Máquinas abstractas y gramáticas formales

Al hablar aquí de *máquinas* se está haciendo referencia a autómatas, es decir, dispositivos formales capaces de exhibir conductas que han sido previamente determinadas. El calificativo de *abstractas* proviene del hecho que estas máquinas no necesariamente existen como objetos materiales. Son conceptualizaciones que se utilizan para el diseño, la implementación y evaluación de algoritmos, el análisis de su complejidad, la evaluación de los recursos necesarios para determinado cómputo, la simulación de su comportamiento sobre nuevos microprocesadores, inclusive todavía no construidos, y la identificación de eventuales conflictos en procesos concurrentes o paralelos, por citar algunos ejemplos típicos.

Por su parte, para hablar de *gramáticas formales* antes es necesario distinguir a los *lenguajes naturales* (castellano, inglés, etc.) de los *formales*, de los cuales los lenguajes de programación son un ejemplo. Los primeros evolucionan permanentemente según usos y costumbres de los pueblos y las épocas, con gramáticas que deben esforzarse para explicarlos y justificarlos. Por el contrario, los lenguajes formales son desarrollados a partir de gramáticas previamente establecidas y no admiten forma alguna de excepción o desviación. Los lenguajes de programación fueron propuestos como consecuencia de la presentación de los primeros computadores y de la necesidad de comunicarse con ellos de manera inequívoca, ya que se trata de un diálogo riguroso que no admite las excepciones ni los errores de interpretación que son provocados por las ambigüedades.

La teoría de los lenguajes y las gramáticas formales se vio revolucionada por el lingüista norteamericano Noam Chomsky en 1956, con la presentación de su trabajo *"Teoría de las Gramáticas Transformacionales".* Este autor estableció las bases de la lingüística matemática y proporcionó una herramienta que fue inmediatamente aprovechada en la formalización de los entonces incipientes lenguajes de computación. A partir de allí, los lenguajes de computación experimentaron gran rapidez en su evolución, hasta alcanzar un nivel de abstracción que hoy permite la especificación de los datos, funciones y su control en forma independiente de los computadores que tendrán a cargo la ejecución de los procesos. Estos son los denominados lenguajes de alto nivel y constituyen las herramientas cotidianas de desarrollo e implementación de sistemas. Entre las gramáticas formales y las máquinas abstractas, hay un fuerte vínculo que se estudiará en los sucesivos capítulos de este libro.

Volviendo ahora a las máquinas abstractas, se trata de sistemas reactivos que operan en respuesta a los sucesivos estímulos recibidos del exterior, que llevan a los mismos a adoptar condiciones características que son denominadas ***estados*** y eventualmente a enviar una respuesta al medio exterior. Por estas razones, los dispositivos de este tipo suelen también ser denominados *máquinas de estados* y son encuadrados como de *estímulo-respuesta*. Nótese así que la conducta de una de estas máquinas depende tanto del estímulo recibido como del estado en el que se encuentra en ese momento. Puede citarse el ejemplo del modelo de un ascensor, que al recibir como estímulo la orden de ir al piso seis se pondrá en marcha hacia arriba si se encuentra en planta baja o hacia abajo si está en un piso superior. El estado de esta máquina está asociado al piso en el que se encuentra y su respuesta dependerá de ese estado y la orden recibida que indica el destino deseado.

El concepto de máquina admitió a lo largo del tiempo sucesivas interpretaciones y formas de implementación según los recursos tecnológicos disponibles en cada época. La primera máquina sumadora de la historia desarrollada por Blaise Pascal (1640), precursora de las actuales calculadoras, era capaz de realizar operaciones tales como la adición y la sustracción mediante ingeniosos mecanismos de engranajes. Mucho más recientemente, Claude Shannon (1938) implementó centrales telefónicas con circuitos eléctricos que incluían dispositivos electromecánicos tales como contactores, relés y temporizadores. En la actualidad, los autómatas son implementados a través de software y en muchos casos deben comunicarse con el medio exterior, para lo que se valen de circuitos electrónicos integrados, sensores y actuadores. Puede citarse como ejemplo el caso de un equipo electrodoméstico, el horno de microondas, que: *i*) recibe estímulos del usuario a través de selectores o teclas y del sensor de apertura de puerta, *ii*) incorpora un reloj para medir intervalos de tiempo y *iii*) genera un campo electromagnético de variada intensidad con el fin de calentar los alimentos. El horno de microondas es un autómata que opera adoptando sucesivos estados a

partir de estímulos recibidos del exterior y su conducta responde a una secuencia de opciones preestablecidas.

Alan Turing, precursor de la ciencia de la computación, propuso en 1936 una máquina abstracta que en la actualidad lleva su nombre, en una época en que la electrónica todavía no se había desarrollado y la materialización de estas ideas eran en ese momento impracticables. Poco tiempo antes, en 1931, el matemático alemán Kurt Gödel había impactado a la comunidad científica de la época al presentar su *Teorema de Incompletitud*, con el que demostró que toda formulación axiomática consistente, capaz de contener la teoría de los números y la aritmética, contendría proposiciones que no se podrían demostrar ni negar. El teorema de Gödel fue un gran golpe a las esperanzas de los matemáticos de formalizar toda su disciplina a partir de un sistema axiomático general, tal como había sido propuesto por Hilbert en el primer cuarto del siglo XX.

