pxyz\Delta]$.)

Para cada transición de la forma $\delta(p, x) = (q, L)$ y cada símbolo de cinta y de M , introducimos la regla

$$qyx \rightarrow ypx$$

(Así, $[\Delta qyx\Delta\Delta]$ se podría reescribir como $[\Delta ypx\Delta\Delta]$.)

La lista de reglas de reescritura de G se completa introduciendo tres reglas que permiten que una derivación elimine los no terminales $[, i, \Delta y]$ en ciertas circunstancias. Estas reglas son

$$\begin{aligned} [i\Delta \rightarrow \lambda \\ \Delta\Delta] \rightarrow \Delta] \end{aligned}$$

y

$$\Delta] \rightarrow \lambda$$

(Por consiguiente, si una derivación produjera la configuración inicial $[i\Delta xyy\Delta\Delta\Delta]$, podría eliminar los no terminales para producir la cadena xyy .)

Por último, planteamos que $L(M) = L(G)$. Si w fuera una cadena de $L(M)$, existiría una secuencia de configuraciones de M que comenzaría con $[i\Delta w\Delta]$ y terminaría con $[h\Delta Y\Delta]$. Por lo tanto, podemos producir una derivación de la cadena w de la forma

$$S \Rightarrow [h\Delta Y\Delta] \Rightarrow \dots \Rightarrow [i\Delta w\Delta] \Rightarrow w\Delta \Rightarrow w$$

Comenzamos por aplicar la regla $S \rightarrow [h\Delta Y\Delta]$ y luego la regla $\Delta] \rightarrow \Delta\Delta]$ repetidamente hasta que la cadena $[h\Delta Y\Delta\Delta\Delta\dots\Delta]$ sea tan larga como cualquier configuración de la secuencia que represente los cálculos de M . Después aplicamos en orden inverso las reglas de reescritura correspondientes a las transiciones de la secuencia de configuraciones original. Esto producirá el patrón $[i\Delta w\Delta\Delta\Delta\dots\Delta]$, el cual podemos

reducir a w aplicando las reglas $\Delta\Delta] \rightarrow \Delta]$, $\Delta] \rightarrow \lambda$, y $[\lambda\Delta \rightarrow \lambda$. Como resultado, w estaría en $L(G)$ (como ejemplo, la figura 3.18 presenta la derivación que corresponde a la secuencia de configuraciones proporcionadas en la figura 3.17).

A la inversa, si tuviéramos una cadena en $L(G)$, su derivación daría origen a una secuencia de configuraciones que mostrarían a su vez cómo podría aceptar M la cadena. Así, cualquier cadena de $L(G)$ también se encuentra en $L(M)$.

El siguiente teorema es lo único que nos falta para justificar nuestra afirmación de que los lenguajes estructurados por frases son exactamente los lenguajes aceptados por máquinas de Turing.

TEOREMA 3.4

Todo lenguaje estructurado por frases es un lenguaje aceptado por máquinas de Turing.

DEMOSTRACIÓN

Comenzamos por observar que al aplicar la demostración del teorema 3.1 a una máquina de Turing no determinista de varias cintas se produciría una máquina no determinista de cinta única que aceptaría el mismo lenguaje que la máquina con varias cintas (es posible que la transición por simular de un estado compuesto de la forma $(p, x_1, x_2, \dots, x_k)$ ya no esté determinada en forma única, pero entonces se conocerán todas las opciones antes de la ejecución y, por tanto, podrán incorporarse al autómata permitiendo el no determinismo). Así, para cualquier máquina de Turing no determinista M de varias cintas existe una máquina de Turing M' de una sola cinta tal que $L(M) = L(M')$. Tal observación nos permite demostrar este teorema mostrando que para cada gramática G existe una máquina de Turing no determinista N de dos cintas tal que $L(G) = L(N)$. Después, N se podría simular con una máquina de Turing no determinista de cinta única (por la observación anterior), la cual podría ser simulada a su vez por una máquina de Turing convencional (según el Teorema 3.2).

A continuación observamos que una máquina de Turing puede realizar cualquier regla de reescritura de la gramática. Es decir, si una cadena de símbolos v aparece en algún lugar de la cinta de la máquina y la gramática contiene la regla $v \rightarrow w$, donde w representa una cadena (posiblemente vacía) de terminales y no terminales, entonces la máquina puede reemplazar la cadena v por la cadena w aplicando operaciones de escritura y de desplazamiento hacia la izquierda y hacia la derecha.

$$\begin{aligned} S &\Rightarrow [h\Delta Y\Delta] \\ &\Rightarrow [h\Delta Y\Delta\Delta] \\ &\Rightarrow [h\Delta Y\Delta\Delta\Delta] \\ &\Rightarrow [\Delta t\lambda\Delta\Delta\Delta] \\ &\Rightarrow [\Delta s\Delta\Delta\Delta\Delta] \\ &\Rightarrow [q\Delta\Delta\Delta\Delta\Delta] \\ &\Rightarrow [\Delta r\Delta\Delta\Delta\Delta] \\ &\Rightarrow [\Delta qx\Delta\Delta\Delta] \\ &\Rightarrow [\Delta xr\Delta\Delta\Delta] \\ &\Rightarrow [\Delta xqx\Delta\Delta] \\ &\Rightarrow [\Delta xxr\Delta\Delta] \\ &\Rightarrow [\Delta xxqy\Delta] \\ &\Rightarrow [\Delta xxyp\Delta] \\ &\Rightarrow [\Delta xxly\Delta] \\ &\Rightarrow [\Delta xlxy\Delta] \\ &\Rightarrow [\Delta lxx\Delta] \\ &\Rightarrow [\lambda xx\Delta] \\ &\Rightarrow xxy\Delta \\ &\Rightarrow xxy \end{aligned}$$

Figura 3.18 Derivación de la cadena $xx\lambda$ correspondiente a la secuencia de configuraciones de la figura 3.17

Construimos ahora una máquina no determinista de dos cintas que opera de la siguiente manera: utiliza la cinta 1 para almacenar la cadena de entrada por evaluar. Escribe el símbolo inicial de la gramática en la cinta 2. Luego aplica repetidamente y de manera no determinista las reglas de reescritura a la cadena de la cinta 2 (decimos "de manera no determinista" ya que puede existir más de una regla aplicable en un momento determinado). Si el contenido de la cinta 2 se convierte en una cadena con sólo terminales, compara esta cadena con la cadena de entrada que está almacenada en la cinta 1. Si ambas cadenas son idénticas, la máquina se detiene; de lo contrario, mueve una de las cabezas hacia la izquierda hasta que ocurra una terminación anormal.

En resumen, este proceso únicamente emplea la cinta 2 para calcular una derivación con base en las reglas de la gramática. Si la cadena de entrada se puede derivar de la gramática, entonces es posible que esta cadena sea la producida en la cinta 2. En este caso, la máquina aceptará la entrada, deteniéndose. Sin embargo, si la entrada no puede derivarse de la gramática, la cadena producida en la cinta 2 nunca podrá ser igual a la de entrada, y sería imposible que la máquina aceptara la cadena. Por consiguiente, el lenguaje aceptado por la máquina es el lenguaje generado por la gramática.

Como resumen, los lenguajes y las máquinas estudiados hasta ahora forman la jerarquía que se presenta en la figura 3.19.

Alcance de los lenguajes estructurados por frases

La equivalencia entre los lenguajes estructurados por frases y los lenguajes aceptados por máquinas de Turing significa que las máquinas de Turing se pueden utilizar para estudiar el alcance de los lenguajes estructurados por frases. Esto es la esencia del teorema siguiente.

TEOREMA 3.5

Para cada alfabeto Σ existe al menos un lenguaje sobre Σ que no es un lenguaje estructurado por frases.

DEMOSTRACIÓN

Sea L un lenguaje estructurado por frases del alfabeto Σ . Entonces, por el teorema 3.4, existe una máquina de Turing M tal que $L(M) = L$. Comenzamos nuestra demostración señalando que existe una M' cuyos símbolos de cinta se eligen del conjunto $\Sigma \cup \{\Delta\}$.

