Lenguaje formal

En matemáticas, lógica, y las ciencias computacionales, un **lenguaje formal** es un conjunto de palabras (cadenas de caracteres) de longitud finita formadas a partir de un alfabeto (conjunto de caracteres) finito.

Informalmente, el término *lenguaje formal* se utiliza en muchos contextos (en las ciencias, en derecho, etc.) para referirse a un modo de expresión más cuidadoso y preciso que el habla cotidiana. Hasta finales de la década de 1990, el consenso general era que un lenguaje formal, en el sentido que trata este artículo, era en cierto modo la versión «límite» de este uso antes mencionado: un lenguaje tan formalizado que podía ser usado en forma escrita para describir métodos computacionales. Sin embargo, hoy en día, el punto de vista de que la naturaleza esencial de los lenguajes naturales (sin importar su grado de «formalidad» en el sentido informal antes descrito) difiere de manera importante de aquella de los verdaderos lenguajes formales (en el sentido estricto de este artículo) gana cada vez más adeptos.

Un posible alfabeto sería, digamos, ***{a, b}***, y una cadena cualquiera sobre este alfabeto sería, por ejemplo, ***ababba***. Un lenguaje sobre este alfabeto, que incluyera esta cadena, sería: el conjunto de todas las cadenas que contienen el mismo número de símbolos *a* que *b*, por ejemplo.

La **palabra vacía** (esto es, la cadena de longitud cero) es permitida y frecuentemente denotada mediante **ε** o **λ**. Mientras que el alfabeto es un conjunto finito y cada palabra tiene una longitud también finita, un lenguaje puede bien incluir un número infinito de palabras.

Algunos ejemplos varios de lenguajes formales:

- el conjunto de todas las palabras sobre $\{a, b\}$
- el conjunto $\{a^n$: *n es un número primo*$\}$
- el conjunto de todos los programas sintácticamente válidos en un determinado lenguaje de programación
- el conjunto de entradas para las cuales una particular máquina de Turing se detiene.

Los lenguajes formales pueden ser especificados en una amplia variedad de maneras, como:

- cadenas producidas por una gramática formal (ver Jerarquía de Chomsky)
- cadenas producidas por una expresión regular
- cadenas aceptadas por un autómata, tal como una máquina de Turing.

Varias operaciones pueden ser utilizadas para producir nuevos lenguajes a partir de otros dados. Supóngase que L_1 y L_2 son lenguajes sobre un alfabeto común.

Entonces:

- la *concatenación* L_1L_2 consiste de todas aquellas palabras de la forma *vw* donde *v* es una palabra de L_1 y *w* es una palabra de L_2
- la *intersección* $L_1\&L_2$ consiste en todas aquellas palabras que están contenidas tanto en L_1 como en L_2
- la *unión* $L_1|L_2$ consiste en todas aquellas palabras que están contenidas ya sea en L_1 o en L_2

- el *complemento* $\sim L_1$ consiste en todas aquellas palabras producibles sobre el alfabeto de L_1 que no están ya contenidas en L_1
- el *cociente* L_1/L_2 consiste de todas aquellas palabras v para las cuales existe una palabra w en L_2 tales que vw se encuentra en L_1
- la *estrella* L_1^* consiste de todas aquellas palabras que pueden ser escritas de la forma $W_1W_2...W_n$ donde todo W_i se encuentra en L_1 y $n \geq 0$. (Nótese que esta definición incluye a & epsilon en cualquier L^*)
- la *intercalación* $L_1^*L_1$ consiste de todas aquellas palabras que pueden ser escritas de la forma $v_1w_1v_2w_2...v_nw_n$ son palabras tales que la concatenación $v_1...v_n$ está en L_1, y la concatenación $w_1...w_n$ está en L_2

Una pregunta que se hace típicamente sobre un determinado lenguaje formal L es cuán difícil es decidir si incluye o no una determinada palabra v. Este tema es del dominio de la teoría de la computabilidad y la teoría de la complejidad computacional.