La máquina de Turing fue un formalismo propuesto con la finalidad de disponer de una herramienta abstracta para estudiar la teoría de la computabilidad. En su trabajo de 1936, Alan Turing desarrolló las ideas de Gödel y demostró que existen problemas irresolubles, que son aquellos que están fuera del alcance de su máquina. En efecto, la máquina de Turing estaba en capacidad de resolver cualquier problema que tuviese solución. Aquí es necesario acotar que otros matemáticos, tales como Emil Post y Alonzo Church, presentaron en ese mismo año enfoques distintos pero con similares conclusiones, que más tarde pudo demostrarse que eran equivalentes al de Turing, aunque en el momento de su enunciación pasaron inadvertidos, no teniendo la repercusión que tuvo la propuesta de Turing.

Los trabajos de Gödel, Turing, Church y Post despertaron un gran interés por los métodos algorítmicos y los matemáticos comenzaron a desarrollar máquinas teóricas y autómatas, estudiando sus capacidades y limitaciones, a pesar de que, como ya se dijo, la tecnología no estaba aún suficientemente desarrollada como para posibilitar una implementación eficaz. En este contexto, se suma el ya mencionado aporte de Claude Shannon, que en su tesis de maestría en MIT (1938) demostró cómo el álgebra booleana se podía utilizar en el análisis y la síntesis de la conmutación y de los circuitos digitales. La tesis de Shannon es considerada como uno de los mayores aportes en la historia, al diseño de circuitos digitales y estimuló el estudio de teorías formalizadas de las máquinas secuenciales y de los autómatas finitos, mucho más simples que las máquinas de Turing.

Características y formalismos de las máquinas abstractas

Con la máquina presentada por Alan Turing (1936) y los autómatas finitos que estaban implícitos en las ideas de Claude Shannon (1938), se dispuso de las máquinas abstractas más complejas y las más sencillas que pueden ser construidas, quedando en el medio toda una familia de máquinas que surgen de introducir sucesivas limitaciones a la máquina de Turing o de incorporar nuevos recursos al autómata finito.

Aquí cabe acotar que, con anterioridad en 1907, Andréi Markov ya había planteado la idea de un proceso formalizado al presentar las cadenas que llevan su nombre. Posteriormente, en 1943, McCulloch y Pitts modelaron artificialmente el funcionamiento de una neurona a través de una máquina abstracta, que presenta gran similitud con los autómatas finitos.

El análisis de las características y propiedades específicas de las distintas máquinas, denominadas genéricamente *abstractas*, será uno de los objetos de este libro. Para comenzar, puede anticiparse que todas ellas comparten las siguientes características comunes:

- Reconocen que el tiempo avanza y lo hace en forma discreta, es decir, de intervalo en intervalo.
- En cada uno de esos intervalos de tiempo una máquina se encuentra en un cierto y único estado. El conjunto de sus estados posibles es finito y está agrupado en un conjunto o *alfabeto de estados*.
- La máquina recibe información o estímulos del medio exterior, para lo cual se incorpora el concepto de una *cinta de entrada*. En cada intervalo de tiempo, la máquina lee un símbolo de la cinta de entrada y el conjunto de todos los símbolos reconocidos por la máquina es agrupado en un conjunto, llamado *alfabeto de entrada*.

- En algunos casos, las máquinas de estados actúan también sobre el medio exterior, grabando símbolos sobre una cinta de salida. Estos símbolos pertenecen a un *alfabeto de salida* y según el tipo de máquina, la cinta de salida puede ser la misma que la de entrada o un medio distinto.
- Las lecturas y grabaciones en cinta son realizadas en cada intervalo de tiempo por *cabezales* apropiados. Estos cabezales se van desplazando sobre los medios de entrada y salida, distinguiéndose las diferentes máquinas según los cabezales se muevan siempre secuencialmente en un mismo sentido o sea posible gobernar el sentido de su próximo movimiento.
- Las aptitudes de las máquinas, y en definitiva su propia esencia, están reunidas en lo que se denomina la función de transición. En esta función, se define, para cada símbolo de entrada y para cada estado posible, el próximo estado a ser adoptado. Además, en el caso en que la máquina gobierne el cabezal, esta función debe también definir el sentido de su movimiento, que lo llevará a desplazarse sobre la cinta una posición a la izquierda o una posición a la derecha. Finalmente, si la máquina tiene salida, en la función de transición, debe definirse el símbolo a ser grabado en la cinta correspondiente.

Aquí es oportuno comentar la adopción de *cintas* como medios de entrada y salida de las máquinas abstractas. Al igual que en la máquina de Turing, en las primeras épocas de la computación las cintas (inicialmente de papel perforado y posteriormente magnéticas) fueron el medio habitual para leer y grabar datos, acciones que estaban a cargo de cabezales sobre los que se desplazaban las cintas. En realidad, estas *cintas* que son utilizadas como medios genéricos de entrada y salida, podrían estar representando teclados, tarjetas *chip*, puertos de comunicaciones, pantallas o medios magnéticos más modernos, como pueden ser las tarjetas de identificación o los discos rígidos. Es así que, más allá de su implementación práctica, las cintas representan la posibilidad de que una máquina abstracta lea datos del medio exterior y genere salidas sobre el mismo, sin entrar en detalles tecnológicos.

Otro aspecto que debe ser comentado es el de la función de transición. Si tal como se dijo arriba, esta función definiera el próximo estado de la máquina para cada símbolo de entrada y para cada estado posible, se trataría de una función total o completa. Es decir, para cada elemento del dominio representado por el producto cartesiano de las entradas y los estados, está definido el correspondiente elemento del codominio, que es un próximo estado. Sin embargo, en muchos casos, esto no es práctico, ya que resulta innecesario definir la función de transición para condiciones en las que la máquina de estados no se va a encontrar nunca y, por este motivo, se admite que se trate de una función parcial. Se recuerda que una función es parcial cuando no todos los elementos del alcance están asociados con algún elemento del rango (el alcance no coincide con el dominio). Esto significa que hay condiciones de operación en las que la máquina abstracta no tiene definido el próximo estado de operación y en el hipotético caso de que llegara a una de estas condiciones se detendría. Las posibles condiciones de detención de las máquinas de estados se analizan con más detalle en los capítulos respectivos.