Si M es la máquina de Turing tal que $L(M) = L$ y los símbolos de cinta de M incluyen más que $\Sigma \cup \{\Delta\}$, podemos construir otra máquina de Turing M' con símbolos de cinta de $\Sigma \cup \{\Delta\}$, de modo que $L(M') = L(M) = L$. De hecho, sea x cualquier símbolo (distinto de espacio en blanco) de Σ . Entonces, disponga en una lista los símbolos distintos de espacios en blanco de M y represente cada entrada de la lista con una cadena de x de longitud igual a la posición del símbolo

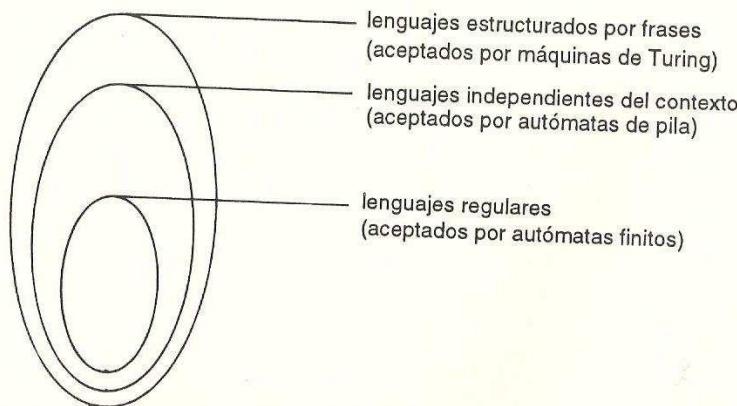


Figura 3.19 Jerarquía de máquinas y lenguajes

en esta lista (el primer símbolo se representa con x , el segundo con xx , etc.). Así, cada uno de los símbolos de la cinta de M que no sea un espacio en blanco, y por consiguiente también cada símbolo del alfabeto de M' , se representa con una cadena de x única. De esta manera, siempre es posible representar el contenido de la cinta de M por medio de cortas cadenas de x separadas por espacios en blanco (un espacio en blanco de la cinta de M se codificaría como dos espacios consecutivos, es decir, una cadena de x vacía).

Ahora construya M' para que traduzca su entrada a esta forma codificada, simule las acciones de M y se detenga únicamente si M se detiene. Así, M' aceptará exactamente las mismas cadenas que acepta M , sin emplear más que los símbolos de cinta $\Sigma \cup \{\Delta\}$.

Concluimos que dado un lenguaje estructurado por frases de un alfabeto Σ , existe una máquina de Turing con símbolos de cinta $\Sigma \cup \{\Delta\}$ tal que $L(M) = L$.

Lo que queda de nuestra demostración es, en esencia, lo mismo que la demostración del teorema 1.1. Podemos generar de manera sistemática una lista de todas las máquinas de Turing con símbolos de cinta $\Sigma \cup \{\Delta\}$, generando primero aquellas máquinas con sólo dos estados (recuerde que una máquina de Turing tiene cuando menos dos estados), seguido por las que tienen tres estados, etcétera. Por lo tanto, existe una cantidad contable de tales máquinas de Turing. Por otra parte, existe una cantidad infinita de cadenas en Σ^* y, debido a ello, es incontable el número de lenguajes que se pueden formar a partir de Σ . Por consiguiente, existen más lenguajes basados en Σ que máquinas de Turing con símbolos de cinta en $\Sigma \cup \{\Delta\}$. Así mismo, deben existir lenguajes basados en Σ que no son lenguajes estructurados por frases.

Aquí es donde comienza a complicarse la trama. Los teoremas 3.3, 3.4 y 3.5 nos dicen que existen lenguajes que no tienen bases gramaticales y además que las máquinas de Turing sólo pueden reconocer aquellos lenguajes que sí tienen bases gramaticales. Si aceptamos la tesis de Turing de que las máquinas de Turing engloban la esencia de cualquier proceso computacional, debemos concluir que ningún proceso algorítmico puede reconocer los lenguajes que no tienen bases gramaticales.

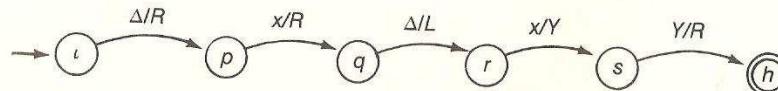
Esta conclusión no sólo implica que existen lenguajes que no podemos analizar sintácticamente con un computador, sino además tiene ramificaciones relacionadas con la búsqueda de sistemas que comprendan los lenguajes naturales, un área de gran actividad en las investigaciones recientes. De hecho, nos indica que en el desarrollo de un sistema de procesamiento de lenguajes naturales existe el requisito de que el lenguaje que se procesa tenga una estructura gramatical bien definida; si el lenguaje natural no cuenta con esta estructura, no seremos capaces de procesarlo algorítmicamente.

Este requisito tiene además un estrecho vínculo con el tema de la inteligencia natural comparada con la inteligencia artificial. Si, como muchos creen, la mente humana opera ejecutando algoritmos, entonces la tesis de Turing indica que existen lenguajes cuya sintaxis no puede analizar la mente humana. Por otra parte, si la inteligencia natural se basa en un nivel más poderoso que la ejecución de algoritmos, entonces la tesis de Turing sugiere que el desarrollo de máquinas verdaderamente inteligentes seguirá por siempre siendo un sueño.

Vemos entonces que la teoría presentada hasta ahora tiene consecuencias significativas. El único punto débil en la cadena es la tesis de Turing, la *conjetura* de que el concepto de computación con una máquina de Turing es tan poderoso (aunque quizás no tan conveniente) como cualquier otro modelo computacional. Como mencionamos antes, la mayoría de los científicos de la computación acepta actualmente esta tesis. Gran parte de esta aceptación proviene de hecho de que la tesis de Turing es congruente con otros resultados y conjeturas que han surgido al estudiar la computación desde otras perspectivas; en el capítulo siguiente consideraremos algunas de estas estrategias.

Ejercicios

- Dibuje un diagrama de transiciones para todas las máquinas de Turing con símbolos de cinta x y Δ que sólo tengan dos estados (uno inicial y uno de parada). ¿Cómo se relacionan su capacidad para llevar a cabo esta tarea y el hecho de que sea contable la colección de todas las máquinas de Turing cuyo conjunto de símbolos de cinta es $\{x, \Delta\}$?
- Muestre que al agregar una segunda pila a un autómata de pila obtenemos una clase de máquinas con el mismo poder de aceptación de lenguajes que la clase de máquinas de Turing.
- Utilice el proceso de construcción descrito en la demostración del teorema 3.3, para desarrollar una gramática estructurada por frases que acepte el mismo lenguaje aceptado por la máquina de Turing cuyo diagrama de transiciones aparece a continuación (la máquina acepta cadenas deteniéndose con la cinta configurada como $\underline{\Delta}Y\Delta\Delta\Delta\dots$). Con la gramática que obtenga, muestre una derivación de la cadena x .



- Muestre que el lenguaje $\{x^n y^{2n} z^{4n} : n \in \mathbb{N}\}$ es aceptado por máquinas de Turing.

3.5 MÁS ALLÁ DE LOS LENGUAJES ESTRUCTURADOS POR FRASES

Hemos demostrado la existencia de lenguajes que no están estructurados por frases (o, lo que es lo mismo, no son aceptados por máquinas de Turing), pero aún no identificamos explícitamente un lenguaje de éstos. Un objetivo fundamental de esta sección es llenar este hueco; la presentación de las máquinas de Turing universales y la distinción entre lenguajes aceptables y decidibles representan importantes extensiones de nuestro análisis.