Por contraposición al lenguaje propio de los seres vivos y en especial el lenguaje humano, considerados lenguajes naturales, se denomina lenguaje formal a los lenguajes «artificiales» propios de las matemáticas o la informática, los lenguajes artificiales son llamados lenguajes formales (incluyendo lenguajes de programación). Sin embargo, el lenguaje humano tiene una característica que no se encuentra en los lenguajes de programación: la diversidad.

En 1956, Noam Chomsky creó la Jerarquía de Chomsky para organizar los distintos tipos de lenguaje formal.

LENGUAJES

Se considera el conjunto S^* que consta de todas las cadenas finitas de elementos del conjunto S. Existen muchas interpretaciones posibles de los elementos de S^*, según la naturaleza de S. Si se piensa en S como un conjunto de "palabras", entonces S^* se puede considerar como la colección de todas las "oraciones" posibles formadas con palabras de S. Por supuesto, tales "oraciones" no necesariamente tienen sentido ni una estructura evidente. Se puede pensar un lenguaje como una especificación completa, al menos en principio, de tres cosas. En primer lugar, debe existir un conjunto S con todas las "palabras" que se consideran parte del lenguaje. En segundo lugar, hay que designar un subconjunto de S" como el conjunto de las "oraciones con construcción adecuada" en el lenguaje.

Supóngase, a manera de ejemplo, que S consta de todas las palabras del español. La especificación de una oración con construcción adecuada implica todas las reglas de la gramática española; el significado de una oración queda determinado por esta construcción y por el significado de las palabras.

La oración "*Iba a la tienda Juan Jorge a cantar*"

Es una cadena en S" pero no una oración con una construcción adecuada. La disposición de los sustantivos y los verbos no es válida.

Por otro lado, la oración "Los sonidos azules se sientan y cruzan la pierna bajo la cima de la montaña" tiene una construcción adecuada pero carece completamente de sentido.

Para otro ejemplo, *S* podría constar de los enteros, los símbolos +, —, X y ÷, así como los paréntesis izquierdo y derecho. Se obtendrá un lenguaje si se designa como adecuadas las cadenas en *S*" que representen sin ambigüedades expresiones algebraicas con paréntesis.

Así, ((3 — 2) + (4 X 7)) ÷ 9 y (7 — (8 — (9 — 10))); son "oraciones" con construcción adecuada en este lenguaje.

Por otro lado, (2 — 3)) + 4, 4 — 3 — 2 y)2 + (3 —) X 4 no tienen una construcción adecuada. La primera tiene demasiados paréntesis, la segunda, pocos (no se sabe cuál resta se debe realizar primero) y la tercera tiene paréntesis y números completamente fuera de lugar.

Todas las expresiones con construcción adecuada tienen sentido, excepto las que implican la división entre cero.

El significado de una expresión es el número racional que representa. Así, el significado de ((2 — 1) ÷ 3) + (4 X 6) es 73/3, mientras que 2 + (3 ÷ 0) y (4 + 2) — (0÷ 0) no tienen sentido.

La disciplina que regula la construcción adecuada de las oraciones es la sintaxis de un lenguaje. La que se encarga del significado de las oraciones es la semántica de un lenguaje.

Gramáticas

Una gramática para estructura de expresiones *G* es una 4-ada $(V,S,v_o,\mapsto)$, donde *V* es un conjunto finito, *S* es un subconjunto de $V, v_0 \in V - S$ y $\mapsto$ es una relación finita en V^*. La idea es que *S* es, como ya se ha analizado, el conjunto de todas las "palabras" permitidas en el lenguaje, y *V* consta de S además de algunos otros símbolos. El elemento v_0 de *V* es un punto de partida para las sustituciones, que serán analizadas en breve. Por último la relación $\mapsto$ sobre *V** especifica los reemplazos permisibles, en el sentido de que, si $w \mapsto w'$ se puede reemplazar *w* con *w*' siempre que aparezca la cadena *w*, ya sea sola o como subcadena de alguna otra cadena. Tradicionalmente, a la proposición $w \mapsto w'$ se le llama producción de *G*. Entonces, *w* y *w*' son los lados izquierdos y derecho de la producción, respectivamente. Supóngase que ninguna producción de *G* tiene a la cadena vacía $\wedge$ como lado izquierdo. Es la $\mapsto$ relación de producción de *G*.