El hecho de que aquí se admita que las funciones de transición de todas las máquinas abstractas sean parciales es un punto muy importante y debe ser comentado, ya que este no es un criterio unánime en la literatura. En efecto, hay algunos autores que le atribuyen a la máquina de Turing una función de transición parcial y al autómata finito le asignan una función de transición completa, lo que además de plantear una inconsistencia, impone a la definición de los autómatas finitos una complicación innecesaria. La aclaración es conceptualmente importante, pero operativamente el problema es poco trascendente, ya que nada impide que a todas las condiciones no definidas de una función de transición parcial se les asigne un estado de error, lo que convierte a la función de transición en completa.

Aclarado este tema, es necesario reconocer que las máquinas de estados admiten diferentes clasificaciones, las que se plantean a partir de sus características operativas o de los recursos que tienen disponibles. Si bien hay numerosos criterios de clasificación, se describen a continuación los que son tratados en este libro por ser los más habituales:

Máquinas traductoras y máquinas reconocedoras

Se denominan *máquinas traductoras* a aquellas que establecen una relación entre las cadenas de entrada y las cadenas de salida. Por el contrario, hay máquinas cuya misión es validar o aceptar ciertas cadenas de entrada, arribando a estados finales definidos como ***de aceptación***, sin producir ninguna cadena de salida. Las máquinas abstractas de este tipo llevan el nombre de *reconocedoras.*

Autómatas finitos y máquinas secuenciales

Los *autómatas finitos* comienzan su operación en un estado conocido, que es denominado *estado inicial*, y completan su ciclo arribando a un *estado de aceptación* que ha sido oportunamente identificado. En esta tarea, emplean una cantidad finita de intervalos de tiempo, por lo que su comportamiento es algorítmico. Por el contrario, las *máquinas secuenciales* no tienen previsto estados de aceptación y en algunos casos, tampoco tienen definidos estados de inicio de su operación. Esto significa que las *máquinas secuenciales* operan en forma ininterrumpida, sin reconocer condiciones específicas para iniciar y terminar su actividad.

Autómatas deterministas y no deterministas

Las funciones de transición de los *autómatas deterministas* contemplan un único próximo estado a partir de cada estado posible y cada símbolo del alfabeto de entrada. Es decir, la conducta del autómata está completamente determinada, y de allí su nombre. Por el contrario, en los *autómatas no deterministas* la función de transición puede prever más de un próximo estado para cierta condición de estado y entrada, lo que provoca una indeterminación en el funcionamiento de la máquina. Otro rasgo que puede diferenciar a los *autómatas no deterministas* es la posibilidad de realizar transiciones de un estado a otro sin necesidad de leer ningún símbolo de entrada.

Según el criterio adoptado, un *autómata finito* cuyo próximo estado está indefinido en alguna condición de operación, está representado por una función de transición parcial y es determinista. En efecto, llegado a esta condición el autómata inevitablemente se detendrá. Aquí tampoco hay un criterio unánime en la literatura y hay autores que consideran que si un *autómata finito* tiene una función de transición parcial es no determinista.

Tal como fueron definidos, es fácil advertir que los *Autómatas Finitos Deterministas* (AFD) son un caso particular de los *Autómatas Finitos No Deterministas* (AFND). Otro aspecto a considerar es que, en algunos casos, el comportamiento de los autómatas no deterministas puede ser representado por autómatas deterministas, llamados equivalentes, mientras que con algunos otros tipos de autómatas esto no es posible. Así, todo *autómata finito no determinista* tiene siempre un *autómata finito determinista* equivalente, mientras que en el caso de los *autómatas con memoria de pila* esta equivalencia no necesariamente existe.

Automatismos y autonomía

Llegado a este punto, en que se ha hecho una descripción introductoria y general de las máquinas abstractas, es conveniente avanzar sobre los conceptos de autómata o automatismo y de autonomía, que con frecuencia son confundidos.

Los autómatas exhiben comportamientos automáticos, es decir que han sido establecidos con anticipación al momento de su diseño y construcción, y su modificación está fuera del alcance del propio autómata. Son automatismos que para ser dotados de un comportamiento diferente deben ser rediseñados. En otras palabras, en la conducta de un sistema automático no habrá nada que no haya sido oportunamente previsto por su creador.

Por el contrario, autonomía implica la capacidad de alterar por sí solo el propio comportamiento y esto surge de la interacción del sistema con su entorno. Los sistemas autónomos comienzan a operar como automáticos y se distinguirán de estos últimos de manera progresiva, a lo largo de su operación. Para que una máquina de estados tenga comportamiento autónomo debe ser capaz de alterar su función de transición para incorporar nuevas conductas en forma dinámica, o más aún, para incorporar el reconocimiento de nuevos estímulos. Es decir, ampliar su alfabeto de entrada. Las máquinas que exhiban autonomía pueden ser calificadas como *inteligentes*.

A partir de lo expuesto, se rescatan tres ideas centrales: i) el comportamiento de los sistemas automáticos está completamente predeterminado, ii) para que un sistema sea autónomo debe ser antes automático y iii) en los sistemas autónomos el comportamiento es una propiedad emergente de la interacción del sistema y su ambiente. En la Teoría de la Computación, y en este libro, se tratan autómatas que exhiben comportamiento automático mientras que el estudio de los sistemas autónomos pertenece al campo de la Inteligencia Artificial.