Sistema de codificación de máquinas de Turing

Para identificar un lenguaje que no es aceptado por máquinas de Turing (y por lo tanto no es estructurado por frases) necesitaremos un sistema de codificación con el cual podamos representar las máquinas de Turing con alfabeto Σ y símbolos de cinta $\Sigma \cup \{\Delta\}$ como cadenas que únicamente contienen ceros y unos. Por ello, hacemos una pausa para presentar dicho sistema antes de proseguir con el tema principal de la sección.

Dada una máquina M por representar, nuestro sistema de codificación necesita que acomodemos los estados de M en una lista cuyo primer elemento corresponda al estado inicial y el segundo al estado de parada. Con base en el orden de esta lista, podemos hablar del primer estado de M , del segundo estado de M y, en general, del estado j de M . Establecemos que el estado j de M se representará con una cadena de ceros de longitud j . Así, el estado inicial estará representado por 0, el estado de parada por 00 y el siguiente estado (si existe) por 000.

Luego, representamos los símbolos L y R , así como los símbolos en Σ (los símbolos de cinta de M distintos de espacio en blanco) como cadenas de ceros. Esto se hace acomodando en una lista los símbolos de Σ y representando L con 0, R con 00, el primer símbolo de la lista con 000, el segundo símbolo con 0000 y, en general, el símbolo j con una cadena de ceros de longitud $j + 2$.

Si ahora establecemos que el símbolo del espacio en blanco se representará con la cadena vacía, obtenemos un sistema en el cual podemos representar los símbolos L y R , los estados de M y los símbolos de la cinta de M por medio de cadenas de ceros. A su vez, esto nos permite representar cualquier transición de M como una cadena de ceros y unos. A fin de cuentas, se puede identificar cualquier transición (que debe tener la forma $\delta(p, x) = (q, y)$ con una cuádrupla (p, x, q, y) donde p es el estado actual, x es el símbolo actual, q es el nuevo estado y y es o bien un símbolo de cinta (si la transición es una operación de escritura) o bien L o R (si la transición es una operación de movimiento de la cabeza)). De esta manera, es posible representar toda la transición como cuatro cadenas de ceros separados por unos. Por ejemplo, la cadena 01000100100 representaría la transición $\delta(i, x) = (h, R)$, donde x es el símbolo representado por 000 y h es el estado de parada de la máquina. Puesto que un espacio en blanco se representa

con la ausencia de ceros, la cadena 011001 representa la transición $\delta(t, \Delta) = (h, \Delta)$, donde h es una vez más el estado de parada de la máquina.

Observe que una lista de todas las transiciones disponibles para una máquina de Turing con símbolos de cinta $\Sigma \cup \{\Delta\}$ constituye una descripción completa de la máquina. Por lo tanto, cualquier máquina de este tipo se puede representar como una lista de transiciones codificada. Adoptaremos la regla convencional de añadir un 1 al inicio y al final de la lista, y un solo 1 para separar las transiciones de la lista. Así, la cadena 10110010001010001001001 (un 1 introductorio seguido por el código de transición 011001000, un 1 separador, el código 01000100100 y un 1 final) representa la máquina con dos transiciones, $\delta(t, \Delta) = (h, x)$ y $\delta(t, x) = (h, R)$, que se muestra en la figura 3.20.

Al representar de esta forma una máquina de Turing, establecemos que las transiciones por incluir en la lista tendrán el orden siguiente: primero presentamos todas las transiciones que surgen del estado 0 (el inicial), luego las que se originan del estado 000, seguidas por las que tienen su origen en el estado 0000, etc., hasta incluir todos los estados donde pueden originarse transiciones. Las transiciones correspondientes a un estado se acomodan de acuerdo con el símbolo que se requiere en la celda actual de la cinta, comenzando por la transición que requiere un espacio en blanco como símbolo actual, luego la transición que requiere el símbolo con código 000, después la transición cuyo símbolo actual es 0000, etcétera. Esta uniformidad hace más fácil evaluar una cadena de ceros y unos para ver si constituye una representación válida de alguna máquina de Turing (determinista y, por lo tanto, completamente definida).

Un lenguaje no estructurado por frases

Al emplear esta forma de codificación, encontramos que cada máquina de Turing con alfabeto Σ y símbolos de cinta $\Sigma \cup \{\Delta\}$ se puede representar como una cadena de ceros y unos, la cual a su vez puede interpretarse como un entero

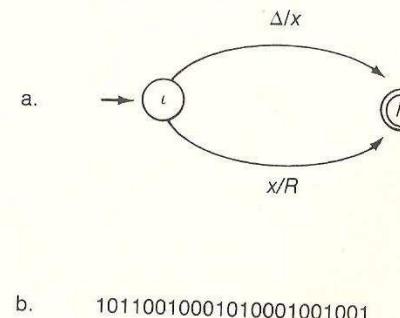


Figura 3.20 a. Máquina de Turing sencilla
b. La misma máquina en forma codificada

no negativo escrito en binario. Es más, si contáramos en binario llegaríamos en algún momento al patrón que representa una máquina de Turing específica cualquiera con símbolos de cinta $\Sigma \cup \{\Delta\}$. Por supuesto, las representaciones binarias de muchos enteros no negativos no corresponden a representaciones de máquinas válidas. Establezcamos que cada uno de estos enteros se asociará con la sencilla máquina que se muestra en la figura 3.21. Así, tenemos entonces una función de \mathbb{N} sobre las máquinas de Turing con alfabeto Σ y símbolos de cinta $\Sigma \cup \{\Delta\}$. Empleamos la notación M_i para representar la máquina que esta función relaciona con el entero i .

Con base en esta función podemos construir otra, esta vez de Σ^* sobre la colección de máquinas de Turing con alfabeto Σ y símbolos de cinta $\Sigma \cup \{\Delta\}$. Basta con asociar una cadena w de Σ^* con la máquina $M_{|w|}$, donde $|w|$ representa la longitud de w . A fin de hacer más sencilla esta notación, utilizamos M_w para representar la máquina asociada a w por medio de esta función.

Observe que los símbolos de w también se encuentran en el alfabeto de M_w , por lo que tiene sentido aplicar M_w a la cadena de entrada w . Definimos que el lenguaje L_0 es el subconjunto $\{w: M_w \text{ no acepta } w\}$ de Σ^* ; una cadena w de Σ^* se halla en L_0 si y sólo si ésta no es aceptada por su máquina M_w correspondiente.

Ahora nuestra tarea es mostrar que L_0 no es aceptado por máquinas de Turing. Para esto, mostramos que se llega a una contradicción si se supone lo contrario. Si L_0 fuera aceptado por alguna máquina de Turing, utilizando un argumento similar al de la demostración del teorema 3.5 podemos suponer que el alfabeto de la máquina es Σ y su conjunto de símbolos de cinta es $\Sigma \cup \{\Delta\}$. Así mismo, L_0 debe ser aceptado por M_{w_0} para alguna cadena w_0 en Σ^* (toda máquina de Turing con alfabeto Σ y símbolos de cinta $\Sigma \cup \{\Delta\}$ es M_w para una w de Σ^*). Por lo tanto, $L_0 = L(M_{w_0})$.

Ahora nos preguntamos si la cadena w_0 existe o no en $L(M_{w_0})$ (tiene que estar o no estar), pero, como veremos en un momento, ambas opciones nos llevan a contradicciones. Con base en la definición de L_0 , sabemos que $w_0 \in L(M_{w_0})$ implica que $w_0 \notin L_0$, y que $w_0 \notin L(M_{w_0})$ implica que $w_0 \in L_0$. Sin embargo, como $L_0 = L(M_{w_0})$, ambos enunciados son contradictorios. En vista de esta paradoja, debemos concluir que nuestra conjectura con respecto a la aceptación de L_0 es falsa; en otras palabras, el lenguaje L_0 no es aceptado por su máquina de Turing.