Esto parecería complicado, pero en realidad es una idea sencilla, como muestran los siguientes ejemplos.

Si $G = (V,S,v_o,\mapsto)$ es una gramática para la estructura de oraciones, *S* es el conjunto de símbolos terminales y $N = V$ — *S* es el conjunto de símbolos no termina obsérvese que $V = S \cup N$.

Ejemplo 1. Sea S = {Juan, Julia, maneja, corre, descuidadamente, rápido, frecuentemente), N = {oración, sujeto, predicado, verbo, adverbio) y sea V=$S \cup N$. Sea v_0=oración, y supóngase que la relación $\mapsto$ en V* queda descrita enumerando todas las producciones como sigue.

Oración $\mapsto$ sujeto predicado

Sujeto $\mapsto$ Juan

sujeto $\mapsto$ Julia

predicado $\mapsto$ verbo adverbio

verbo $\mapsto$ maneja

verbo $\mapsto$ corre

adverbio $\mapsto$ descuidadamente

adverbio $\mapsto$ rápido

adverbio $\mapsto$ frecuentemente

El conjunto S contiene todas las palabras permitidas en el lenguaje; N consta de las palabras que describen partes de la oración, pero que en realidad no están contenidas en el lenguaje.

Se afirma que la oración "Julia maneja frecuentemente", que será denotada por w, es una oración permisible o con sintaxis correcta, de acuerdo con las reglas de este lenguaje. Para demostrar esto, considérese la siguiente serie de cadenas en V*. Oración.

sujeto	predicado	
Julia	predicado	
Julia	verbo	adverbio
Julia	maneja	adverbio
Julia	maneja	frecuentemente

Ahora bien, cada una de las cadenas es consecuencia de la anterior, utilizando una producción para realizar una sustitución parcial o completa. En otras palabras, cada cadena está relacionada con la siguiente cadena por la relación $\Rightarrow$, de modo que oración $\Rightarrow^{\infty}$ w. Por definición, w tiene una sintaxis correcta ya que, en este ejemplo, y es una oración. En las gramáticas para la estructura de oraciones, la búsqueda de una sintaxis correcta se refiere sólo al proceso mediante el cual al formar una oración se procura que ésta sea correcta gramaticalmente hablando, y nada más.

Hay que observar que la serie de sustituciones que producen una oración válida, serie que es llamada deducción (o derivación) de la oración, no es única.

La siguiente deducción produce la oración *w* del ejemplo 1, pero no es idéntica a la deducción dada antes. Oración