Jerarquía de máquinas y gramáticas

Hasta aquí se ha discutido brevemente lo que se entiende por máquinas abstractas y se han descrito, en forma muy general, algunas de sus principales clasificaciones. Para continuar, se identificarán las principales máquinas a ser estudiadas y luego se tratará el vínculo entre estas máquinas y las diferentes gramáticas reconocidas por la jerarquía de Chomsky.

Como ya fue anticipado, los *autómatas finitos* y la *máquina de Turing* representan los dos extremos de una jerarquía creciente en complejidad y capacidad. Si se comienza por el *autómata finito* se puede incrementar su capacidad de cómputo incorporándole una memoria *LIFO*, obteniéndose el *autómata con pila*. Como alternativa, pueden lograrse soluciones más eficientes permitiendo el movimiento del cabezal de entrada en dos sentidos con el *autómata finito bidireccional*. Esta máquina, a su vez, mejorará su capacidad de cómputo al poder grabar sobre el medio de entrada, dando lugar al *autómata linealmente acotado*. Finalmente, esta evolución conduce a la *máquina de Turing* al liberarse los extremos de la cinta y hacerla infinita, todo lo cual se representa en la Figura 1.1. Nótese que la *máquina de Turing* es capaz de resolver todo problema que tenga solución y, por lo tanto, representa el límite natural de lo computable.

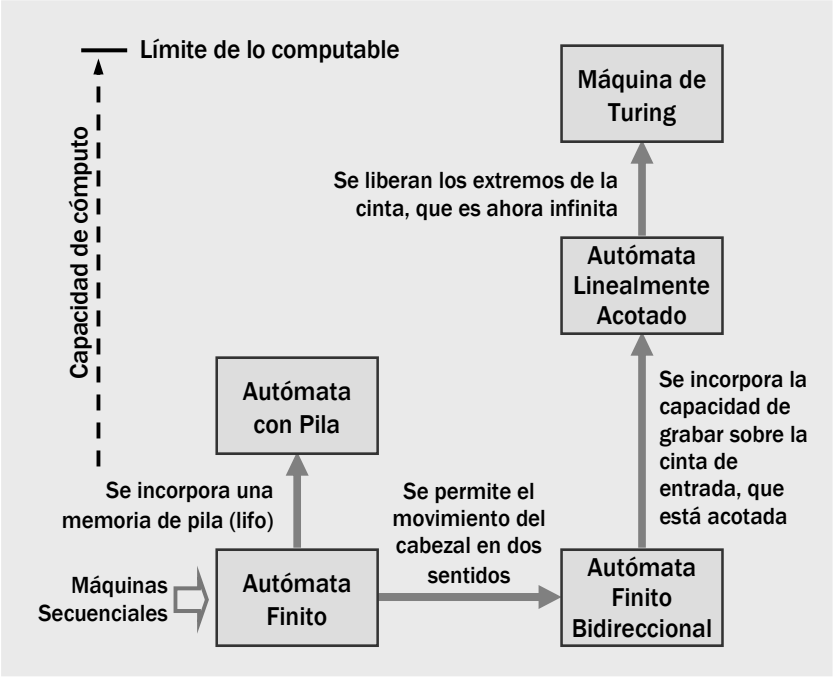

Figura 1.1: Jerarquía de máquinas abstractas.

Al remitirse a su historia se comprueba que las máquinas abstractas presentadas hasta ahora tuvieron como objetivo inicial el análisis de la computabilidad de funciones, así como el cálculo y evaluación de la complejidad de algoritmos. Con posterioridad, el interés se orientó hacia el estudio de la potencialidad de las máquinas como reconocedoras de lenguajes, lo que estuvo estimulado por el desarrollo de compiladores.

Es en esta última aplicación que importa precisar el diferente rol que tienen las gramáticas y las máquinas o autómatas: las primeras son metalenguajes destinados a la generación de los lenguajes formales y las últimas son modelos de entidades reconocedoras de esos lenguajes. El lenguaje es el vínculo que establece un isomorfismo entre ambas, como se muestra en la Figura 1.2.

Recuérdese que se habla de isomorfismo entre dos estructuras cuando el estudio de cada una puede reducirse al de la otra, lo que brinda dos puntos de vista diferentes sobre una misma

cuestión. El concepto matemático de isomorfismo (del griego isomorfos: igual forma) transmite la idea de estructuras equivalentes.

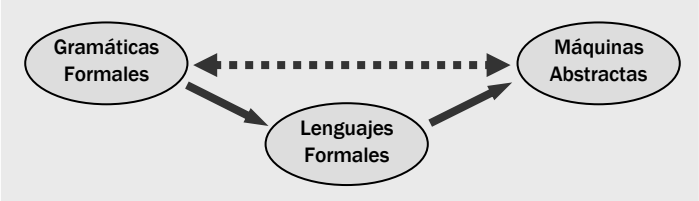

Figura 1.2: Isomorfismo entre gramáticas y autómatas.

Ahora debe anticiparse que Chomsky utilizó las formas que toman las reglas de producción de las gramáticas para proponer una clasificación en cuatro tipos, en las que las menos restringidas incluyen sucesivamente a las siguientes, lo que queda resumido en la Figura 1.3.