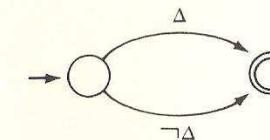


Figura 3.21 Máquina de Turing sencilla

Máquinas de Turing universales

Nuestro siguiente objetivo es mostrar que el complemento de L_0 , el conjunto $\{w : M_w \text{ acepta } w\}$, es aceptado por máquinas de Turing. Esto demostrará que la capacidad de una máquina de Turing para aceptar un lenguaje no es simétrica con la capacidad para rechazar el complemento del lenguaje. En otras palabras, existen casos donde se puede construir una máquina de Turing que identifique las cadenas de un lenguaje, pero no se puede construir una máquina de Turing que identifique las cadenas que no están en el lenguaje.

Sin embargo, para mostrar que el complemento de L_0 es aceptado por máquinas de Turing, necesitamos presentar el concepto de **máquina de Turing universal**. Ésta no es más que una máquina de Turing "programable" que, dependiendo de su programa, puede simular cualquier otra máquina de Turing. De esta forma, las máquinas de Turing universales son los antepasados abstractos de nuestros modernos computadores programables que leen y ejecutan los programas almacenados en sus memorias. Es más, las máquinas de Turing universales están diseñadas para ejecutar programas que se almacenan en sus cintas.

Un programa para una máquina de Turing universal no es más que una versión codificada de una máquina de Turing que lleva a cabo la tarea que se desea ejecutar la máquina universal. Suponga que queremos programar una máquina de Turing universal para que desempeñe una actividad específica. Primero diseñaríamos una máquina de Turing tradicional que realizará la tarea; luego codificaríamos esta máquina como una cadena de ceros y unos, como ya lo hemos hecho. La cadena de código sería el programa de la máquina universal.

El siguiente paso sería codificar los datos que se usarían como entrada para los cálculos deseados. Para esto recordamos que a cada símbolo de la cinta distinto de espacio en blanco, correspondiente a la máquina que acabamos de codificar, se le asigna un código de tres o más ceros. Por consiguiente, cualquier cadena de estos símbolos se puede representar con la secuencia de códigos adecuada. En esta secuencia separamos con un 1 los códigos adyacentes y encerramos entre corchetes toda la secuencia, con un 1 en cada extremo (Fig. 3.22). Observe que la cadena vacía estaría representada por 11, una secuencia vacía de códigos.

Después de codificar la máquina que se simulará y la cadena que servirá como entrada, colocamos estos códigos en la cinta de entrada de la máquina de Turing universal, como se describe a continuación. La celda del extremo izquierdo permanece en blanco, luego se encuentra la versión codificada de la máquina que se simulará, seguida por la cadena de entrada codificada (la máquina universal puede detectar la separación entre la codificación de la máquina y los datos, ya que el código de la máquina termina con un 1 y el código de los datos comienza con 1. Para ser más precisos, toda transición codificada debe comenzar con el código de un estado, lo que requiere por lo menos un 0. Así, es posible detectar el fin de la lista de transiciones con la presencia de un código de estado incorrecto. Véase Fig. 3.23).

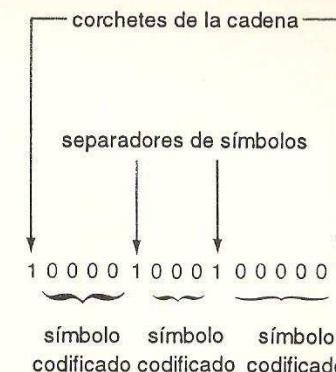


Figura 3.22 Cadena de tres símbolos en forma codificada

Una vez que esta información se ha colocado en la cinta de la máquina universal, ubicamos la cabeza de la máquina sobre la celda del extremo izquierdo de la cinta y ponemos en marcha la máquina a partir de su estado inicial. Partiendo de esta configuración, la máquina de Turing universal extrae y simula las transiciones que encuentra en la primera porción de la cinta conforme manipula la cadena existente en la segunda porción de la cinta.

Para ser más precisos, podríamos visualizar una máquina de Turing universal como una máquina de tres cintas. La primera cinta se usa para almacenar el programa de entrada y los datos, así como para contener cualquier salida; la segunda cinta sirve como área de trabajo, en la cual se manipulan los datos; y la tercera cinta se emplea para contener una representación del estado actual de la máquina simulada.

Esto no puede ser el inicio de otra transición, ya que esa interpretación produciría un código de estado inválido.

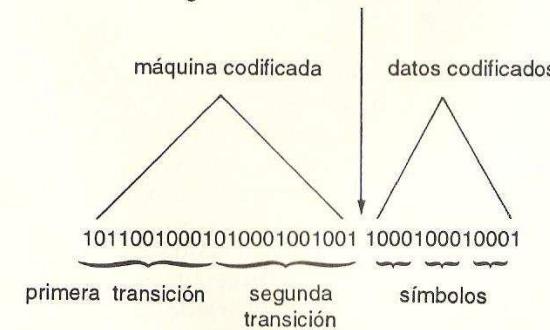


Figura 3.23 Máquina de Turing y sus datos en forma codificada

La figura 3.24 describe tal máquina como una composición de los bloques de construcción de máquinas, descritos antes. Los exponentes se emplean para indicar la cinta destino (por ejemplo, R^1 mueve una celda hacia la derecha la cabeza de la cinta 1, mientras que R^2 mueve la cabeza de la cinta 2). Las principales funciones de la máquina compuesta son: primero encuentra el inicio de la cadena de entrada codificada, copia esta cadena a la cinta 2 y coloca en la cinta 3 el código del estado inicial. Luego, prosigue con la búsqueda de las transiciones codificadas en la cinta hasta encontrar una que sea aplicable. Una vez que encuentra esta transición, la simula en la cinta 2 y actualiza el código de estado de la cinta 3 para reflejar el nuevo estado. Si el estado simulado se convierte en el estado de parada, borra la cinta 1, copia el contenido de la cinta 2 en la cinta 1, coloca la cabeza de la cinta 1 en el lugar donde se encontraba la cabeza de la cinta 2 al llegar al estado de parada, y se detiene.

Esta presentación de una máquina de Turing universal se ha hecho en el contexto de una máquina de tres cintas, pero sabemos que es posible simular cualquier máquina de Turing de tres cintas con una máquina de Turing que sólo tenga una. Consideramos entonces que una máquina de Turing universal es una máquina de una cinta denotada por T_u .

Comparación entre lenguajes aceptables y decidibles

Con el apoyo de una máquina de Turing universal, ahora podemos construir una máquina de Turing que acepte el complemento del lenguaje L_0 . Comenzamos por construir una máquina de Turing M_{pre} que procese una cadena de entrada w de Σ^* de la siguiente manera:

1. Genera una cadena de ceros y unos que represente a la máquina M_w (se trata de un proceso bastante directo, ya que la representación deseada para M_w es el código de la máquina presentada en la figura 3.21 o simplemente la representación binaria de la longitud de w).
2. Coloca la cadena obtenida en la cinta de la máquina, seguida por la versión codificada de w .

Por lo tanto, se puede construir una máquina que acepta el complemento de L_0 formando la máquina compuesta $\rightarrow M_{pre} \rightarrow T_u$. De hecho, dada una cadena de entrada w , esta máquina de Turing aplicaría M_w a w y se detendría si y sólo si se encontrara que M_w acepta w (véase Fig. 3.25). Así, acepta precisamente el lenguaje $L_1 = \{w : M_w \text{ acepta } w\}$, el cual es el complemento de L_0 .