sujeto predicado

sujeto verbo adverbio

sujeto verbo frecuentemente

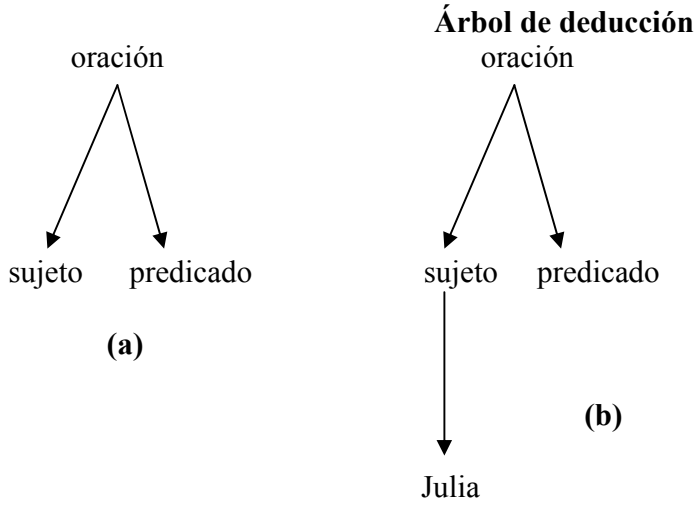
sujeto maneja frecuentemente

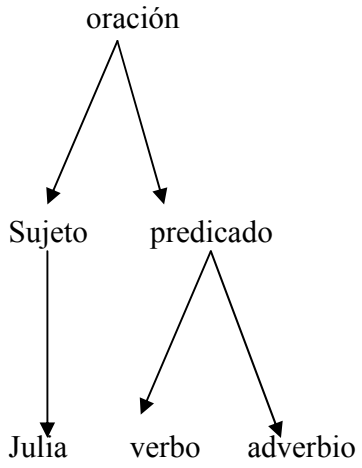
Julia maneja frecuentemente

El conjunto de todas las oraciones con construcción adecuada que pueden ser producidas aplicando las reglas de una gramática *G* es el lenguaje de *G* y se denota *L*(*G*). El lenguaje de la gramática del ejemplo 1 constituye un sublenguaje sencillo del español, y contiene precisamente 12 oraciones. El lector puede verificar que "Juan corre descuidadamente" está en el lenguaje *L*(*G*) de esta gramática, mientras que "Julia frecuentemente corre" no está en *L*(*G*).

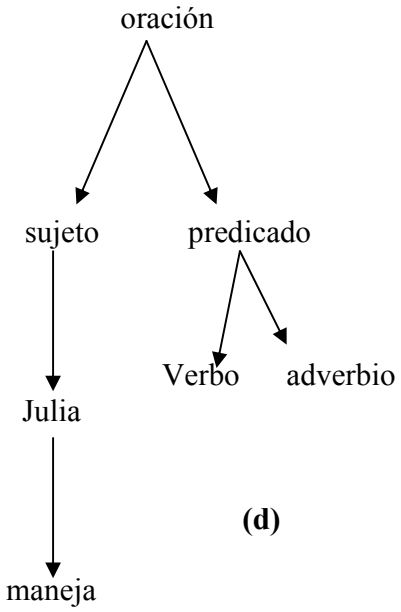
Diagrama de sintaxis

Un segundo método para desplegar las producciones de ciertas gramáticas de tipo 2 es el diagrama de sintaxis. Permite ver al usuario como un movimiento a través de diagramas

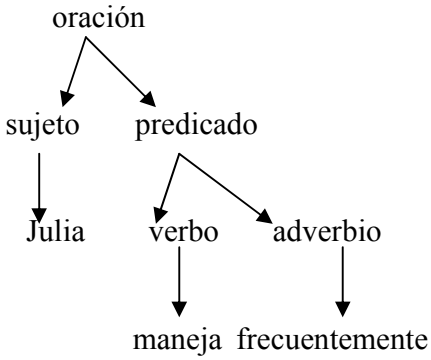




(c)



(d)



(e)

Tipo 0 si no se establecen restricciones sobre las producciones de *G*.

Tipo 1 si para cualquier producción $w_1 \mapsto w_2$, la longitud de w_1 es menor o igual que la longitud de w_2 (donde la longitud de la cadena es el numero de palabras en la cadena).

Tipo 2 si el lado izquierdo de cada producción es un único símbolo no terminal y el lado derecho consta de uno o mas símbolos.

Tipo 3 si el lado izquierdo de cada producción es un único símbolo no terminal y el lado derecho tiene uno o mas símbolos, incluyendo a lo mas un símbolo no terminal, que debe estar en el extremo derecho de la cadena.

REPRESENTACIONES DE LENGUAJES Y GRAMÁTICAS ESPECIALES

Notación BNF

Una alternativa que se encuentra con frecuencia es la flotación BNF (forma Backus-Naur). Se sabe que los lados izquierdos de todas las producciones en una gramática de tipo 2 son símbolos no terminales únicos. Para cada uno de tales símbolos w, se combina todas las producciones que tienen a w como lado izquierdo. El símbolo w permanece a la izquierda, y todos los lados derechos asociados con w son enumerados juntos, separados por el símbolo $\mid$. El símbolo $\mapsto$ relacional se reemplazan por el símbolo :: =. Por último, los símbolos no terminales, cuando aparezcan, serán encerrados entre paréntesis agudos $\langle\ \rangle$.

Ejemplo 1. En la notación BNF, las producciones del ejemplo 1 aparecen Como sigue.