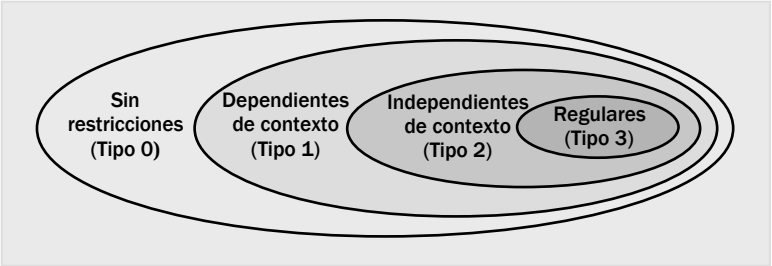

Figura 1.3: Jerarquía de gramáticas de Chomsky.

Es interesante comprobar que en los diez años que siguieron a la presentación de las *Gramáticas Generativas* (1956) se completó el vínculo entre la jerarquía de Chomsky y la familia de máquinas abstractas, que se representa en la Figura 1.4 y que se describe a continuación.

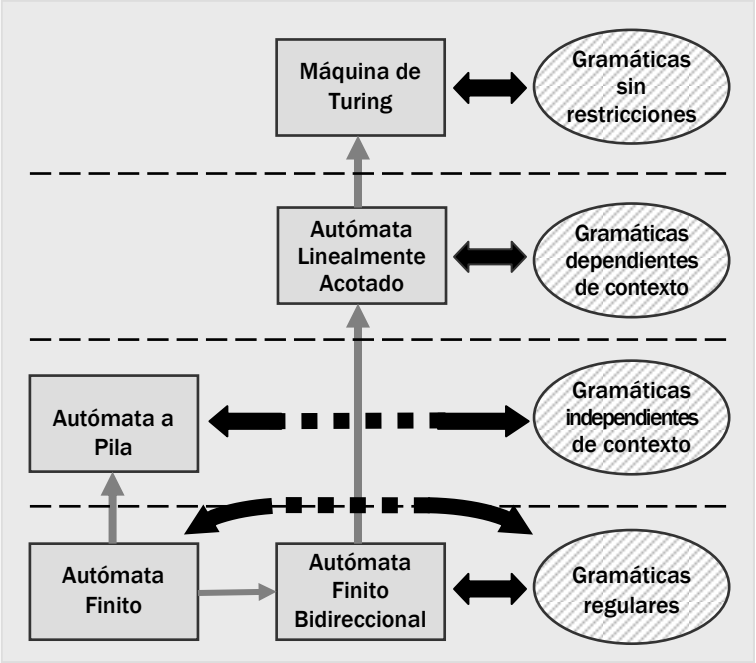

Figura 1.4: Relaciones entre las máquinas y los tipos de gramáticas según la jerarquía de Chomsky.

En 1958, el propio Chomsky y Miller comprobaron que las *gramáticas regulares* (Tipo 3) están directamente relacionadas con los autómatas finitos; luego un año después Chomsky estableció la equivalencia entre las *gramáticas sin restricciones* (Tipo 0) y las máquinas de Turing. Hubo que esperar hasta 1963 para que se demostrara el vínculo entre las *gramáticas independientes del contexto* (Tipo 2) y los autómatas con pila, por Chomsky y Schutzenberger y, por último, Kuroda, en 1964, descubrió que los lenguajes generados por las *gramáticas dependientes del contexto* (Tipo 1) son reconocidos por los autómatas linealmente acotados.

Resumen del vínculo entre máquinas y gramáticas

Todo lo dicho hasta ahora con referencia a las máquinas reconocedoras, sus características y sus vínculos con las gramáticas formales, puede ser resumido con unos esquemas muy simples, pero que han demostrado ser muy eficaces con los alumnos en clase. Para comenzar, se representa en la Figura 1.5 un autómata finito y su gramática asociada.

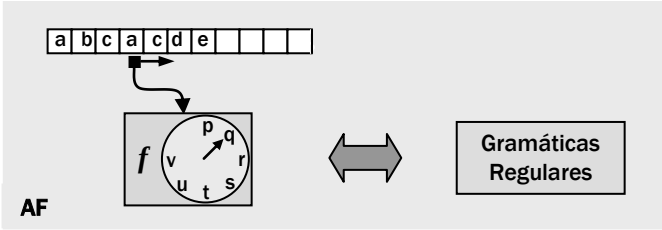

Figura 1.5: Autómata finito y gramáticas regulares.

El autómata muestra sus principales componentes, que son: un conjunto de estados (p, q, r, etc.), su función de transición ***f*** , una cinta que contiene símbolos del alfabeto de entrada (a, b, c, d, etc.) y su cabezal de lectura, que se mueve en un mismo sentido con el fin de leer secuencialmente la cadena a ser procesada.

Incorporando al autómata finito una memoria *LIFO* (*Last In First Out*, es decir, lo último que ingresa es lo primero que sale) se obtiene un nuevo autómata denominado *con pila*. Al proponer este autómata es necesario definir el alfabeto vinculado a la pila, que puede tener símbolos en común con el alfabeto de entrada, y en el que se debe identificar un símbolo destinado a señalar el fondo de la pila (en el ejemplo de la figura, se utiliza el #). Naturalmente, la función de transición debe ser ampliada para incorporar como argumento el símbolo leído de la pila y para prever la salida sobre ella. El autómata con pila está relacionado con las gramáticas independientes (o libres) del contexto y es representado en la Figura 1.6.