Hemos confirmado entonces que no es obligatorio que el complemento de un lenguaje aceptados por máquinas de Turing sea también aceptados por máquinas de Turing; L_1 lo es, no así su complemento L_0 . Esta falta de simetría no es un simple hecho curioso, sino que tiene repercusiones considerables. En nuestro caso, quiere decir que la capacidad para aceptar un lenguaje no es

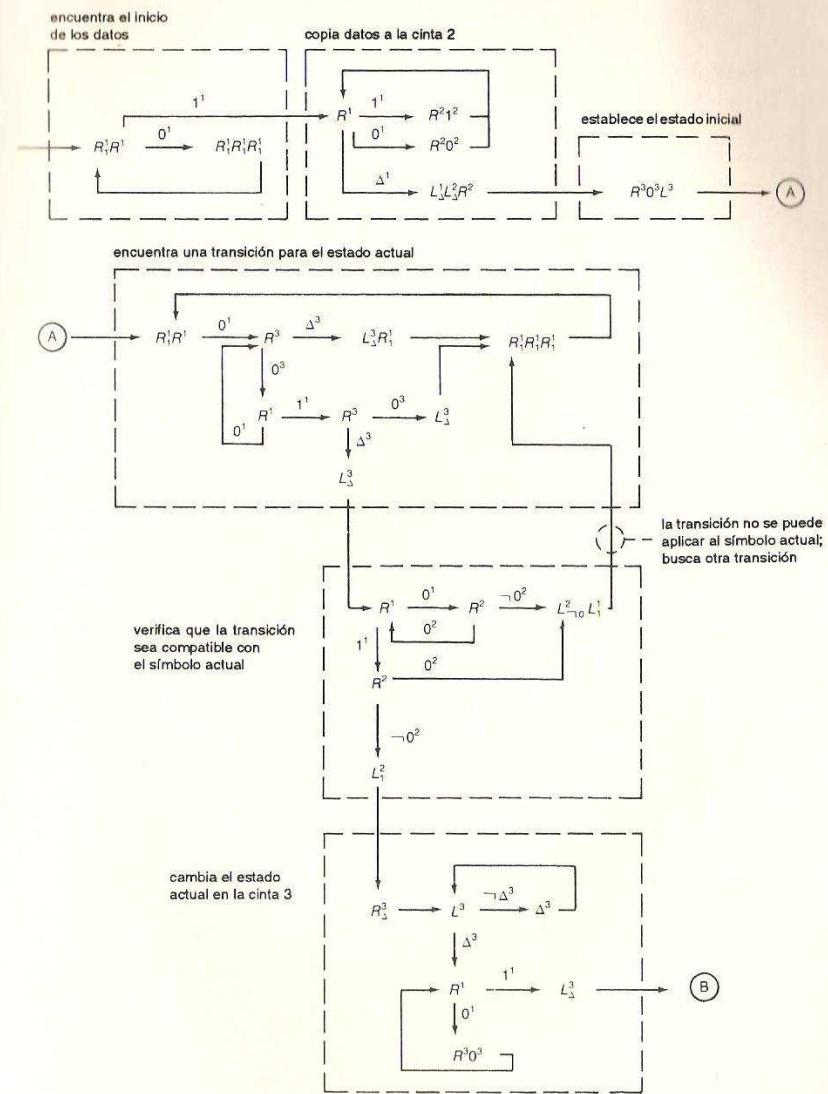
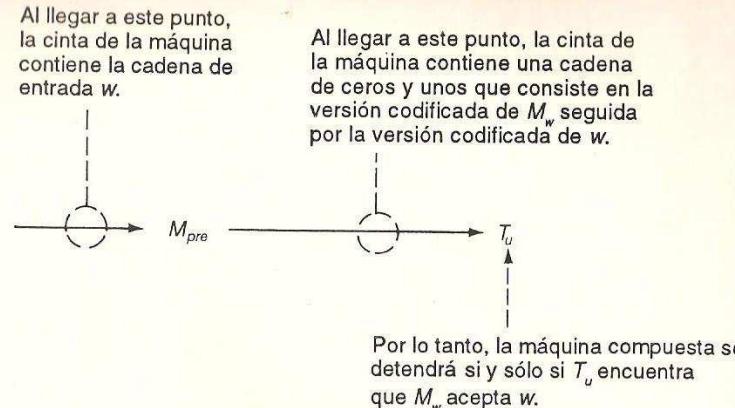


Figura 3.24 Máquina de Turing universal (continúa en la página siguiente)

Jorge Rafael Cruz de León
INGENIERO EN SISTEMAS DE INFORMACIÓN
Y CIENCIAS DE LA COMPUTACIÓN
Colegio de Bachilleres No. 6 492

Figura 3.25 Aplicación de la máquina $\rightarrow M_{pre} \rightarrow T_u$ a la cadena w

obtenidas por la concatenación de todas las cadenas de terminales de longitud uno que se pueden derivar de P con todas las cadenas de terminales de longitud uno que se pueden derivar de Q . Observe que estas cadenas de un solo terminal están registradas en las filas anteriores de la tabla). En la tercera fila de la tabla, bajo cada no terminal, registre todas las cadenas de terminales de longitud tres que pueden derivarse de dicho no terminal (si $N \rightarrow PQ$ es una regla de G , concatene todas las cadenas de terminales de longitud uno que pueden derivarse de P con cada una de las cadenas de terminales de longitud dos que se pueden derivar de Q . Luego, concatene todas las cadenas de terminales de longitud dos que pueden derivarse de P con las de longitud uno que se pueden derivar de Q . Una vez más, todas las cadenas que se deben concatenar se encuentran en las casillas de la tabla correspondientes a las filas por encima de la que estamos llenando.)

En general, registramos en la fila n bajo cada no terminal N todas las cadenas de terminales de longitud n que se pueden derivar de dicho no terminal. Estas cadenas se pueden encontrar localizando primero cada regla de la forma $N \rightarrow PQ$ y luego, para cada i en $\{1, 2, \dots, n-1\}$, concatenando cada cadena de terminales de longitud i derivable de P con cada cadena de terminales de longitud $n-i$ derivable de Q (véase Fig. 3.26). Continuamos con la construcción de esta tabla hasta completar la fila número $|w|$. Al llegar a este punto verificamos si w se ha registrado bajo el símbolo inicial de G . Si es así, entonces $w \in L$; de lo contrario, $w \notin L$. Por lo tanto, concluye el proceso de decisión.

Para realizar este procedimiento con una máquina de Turing, podemos usar una máquina con cintas múltiples que tenga una cinta más que el número de no terminales en G . La primera cinta se puede usar para almacenar la cadena de entrada y la respuesta afirmativa o negativa. Cada una de las cintas restantes se puede emplear para representar una columna de la tabla, separando con diagonales los elementos de distintas filas. Así, el proceso de decidir si

la cadena xyx se encuentra o no en el lenguaje generado por la gramática de la figura 3.26 produciría el contenido de las cintas que está representado en la figura 3.27, de donde la máquina puede determinar que xyx no se encuentra en el lenguaje.

Por supuesto, existen también lenguajes decidibles que no son independientes del contexto. Un ejemplo es el lenguaje $\{x^n y^n z^n : n \in \mathbb{N}\}$, que no es independiente del contexto pero sí puede ser decidido por la máquina de Turing de la figura 3.28.

Por último, debemos señalar que la terminología "decidible por máquinas de Turing" y "aceptado por máquinas de Turing" no es universal. Ya hemos visto que un lenguaje aceptado por máquinas de Turing es idéntico a un lenguaje estructurado por frases. Sin embargo, en otros contextos un lenguaje aceptado por máquinas de Turing se conoce como **lenguaje enumerable recursivamente**. Esta terminología se debe a que los lenguajes aceptados por máquinas de Turing son precisamente los lenguajes cuyas cadenas puede enumerar –o, en otras palabras, listar– una máquina de Turing. Además, en

$$\begin{aligned} S &\rightarrow MN \\ M &\rightarrow MP \\ N &\rightarrow PN \\ M &\rightarrow x \\ N &\rightarrow y \\ P &\rightarrow x \\ P &\rightarrow y \end{aligned}$$

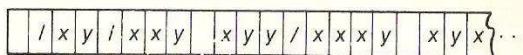
	S	M	N	P
1		x	y	x y
2	xy	xx xy	xy yy	
3	xx xy	xxx xxy xyx yyy	xx yxy xyy yyy	
4	xxx xxy xyy yyy	$xxxx$ $xxxy$ $xyxx$ $yyxx$ $xyyx$ $yyxy$ $xyyy$ $yyyy$	xx yxy xyy yyy	