$\langle$ oración $\rangle$ :: = $\langle$ sujeto $\rangle$ $\langle$ predicado $\rangle$

$\langle$ sujeto $\rangle$ :: = Juan $\mid$ Julia

$\langle$ predicado $\rangle$ :: = $\langle$ verbo $\rangle$ $\langle$ adverbio $\rangle$

$\langle$ verbo $\rangle$ :: = maneja $\mid$ corre

$\langle$ adverbio $\rangle$ :: = descuidadamente $\mid$ rápido $\mid$ frecuentemente

Ejemplo 2. En la notación BNF, las producciones aparecen como sigue.

$\langle v_0 \rangle$ ::= $a\langle\ w\ \rangle$

$\langle\ w\ \rangle$ ::= $bb\langle\ w\ \rangle \mid c$

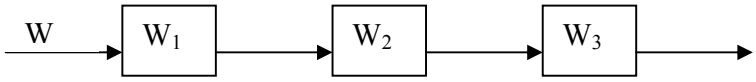
Observe que el lado izquierdo de una producción puede aparecer también en una de las cadenas del lado derecho. Así, en la segunda línea del ejemplo 2, (w) aparece a la izquierda, y aparece en la cadena $bb(w)$ de la derecha. Cuando esto ocurre, se dice que la producción correspondiente $w \mapsto bbw$ es recursiva. Si una producción recursiva tiene a w como lado izquierdo, la producción es normal si w aparece sólo una vez en el lado derecho y es el símbolo del extremo derecho. En el lado derecho también pueden aparecer otros símbolos no terminales.

Diagramas de sintaxis

Es una imagen de las producciones que permite al usuario ver las sustituciones en forma dinámica es decir verlas como un movimiento a través del diagrama. Son los diagramas de traducción de conjuntos de producciones típicos.

Un enunciado BNF que tiene una única producción, como

$\langle w \rangle :: = \langle w_1 \rangle \langle w_2 \rangle \ \langle w_3 \rangle$ produce como resultado al siguiente diagrama:



Los símbolos (palabra) que forman el lado derecho de la producción son trazados en serie de izquierda a derecha. Las flechas indican la dirección la dirección en la que se debe hacer los movimientos para realizar una sustitución, mientras que la etiqueta w indica lo que se esta sustituyendo en vez del símbolo w. Los rectángulos que encierran a w1, w2, w3, denotan el hecho de que son símbolos no terminales. Si existen símbolos terminales, se los encierra en círculo o elipse.

Gramáticas regulares y expresiones regulares.

Existe una conexión estrecha entre el lenguaje de una gramática regular y una expresión regular.

Teorema 1. Sea S un conjunto finito y $L \in S^*$.

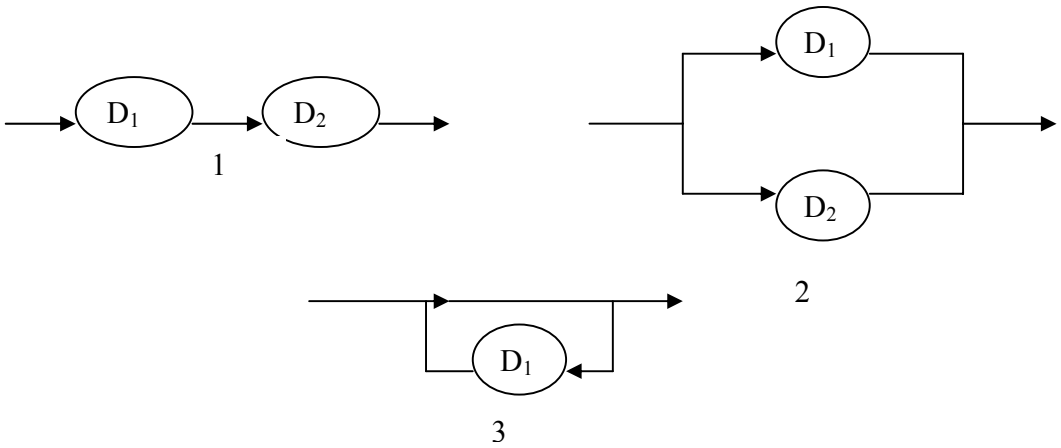
Entonces L es un conjunto regular si y solo si L = L(G) para alguna gramática regular $G = (V, S, v_{0,} \mapsto)$.