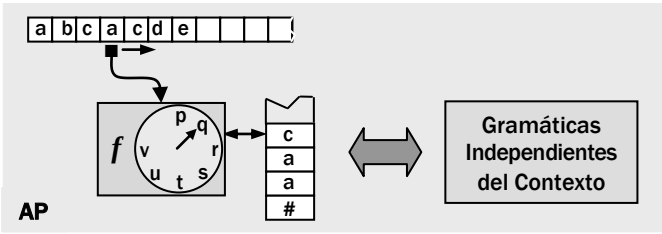

Figura 1.6: Autómata con pila y gramáticas independientes del contexto.

Una opción para desarrollar nuevos autómatas es volver al autómata finito de la Figura 1.5 y posibilitar que el sentido del movimiento del cabezal de entrada sea controlado por la propia máquina, obteniéndose el *autómata finito bidireccional*. Obviamente, también aquí la función de transición debe ser ampliada para incorporar la definición del movimiento del cabezal. Además, la cadena de datos a ser procesada es contenida entre dos símbolos especiales (├ y ┤) destinados a limitar el movimiento del cabezal sobre la cinta.

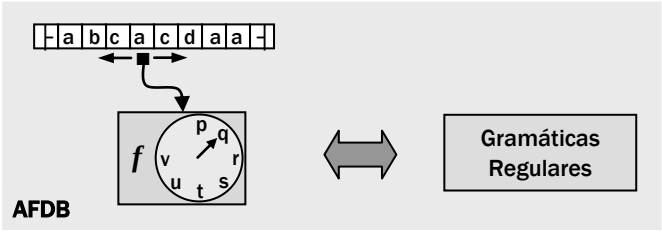

Figura 1.7: Autómata finito bidireccional y gramáticas regulares.

Sin embargo, a pesar de esta mejora, el autómata finito bidireccional no incrementa su capacidad y se mantiene vinculado a las gramáticas regulares, tal como se muestra en la Figura 1.7.

El próximo recurso a ser considerado, es la posibilidad de grabar sobre la misma cinta de entrada, con lo que se obtiene una nueva máquina que incrementa notablemente su capacidad y es denominado *autómata linealmente acotado*. Esto implica ampliar nuevamente la función de transi-

ción para incorporarle la definición del símbolo que es grabado en cada caso. Es decir que esta función ahora define el próximo estado, el símbolo grabado y el sentido del movimiento del cabezal. Este nuevo autómata está relacionado con las gramáticas dependientes del contexto y es representado en la Figura 1.8.

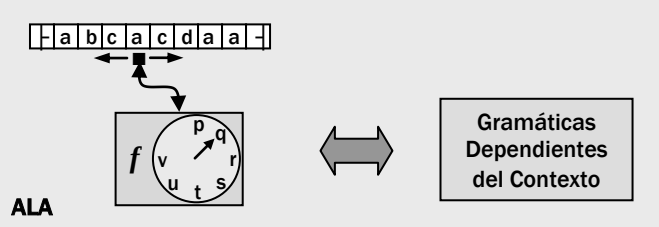

Figura 1.8: Autómata linealmente acotado y gramáticas dependientes de contexto.

El último paso en esta progresiva búsqueda de autómatas de mayor capacidad, consiste en eliminar los límites en los extremos de la cinta de entrada / salida, permitiendo el acceso a un espacio ilimitado que puede ser usado como un área auxiliar de almacenamiento. Se llega así a la *máquina de Turing*, mostrada en la Figura 1.9, que como ya se dijo con anterioridad es capaz de resolver cualquier problema que tenga solución, es decir que sea computable. Como es de esperar, esta máquina está relacionada con todas las gramáticas, incluyendo la última en la escala de Chomsky que no tiene restricciones.

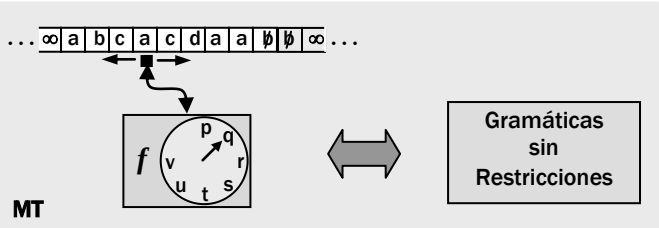

Figura 1.9: Máquina de Turing y gramáticas sin restricciones.

¿Para qué sirven las máquinas abstractas?

Invariablemente, a poco de que la primera máquina abstracta es presentada en clase, surge de parte de los alumnos una pregunta tan básica como fundamental: ¿… y eso para qué sirve? Sin ánimo de agotar las posibles respuestas, se brindan a continuación algunos de los motivos que justifican el estudio de las máquinas abstractas, dando prioridad a aplicaciones de interés de la ingeniería de software y en sistemas de información.

Construcción de compiladores

Los *compiladores* están destinados a convertir programas escritos en algún lenguaje de nivel superior en otros programas en lenguaje máquina. En los inicios de la computación, los programas eran escritos directamente en lenguaje máquina, pero rápidamente se advirtió la necesidad de hacerlo en lenguajes con mayor poder expresivo, más acordes a la naturaleza del razonamiento humano. Se inventaron así los primeros lenguajes superiores de programación e inmediatamente debieron desarrollarse los conversores para llevarlos al nivel de las máquinas. Nótese que no se trata de una traducción de un lenguaje a otro, sino de una verdadera conversión por existir una importante diferencia en las capacidades de abstracción de los lenguajes. Esta conversión implica el entendimiento y validación del programa superior, llamado programa fuente, y su reescritura en el lenguaje de máquina, llamado programa objeto, el cual debe ser semánticamente equivalente al programa fuente (debe hacer lo mismo).

Este desafío de disponer de una alternativa más eficiente que la del lenguaje ensamblador para programar computadoras llevó a John Backus a proponer el lenguaje **Fortran** (***For****mula* ***Tran****slation*) y desarrollar su compilador entre los años 1953 y 1957. Nació así el primer lenguaje superior de programación de la era de la computación.