Figura 3.26 Gramática independiente del contexto y primeras cuatro filas de la tabla construida a partir de ella

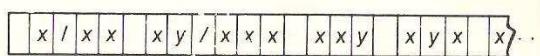
cinta uno, que contiene la cadena de entrada



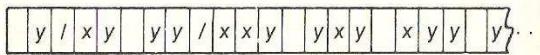
cinta dos, que representa la columna S



cinta tres, que representa la columna M



cinta cuatro, que representa la N



cinta cinco, que representa la columna P



Figura 3.27 Realización con cinco cintas del proceso de construcción de la tabla de la figura 3.26

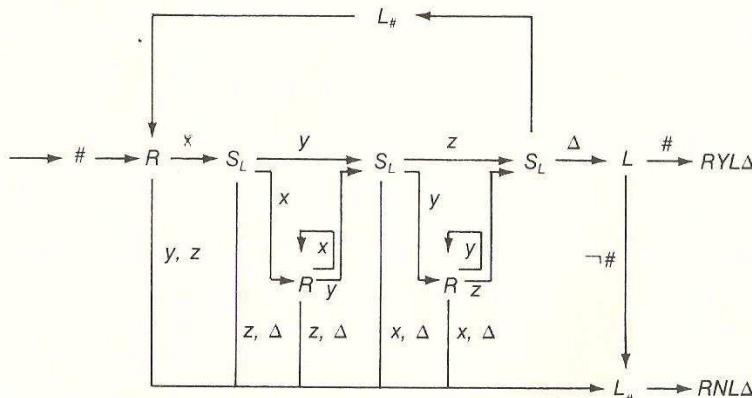


Figura 3.28 Máquina de Turing que decide el lenguaje $\{x^n y^n z^n : n \in \mathbb{N}\}$

en aquellas situaciones en las cuales se habla de lenguajes enumerables recursivamente, por lo general se hace referencia a los lenguajes decidibles por máquinas de Turing como lenguajes recursivos.

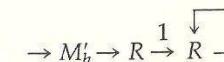
El problema de la parada

Damos fin a esta sección con un ejemplo de otro lenguaje que no es decidible por una máquina de Turing, no sólo porque precisemos otra muestra sino porque el caso que se presentará, conocido como problema de la parada, es un ejemplo clásico en la teoría de la computación.

Recuerde que cualquier máquina de Turing se puede codificar como una cadena de ceros y unos. Representemos con $p(M)$ esta versión codificada de una máquina de Turing M . Si ahora centramos nuestra atención en las máquinas con alfabeto $\{0, 1\}$ y símbolos de cinta $\{0, 1, \triangle\}$, entonces la cadena de código $p(M)$ se podría usar como entrada para la misma máquina M . No nos interesa si esta forma de usar M tiene relación con el fin para el cual se diseñó originalmente M . Lo único que nos importa es si M se detiene o no cuando inicia su operación con $p(M)$ como entrada. Si se detiene, decimos que M es autotérminante. De esta manera, cualquier máquina de Turing con alfabeto $\{0, 1\}$ y símbolos de cinta $\{0, 1, \triangle\}$ es o bien autotérminante o bien no autotérminante.

Definamos ahora el lenguaje L_h de $\{0, 1\}^*$ como $\{\rho(M) : M \text{ es auteterminante}\}$ y preguntémonos si L_h es decidable por máquinas de Turing. Para efectuar una decisión con respecto a L_h , se requiere la capacidad para detectar si una cadena de $\{0, 1\}^*$ es la versión codificada de una máquina auteterminante, lo que en esencia es el problema de decidir si una máquina se para cuando se le aplica una entrada específica; por consiguiente, el problema de decisión de L_h se conoce como **problema de la parada**.

Por desgracia, el lenguaje L_h no es decidable según Turing. Para convencernos de esto, supongamos lo contrario, que existe una máquina de Turing M_h que decide L_h . Entonces, modificaríamos M_h para obtener otra máquina M'_h que responda con el mensaje 1 en caso de que M_h hubiera respondido Y o 0 cuando la respuesta de M_h fuera N . Como en la demostración del teorema 3.6, podemos establecer que los símbolos de cinta que necesita M'_h consisten sólo en $\{0, 1, \Delta\}$. Así, se podría emplear M'_h para construir la máquina compuesta



con símbolos de cinta $\{0, 1, \triangle\}$, que se detiene si y sólo si M'_h llega a su estado de parada con salida 0. Representemos con M_0 a esta máquina compuesta.

A partir de aquí, nuestro argumento gira alrededor de la pregunta ¿es M_0 auteterminante o no auteterminante? Puesto que se trata de una máquina de Turing con símbolos de cinta $\{0, 1, \Delta\}$, debe ser una cosa o la otra. Supongamos que es auteterminante. Entonces, al recibir la entrada $p(M_0)$, M'_h se detendrá con salida 1. Esto quiere decir que M_0 no se detendría si iniciara con la entrada $p(M_0)$. (La ejecución de M_0 recorrería el arco $R \xrightarrow{1} R$ y quedaría atrapada en el proceso infinito de movimiento de la cabeza a la derecha. Véase Fig. 3.29.) Sin embargo, ésta es la característica que define a una máquina que no es auteterminante. Entonces, si suponemos que M_0 es auteterminante, llegamos a la contradicción de que no es auteterminante.

1. Si M_0 fuera auteterminante y se pusiera en marcha con la entrada $p(M_0)$,

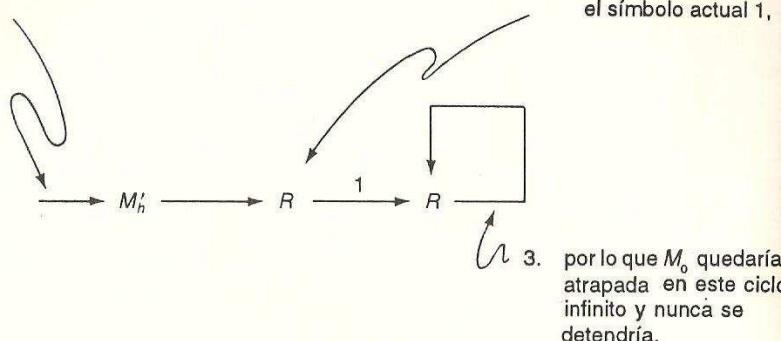


Figura 3.29 Ejecución de la máquina M_0 con entrada $p(M_0)$ bajo el supuesto de que M_0 es auteterminante

1. Si M_0 no fuera auteterminante y se pusiera en marcha con la entrada $p(M_0)$

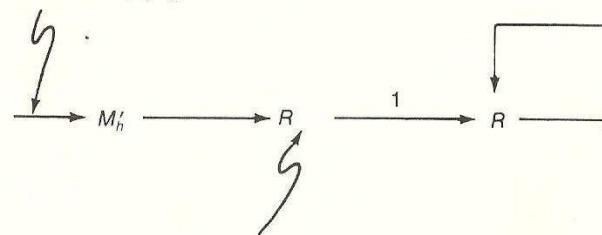


Figura 3.30 Ejecución de la máquina M_0 con entrada $p(M_0)$ bajo el supuesto de que no es auteterminante

Puesto que M_0 debe ser auteterminante o no auteterminante, al llegar a este punto debemos concluir que no es auteterminante. Empero, si M_0 no fuera auteterminante, entonces, al recibir la entrada $p(M_0)$, M'_h se detendría con salida 0. Esto quiere decir que M_0 se detendría si iniciara con la entrada $p(M_0)$.

(La ejecución de M_0 no podría recorrer el arco $R \xrightarrow{1} R$. Véase Fig. 3.30.) Sin embargo, ésta es la característica que define a una máquina auteterminante. Por lo tanto, también debe ser falsa nuestra suposición de que M_0 no es auteterminante.