El teorema dice que el lenguaje L(G) de una gramática regular G debe ser el conjunto correspondiente a cierta expresión regular sobre S, pero no dice cómo determinar tal expresión regular.

Si la relación de G se especifica en forma BNF o en forma de diagrama de sintaxis, se puede calcular la expresión regular deseada de manera razonablemente directa.

Ejemplo:

$G = (V, S, v_{0,} \mapsto)$ y que $\mapsto$ está dada por un conjunto de diagramas de sintaxis.



Gramática formal

La gramática es un ente o modelo matemático que permite especificar un lenguaje, es decir, es el conjunto de reglas capaces de generar todas las posibilidades combinatorias de ese lenguaje, ya sea éste un lenguaje formal o un lenguaje natural.

La expresión «gramática formal» tiene dos sentidos:

(a) gramática de un *lenguaje formal*.

(b) *descripción formal* de la gramática de un lenguaje natural.

Cuando nos referimos a lenguaje natural estas reglas combinatorias reciben el nombre de sintaxis, y son inconscientes.

Hay distintos tipos de gramáticas formales que generan lenguajes formales (véase la Jerarquía de Chomsky).

Imaginemos una gramática con estas dos reglas:

1. $A \rightarrow bAc$
2. $A \rightarrow de$

La idea es sustituir el símbolo inicial de la izquierda por otros símbolos aplicando las reglas. El lenguaje al cual representa esta gramática es el conjunto de cadenas de símbolos que pueden ser generados de esta manera: en este caso, por ejemplo:

$$A \rightarrow bAc \rightarrow bbAcc \rightarrow bbbAccc \rightarrow bbbdeccc.$$

El elemento en mayúsculas es el símbolo inicial. Los elementos en minúsculas son símbolos terminales. Las cadenas de la lengua son aquellas que solo contienen elementos terminales, como por ejemplo:

bbbdeccc, de, bdec,...

Estas serían tres posibles realizaciones del lenguaje cuya gramática hemos definido con dos reglas.

Para comprender mejor el concepto pondremos algunas reglas de la gramática castellana:

- Una FRASE se puede componer de SUJETO + PREDICADO
- Un SUJETO se puede componer de un ARTICULO + NOMBRE
- Un PREDICADO se puede componer de un VERBO conjugado
- Un ARTICULO puede ser la palabra "el"
- Un NOMBRE puede ser "niño"
- etc.

Vemos que existen unas definiciones especiales como FRASE, SUJETO, etc... que no aparecen en la frase final formada. Son unas entidades abstractas denominadas Categorías Sintácticas que no son utilizables en una frase.

Las categorías sintácticas definen la estructura del lenguaje representando porciones más o menos grandes de las frases. Existe una jerarquía interna entre las categorías sintácticas.

La categoría superior sería la FRASE que representa una oración válida en lengua castellana.

Por debajo de ella se encuentran sus componentes. Ninguna de estas categorías dan lugar a frases válidas solo la categoría superior.

Al finalizar toda la jerarquía llegamos a las palabras que son las unidades mínimas con significado que puede adoptar una frase.

Aplicando las jerarquías y sustituyendo elementos, llegamos al punto en donde todas las categorías sintácticas se han convertido en palabras, obteniendo por tanto una oración VALIDA.

(Como por ejemplo: El niño corre). Este proceso se llama producción o generación.

En resumen:

Elementos constituyentes

- Una gramática formal es un modelo matemático compuesto por una serie de categorías sintácticas que se combinan entre sí por medio de unas reglas sintácticas que definen como se crea una categoría sintáctica por medio de otras y/o símbolos de la gramática.
- Existe una única categoría superior que denota cadenas completas y válidas.

Mecanismos de Especificación

- Por medio de estos elementos constituyentes se define un mecanismo de especificación consistente en repetir el mecanismo de sustitución de una categoría por sus constituyentes en función de las reglas comenzando por la categoría superior y finalizando cuando la oración ya no contiene ninguna categoría.

De esta forma, la gramática puede generar o producir cada una de las cadenas del lenguaje correspondiente y solo estas cadenas.

Categoría: Lenguajes formales

Hay 10 artículos en esta categoría.