Para la época en la que Backus desarrollaba el Fortran, aparece un aporte inesperado del campo de la lingüística: las gramáticas generativas de Noam Chomsky, que si bien estaban destinadas al tratamiento de los lenguajes naturales (inglés) rápidamente fueron adoptadas para la

formalización de los lenguajes de computación. Estas nuevas ideas enriquecieron el formalismo que venía utilizando Backus basado en las producciones de Emil Post y se aplicaron en el desarrollo del siguiente lenguaje superior, que fue el Algol (***Algo****rithmic* ***L****anguage*, 1960). Estos novedosos progresos en el campo de las gramáticas estimularon y favorecieron el desarrollo progresivo de nuevos lenguajes de programación, hasta llegar a los que hoy se usan habitualmente.

¿Y qué tiene esto que ver con las máquinas abstractas? Bien, para la identificación de los componentes (palabras) del lenguaje se utilizan autómatas finitos, que toman la forma de analizadores léxicos. Para su validación gramatical (sintáctica), se utilizan analizadores sintácticos, que son implementados a través de autómatas con pila que operan a partir de las gramáticas de los lenguajes estudiados. Finalmente, para la verificación de consistencia (semántica), hay diferentes enfoques, algunos de los cuales están también a cargo de autómatas con pila. Así, los autómatas finitos y autómatas con pila son una parte de las máquinas abstractas que serán estudiadas en detalle en los siguientes capítulos y que encuentran en el desarrollo e implementación de compiladores algunas de sus más importantes aplicaciones.

Procesamiento del lenguaje natural

El procesamiento del lenguaje natural consiste en el análisis e interpretación de los aspectos lingüísticos de un mensaje a través de programas informáticos. Algunas de las principales aplicaciones son la traducción automática, los correctores ortográficos en los procesadores de textos, la comunicación hombre-máquina, la enseñanza asistida por computadora, la recuperación de información, el procesamiento del habla, la elaboración automática de resúmenes de documentos y la generación automática de índices. La actividad en este amplio campo se inició tan pronto se dispuso de los primeros computadores, involucrando a los ya conocidos Alan Turing, Claude Shannon y Noam Chomsky entre otros, y continúa siendo objeto de intensa investigación en nuestros días.

Según María Felisa Verdejo, prestigiosa investigadora de este campo, el lenguaje natural se distingue de los lenguajes artificiales en cinco aspectos fundamentales: *i*) riqueza en vocabulario y construcciones, *ii*) flexibilidad al admitir reglas con múltiples excepciones, *iii*) ambigüedad, pudiendo darse diversos significados a una palabra o a una frase según el contexto, *iv*) indeterminación, permitiendo referencias y elipsis, y *v*) posibles interpretaciones del sentido literal según la situación en la que se produce el mensaje. Lo que son ventajas para la comunicación humana se convierten en problemas a la hora de un tratamiento computacional, ya que implican conocimiento y procesos de razonamiento que son muy difíciles de formalizar.

En la actualidad, el campo del procesamiento del lenguaje natural lleva el nombre de *Lingüística Computacional* y reconoce la imposibilidad de analizar e interpretar una frase sin comprenderla, concepto universalmente admitido hoy. De esta manera, se llega así a un enfoque sintáctico-semántico donde, una vez analizada su estructura (sintaxis), se efectúa una interpretación semántica en un tratamiento integral que se denomina *teoría de la dependencia conceptual*. Al igual que en el caso de los compiladores, las máquinas abstractas constituyen el núcleo de los sistemas de la lingüística computacional, tomando la forma de analizadores morfológicos o léxicográficos, analizadores sintácticos y analizadores semánticos.

Modelado de sistemas

La construcción de modelos es una actividad fundamental en el desarrollo de buen software y tiene numerosas finalidades: comprender mejor el problema que se desea resolver, especificar un comportamiento deseado, comunicar una estructura y descubrir oportunidades de simplificación y reutilización, entre otras. Un modelo es una simplificación de la realidad, que será capaz de cumplir su finalidad en la medida que rescate las características esenciales del objeto estudiado. Por ello, al desarrollarse un modelo se debe trabajar en sus tres dimensiones ortogonales, que son: funcional, estática y dinámica.

En la actualidad, el lenguaje estándar universalmente aceptado para el modelado de sistemas es el UML (*Unified Modeling Language*) que, entre otras herramientas, prevé: *i*) *Casos de Uso* para describir la dimensión funcional, *ii*) *Diagramas de Clases* para la dimensión estática y *iii*) *Diagramas de Secuencia* y *Diagramas de Estados* para cubrir la dimensión dinámica. Los *diagramas de secuencia* describen las interacciones entre objetos ordenadas en una secuencia temporal

y los *diagramas de estados* describen el comportamiento de un objeto individual, es decir, las sucesivas condiciones (estados) en las que puede encontrarse un cierto objeto y las acciones que provocan las transiciones de uno a otro de esos estados.

Estos diagramas de estados previstos por UML son, con algunas variantes, la expresión gráfica de algunas de las máquinas abstractas, máquinas secuenciales y autómatas finitos, a las que ya se hizo referencia y que serán tratadas en profundidad en este libro. Cabe acotar que UML incorpora variantes a las máquinas de estados, entre las que se destacan: *i*) los estados jerarquizados o estados compuestos, que contienen a su vez estructuras internas en niveles inferiores que pueden ser arbitrariamente complejas, *ii*) las llamadas *regiones ortogonales*, que se utilizan cuando un comportamiento es fragmentado en conductas independientes, pero que a su vez son concurrentes, y *iii*) su representación gráfica. Sería muy interesante y útil hacer un análisis comparativo entre ambas, pero esto queda fuera del alcance de este libro.