Hemos llegado a la inconsistencia de una máquina M_0 con símbolos de cinta $\{0, 1, \Delta\}$ que no es auteterminante ni no auteterminante, cuando la máquina debe ser una u otra cosa. Por consiguiente, nos vemos obligados a concluir que nuestra suposición inicial es falsa; es decir, el lenguaje L_h no es decidable por máquinas de Turing.

Ejercicios

- Diseñe una máquina de Turing que acepte exactamente aquellas cadenas de ceros y unos que sean representaciones válidas de máquinas de Turing deterministas, de acuerdo con el sistema de codificación descrito en esta sección.

- Identifique la paradoja que existe en el siguiente enunciado:

El cocinero de una comunidad aislada cocina para aquellas personas, y sólo aquellas personas, que no cocinan para sí mismas.

(Sugerencia: ¿Quién cocina para el cocinero?) ¿Cómo se relaciona esta pregunta con esta sección?

- Diseñe una máquina de Turing que acepte las cadenas del lenguaje $L = \{w^n : w = xy\}$ al detenerse con su cinta configurada como $\underline{\Delta}Y\underline{\Delta}\Delta\Delta\cdots$ y rechace las cadenas que no se hallan en L al detenerse con la configuración $\underline{\Delta}N\underline{\Delta}\Delta\Delta\cdots$.

Explique por qué no puede construirse esta máquina en el caso de todos los lenguajes aceptados por máquinas de Turing.

- Presente un argumento de que cualquier lenguaje que contenga un número finito de cadenas es decible por máquinas de Turing.
- Amplíe la tabla de la figura 3.26 para solucionar la pregunta de si la cadena $xyyyxy$ es generada por la gramática.

3.6 COMENTARIOS FINALES

En este capítulo y los anteriores presentamos una jerarquía de lenguajes junto con los autómatas que se requieren para aceptarlos. Se trata de una variante de la jerarquía de lenguajes conocida como jerarquía de Chomsky (nombrada así

en honor de N. Chomsky, pionero del desarrollo de la teoría de lenguajes formales durante la década de los cincuenta). En la figura 3.31 se presenta un resumen de la jerarquía que hemos presentado. Mostramos así mismo que cada nivel de esta jerarquía contiene propiamente el siguiente nivel inferior. De hecho, en cada nivel proporcionamos un ejemplo explícito de un lenguaje que reside en ese nivel pero no en el inferior. En este momento, un excelente ejercicio de repaso sería recopilar estos ejemplos e incluirlos en el diagrama de la figura 3.31.

Uno de los principales resultados de este capítulo ha sido la presentación de la tesis de Turing, la cual, traducida al contexto de la figura 3.31, nos dice que un sistema computacional nunca podrá efectuar un análisis sintáctico de aquellos lenguajes que se encuentran más allá de los lenguajes estructurados por frases. Como se señaló al final de la sección 3.4, esta tesis tiene grandes repercusiones. Por esto, no es ninguna sorpresa que la tesis de Turing haya llamado la atención de muchos investigadores, quienes han tratado de apoyar sus afirmaciones o de desafiar su validez.

En el próximo capítulo veremos algunos de los resultados de las actividades de estos investigadores. Por ahora sólo señalaremos que la amplia gama de terminología que hemos empleado (como "lenguajes aceptable por máquinas de Turing", "lenguajes estructurados por frases" y "lenguajes recursivamente enumerables") es el resultado del alcance de la tesis de Turing. En varias disciplinas se han observado y estudiado los mismos límites aparentes del poder de los procesos computacionales que surgen de la hipótesis de Turing, incluso cuando cada quien ha atacado el problema de la computabilidad desde una perspectiva distinta y con diferente terminología.

Problemas de repaso del capítulo

1. Con los bloques de construcción presentados en la sección 3.2, construya una máquina de Turing compuesta que convierta su cinta de la configuración $\triangle w \triangle \triangle \triangle \cdots$, donde w es cualquier cadena de x y y , en la configuración $\triangle w \triangle v \triangle \triangle \triangle \cdots$, donde v es la cadena w escrita a la inversa.
2. Con los bloques de construcción presentados en la sección 3.2, construya una máquina de Turing compuesta que forme la concatenación de dos cadenas v y w en $\{x, y\}^*$ convirtiendo su cinta de la configuración $\triangle v \triangle w \triangle \triangle \triangle \cdots$ en $\triangle v w \triangle \triangle \triangle \cdots$.
3. ¿Cómo puede el resultado de la ejecución de $\rightarrow RL$ diferir del de $\rightarrow LR$?
4. Dibuje un diagrama de transiciones para la máquina de Turing compuesta $\rightarrow R_x \triangle L$.

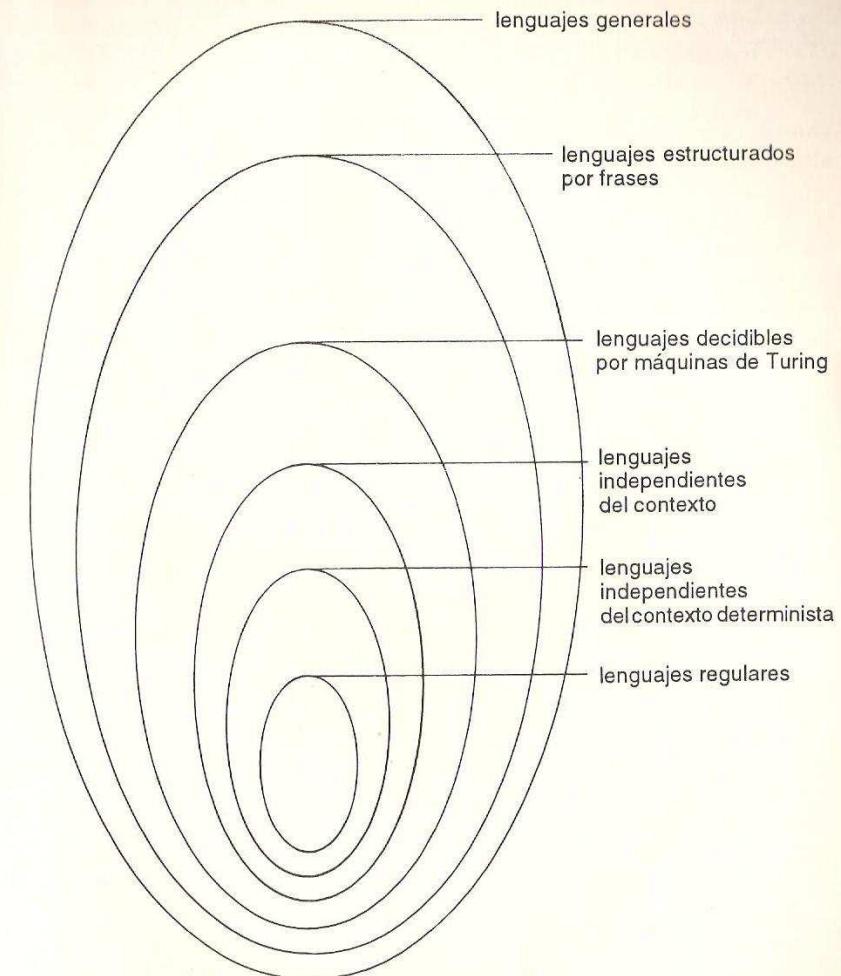
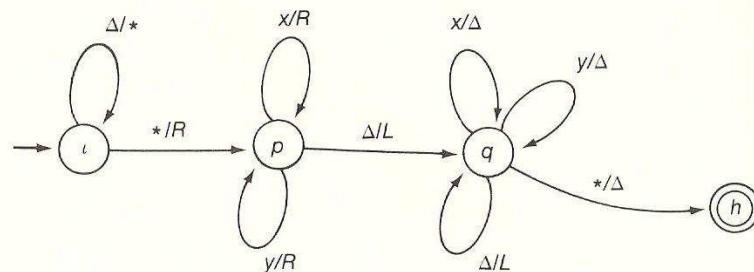


Figura 3.31 Variante de la jerarquía de lenguajes de Chomsky

5. Diseñe una máquina de Turing M tal que $L(M) = \{x^n y^{2n} z^n : n \in \mathbb{N}\}$.
6. Diseñe una gramática G estructurada por frases tal que $L(G) = \{x^m y^n x^m y^n : m, n \in \mathbb{N}\}$.
7. Diseñe una gramática G estructurada por frases tal que $L(G) = \{x^n y^{2n} z^n : n \in \mathbb{N}\} \cup \{x^n y^n z^n : n \in \mathbb{N}\}$.

8. Utilizando la notación para representar toda la configuración de una máquina de Turing, como la que se usó en la demostración del teorema 3.3, rastree la ejecución de la máquina compuesta de la figura 3.3 al iniciar con la configuración de cinta $\Delta\Delta yx\Delta\Delta\Delta\cdots$.
9. Utilizando la notación para representar toda la configuración de una máquina de Turing, como la que se usó en la demostración del teorema 3.3, rastree la ejecución de la máquina cuyo diagrama de transiciones se presenta a continuación, suponiendo que la configuración inicial de la cinta es $\Delta yx\Delta\Delta\cdots$.



10. ¿Es independiente del contexto el lenguaje generado por la gramática siguiente (con símbolo inicial S)? Describa el lenguaje.

$$\begin{aligned} S &\rightarrow xN \\ N &\rightarrow Sx \\ xNx &\rightarrow y \end{aligned}$$

11. ¿Es independiente del contexto el lenguaje generado por la gramática siguiente (con símbolo inicial S)? Describa el lenguaje.

$$\begin{aligned} S &\rightarrow SPQR \\ S &\rightarrow \lambda \\ QP &\rightarrow PQ \\ PQ &\rightarrow QP \\ PR &\rightarrow RP \\ RP &\rightarrow PR \\ QR &\rightarrow RQ \\ RQ &\rightarrow QR \\ P &\rightarrow x \\ Q &\rightarrow y \\ R &\rightarrow z \end{aligned}$$

12. Diseñe una máquina de Turing M tal que $L(M) = \{x, y, z\}^* - \{x^n y^n z^n : n \in \mathbb{N}\}$.

13. Muestre que si el lenguaje L es aceptado por máquinas de Turing, entonces existe una máquina de Turing que termina anormalmente si y sólo si su cadena de entrada se encuentra en L .
14. Si el lenguaje L de Σ es aceptable según Turing, ¿existe alguna máquina de Turing que se detenga si su entrada está en L y termine anormalmente si su entrada en $\Sigma^* - L$? Explique su respuesta.
15. a. ¿Forma la unión de un número finito de lenguajes estructurados por frases un lenguaje estructurado por frases? Justifique su respuesta.
 b. ¿Forma siempre la unión de una colección de lenguajes estructurados por frases un lenguaje estructurado por frases? Justifique su respuesta.
16. a. ¿Forma la intersección de un número finito de lenguajes estructurados por frases un lenguaje estructurado por frases? Justifique su respuesta.
 b. ¿Forma siempre la intersección de una colección de lenguajes estructurados por frases un lenguaje estructurado por frases? Justifique su respuesta.
17. Podemos expresar una gramática como una cadena de símbolos en donde las reglas de reescritura se separan con diagonales ($S \rightarrow xS / S \rightarrow \lambda$). Diseñe una máquina de Turing que acepte únicamente las cadenas que representen gramáticas independientes del contexto con terminales del conjunto $\{x, y\}$ y no terminales de $\{S, M, N\}$.
18. Diseñe una gramática G que no sea independiente del contexto con la cual $L(G)$ sea el lenguaje independiente del contexto $\{x^n y^n : n \in \mathbb{N}\}$.
19. Muestre que para cada lenguaje L estructurado por frases existe una cantidad infinita de gramáticas que generan L .
20. Diseñe una máquina de Turing (quizás con varias cintas) que ordene alfabéticamente una lista de cadenas de $\{x, y, Z\}^*$. Suponga que las cadenas de la lista de entrada están separadas por un espacio en blanco.
21. Muestre que la colección de lenguajes no estructurados por frases tiene mayor cardinalidad que la colección de los lenguajes estructurados por frases (por lo tanto, existen más lenguajes no aceptados por máquinas de Turing que lenguajes que sí lo son).
22. Muestre que para cualquier alfabeto Σ existe un lenguaje L de Σ tal que ni L ni $\Sigma^* - L$ son aceptados por máquinas de Turing.

23. Muestre que la colección de lenguajes decidibles por máquinas de Turing para un alfabeto cualquiera es infinita, pero contable.
24. Muestre que si una máquina de Turing M acepta cada cadena w en $L(M)$ sin tener que ejecutar más de $|w| + 1$ pasos, entonces $L(M)$ debe ser regular.
25. ¿Es un subconjunto de un lenguaje estructurado por frases siempre un lenguaje estructurado por frases? Explique su respuesta.
26. Muestre que la estrella de Kleene de un lenguaje aceptado por máquinas de Turing es también aceptable según Turing.
27. Muestre que el lenguaje $\{x^n : n \text{ es un entero primo positivo}\}$ es aceptado por máquinas de Turing. ¿Es decidable? ¿Por qué?
28. Muestre que el lenguaje $\{x^n : n \in \mathbb{N}\}$ es aceptado por máquinas de Turing. ¿Es decidable? ¿Por qué?
29. Dibuje un diagrama de transiciones para una máquina de Turing que decida el lenguaje $\{x^m y^n : m, n \in \mathbb{N} \text{ y } m \geq n\}$.
30. Construya, con los bloques de construcción presentados en la sección 3.2, una máquina de Turing que acepte el lenguaje $\{x^m y^n x^m y^n : m, n \in \mathbb{N}\}$.
31. Aplique el algoritmo de construcción de tablas descrito en la sección 3.5 para decidir si la cadena $xxyyxxxyyy$ se encuentra en el lenguaje generado por la gramática siguiente, cuyo símbolo inicial es S .

$$\begin{aligned} S &\rightarrow SS \\ S &\rightarrow MN \\ N &\rightarrow SP \\ M &\rightarrow x \\ N &\rightarrow y \\ P &\rightarrow y \end{aligned}$$

¿Se encuentran también las cadenas $xxxyyxyy$, $xyxyxyxy$ y $xyxxxxyy$?

32. Amplíe la tabla de la figura 3.26 para determinar si la cadena $xyyxyyyy$ es generada por la gramática de dicha figura.
33. Suponga que $p(n)$ es una expresión polinomial en n y que M es una máquina de Turing que acepta toda cadena w de $L(M)$ sin tener que ejecutar más de $p(|w|)$ pasos. Muestre que el lenguaje $L(M)$ es decidable.

34. Muestre que todo lenguaje aceptado por máquinas de Turing puede generarse a partir de una gramática estructurada por frases en la que ninguna regla de reescritura tiene una terminal como parte de su lado izquierdo.
35. Muestre que para cualquier lenguaje L recursivamente enumerable existe una máquina de Turing que, al ponerse en marcha con una cinta en blanco, comenzará a generar en su cinta una lista de las cadenas en L , de manera que cada una de ellas aparecerá tarde o temprano en la cinta.
36. Muestre que todo lenguaje enumerable recursivamente infinito contiene un lenguaje recursivo infinito.

Problemas de programación

1. Desarrolle un simulador de máquinas de Turing. Diseñe este simulador de manera que se proporcione en forma tabular la descripción de la máquina que se simulará, para que pueda reemplazarse fácilmente con la descripción de otra máquina.
2. Escriba un programa que simule la máquina de Turing universal de tres cintas descrita en este capítulo.
3. Escriba un programa para decidir si la cadena que se proporcione como entrada se halla o no en el lenguaje $\{x^m y^n z^m : m, n \in \mathbb{N}\}$, aplicando el algoritmo de construcción de tablas descrito en la sección 3.5.