Implementación de algoritmos

Parece natural que, si el comportamiento de un objeto ha sido descripto a través de una máquina de estados, a la hora de programar este desempeño, es decir convertir el comportamiento deseado en código, se implemente esta misma máquina. Más aún, las máquinas de estados son siempre una opción conveniente en la implementación de algoritmos, prescindiendo de las técnicas utilizadas en su análisis y diseño. Esto es debido a que las máquinas de estados representan una forma muy compacta de representar conjuntos complejos de reglas y condiciones que deben responder a variadas condiciones de entrada. Además, permiten una importante reducción en la demanda de recursos y el código resultante queda autodocumentado.

Un buen programa debe ser un producto de calidad y los atributos de la calidad no son fáciles de alcanzar: corrección, confiabilidad, integridad, eficiencia, facilidad de uso, entre otros. Además, un nuevo programa deberá ser testeado antes de entrar en servicio y mantenido con posterioridad, acciones ambas que son definitivamente dependientes de la claridad con la que ha sido expresado el código. Sin lugar a dudas, las máquinas de estados son una opción muy conveniente, en algunos casos indispensables, para alcanzar todos estos atributos que distinguen un buen software.

Sistemas embebidos

Los sistemas embebidos están destinados a fines muy específicos, son programados en lenguajes de bajo nivel o nivel intermedio, operan normalmente en ausencia de sistemas operativos y se orientan a aplicaciones de tiempo real. La ausencia del sistema operativo requiere que el software de aplicación opere directamente sobre los recursos del hardware: bancos de registros, niveles de interrupción y puertas de entrada/salida. Las características ya señaladas en el punto anterior referidas a códigos compactos, muy eficientes y de baja demanda de recursos hacen que las máquinas de estados sean preferidas en la implementación de este tipo de aplicaciones.

Identificación de patrones y virus informáticos

La identificación de patrones admite diferentes enfoques, entre ellos, el sintáctico. Éste se basa en utilizar la teoría de lenguajes formales para encontrar similitudes en las relaciones estructurales que guardan los objetos de estudio. El objetivo es construir una gramática que describa la estructura del universo de objetos y una máquina abstracta que reconozca esos lenguajes.

Entre las muchas aplicaciones de la identificación de patrones pueden citarse: *i*) la identificación del idioma que corresponde a determinado texto, *ii*) el agrupamiento de noticias según las distintas secciones de un diario, *iii*) la identificación de transacciones telefónicas o de tarjetas de crédito sospechosas de fraude, *iv*) el análisis de la efectividad de las campañas de márketing o *v*) la comprobación de la presencia de virus informáticos.

Los virus informáticos merecen un comentario aparte. A mediados del siglo pasado, John von Neumann trabajó en la demostración de la factibilidad de la autoréplica (autorreproducción) de elementos no biológicos, es decir que elementos artificiales puedan multiplicarse por sí mismos, sin perder sus cualidades ni degradarse. Para estos estudios, von Neumann utilizó máquinas de Turing y autómatas celulares, pero murió antes de completar sus demostraciones. Es sorprendente comprobar lo cerca que estuvo de anticipar conceptualmente, a través de sus teorías, los ele-

mentos de la célula que muchos años después serían revelados como el código genético. Obviamente, en esa época, poco se sabía sobre el ADN. Posteriormente, en los años setenta, en los laboratorios Xerox PARC (*Palo Alto Research Center*), se confirmaron las predicciones de von Neumann al ser autorreplicado por primera vez un elemento no biológico, en este caso una pieza de software. Este enorme progreso en el campo de la Ciencia de la Computación tuvo una consecuencia inesperada: orientó el futuro desarrollo de los virus informáticos.

Sistemas industriales

Las máquinas secuenciales y los autómatas finitos tienen una amplia aplicación en el campo industrial. Numerosos sistemas en la industria tienen a estas máquinas como modelo de diseño y sus conductas están, por lo tanto, regidas por ellas. Es así que se encuentran máquinas abstractas tanto en productos industriales como así también en dispositivos de montaje y en herramientas de producción. Como ejemplos típicos de los primeros, se encuentran los equipos electrodomésticos, tales como los lavarropas y los hornos de microondas.

Entre los últimos, pueden mencionarse los robots manipuladores, máquinas herramientas, cintas transportadoras, elevadores, máquinas envasadoras, balanzas automáticas, hornos industriales y un sinnúmero de otros dispositivos que asocian su comportamiento a condiciones de operación, o estados, los que son inmediatamente asimilables a máquinas abstractas y permiten definir secuencias de trabajo tan complejas como sean necesarias.

Para ello, estas máquinas adoptarán sucesivos estados y transitarán de uno a otro estado según los estímulos o señales recibidas del entorno que los rodea. Estos estímulos provienen de pulsadores instalados en las consolas de comando y de sensores de presencia de diversos tipos, tales como es el caso de las barreras luminosas o sensores magnéticos de proximidad. A su vez, las salidas de estas máquinas son señales de control de los procesos.

Debe recordarse que ya se distinguieron con anterioridad los conceptos de automatismo y autonomía, y aquí cabe reafirmar que todos los dispositivos industriales citados, tales como las balanzas o los robots manipuladores, deben ser reconocidos como autómatas. Es decir, exhiben conductas definidas con anticipación y su alteración está fuera del alcance del propio dispositivo.

//