A

- Alfabeto
- Autómata finito

C

- Cadena de caracteres
- Clausura de Kleene

E

- Expresión regular

G

- Gramática formal
- Gramática regular

J

- Jerarquía de Chomsky

L

- Lenguaje formal
- Lenguaje regular

Jerarquía de Chomsky

En 1956, Noam Chomsky clasificó las gramáticas en cuatro tipos de lenguajes y esta clasificación es conocida como la jerarquía de Chomsky, en la cual cada lenguaje es descrito por el tipo de gramática generado. Estos lenguajes sirven como base para la clasificación de lenguajes de programación. Los cuatro tipos son: lenguajes recursivamente enumerables, lenguajes sensibles al contexto, lenguajes libres de contexto y lenguajes regulares. Dichos lenguajes también se identifican como lenguajes de tipo 0, 1, 2 y 3.

Existe una exacta correspondencia entre cada uno de estos tipos de lenguajes y particulares arquitecturas de máquinas en el sentido que por cada lenguaje de tipo T hay una arquitectura de máquina A que reconoce el lenguaje de tipo T y por cada arquitectura A hay un tipo T tal que todos los lenguajes reconocidos por A son de tipo T. La correspondencia entre lenguajes y arquitectura son mostrados en la siguiente tabla.

Gramática	Lenguaje	Autómata	Normas de producción
Tipo-0	recursivamente enumerable (LRE)	Máquina de Turing (MT)	Sin restricciones
Tipo-1	dependiente del contexto(LSC)	Autómatas Linealmente Acotados	$\alpha A\beta \rightarrow \alpha\gamma\beta$
Tipo-2	independiente del contexto (LLC)	Autómata a Pila	$A \rightarrow \gamma$
Tipo-3	regular (RL)	Autómata Finito	$A \rightarrow aB$ $A \rightarrow a$

Lenguajes Recursivamente Enumerables (de tipo 0)
Son los lenguajes naturales. Las gramáticas pueden tener reglas compresoras.

Lenguajes Dependientes del Contexto (sensibles al contexto, de tipo 1)
No existen reglas compresoras, salvo, opcionalmente, la que deriva el axioma a la palabra vacía. Existen reglas en las que un símbolo no terminal puede derivar a formas sentenciales distintas, según los símbolos que aparezcan a su alrededor.

Lenguajes Independientes del Contexto (de contexto libre, de tipo 2)
La mayoría de los lenguajes de programación entran en ésta categoría.

Lenguajes Regulares (de tipo 3)
Se pueden expresar también mediante expresiones regulares.

Expresión regular

Una **expresión regular** es una forma de representar a los lenguajes regulares (finitos o infinitos) y se construye utilizando caracteres del alfabeto sobre el cual se define el lenguaje.

Más específicamente, las expresiones regulares se construyen utilizando los operadores unión, concatenación y clausura de Kleene.

Ejemplos simples

La expresión regular **a.b** (o simplemente **ab**) se lee *a concatenado con b* y representa al lenguaje L = {ab}

La expresión regular dada por **a + ab** se lee *a unión ab*, es decir, *a o ab* y representa al lenguaje L = {a, ab}

El lenguaje de todas las palabras definidas sobre el alfabeto {*a*, *b*} que comienzan con *a* se puede representar mediante la expresión regular *a*(*a* + *b*) *

Expresiones regulares en teoría de lenguajes formales

Las expresiones regulares están formadas por constantes y operadores y denotan conjuntos de palabras llamados *conjuntos regulares*. Dado un alfabeto finito Σ, se definen las siguientes constantes:

1. (*conjunto vacío*) ∅ que denota el conjunto ∅
2. (*palabra vacía*) ε que denota el conjunto {ε}
3. (*carácter del alfabeto*) *a* elemento de Σ que denota el conjunto {"*a*"}

y las siguientes operaciones:

1. (*unión*) *r*|*s* que denota la unión de *R* y *S*, donde *R* y *S* son respectivamente los conjuntos denotados por las expresiones *r* y *s*.
2. (*concatenación*) *rs* que denota el conjunto {αβ | α en *R* y β en *S*}, donde *R* y *S* representan respectivamente los conjuntos denotados por las expresiones *r* y *s*. Por ejemplo, la expresión "(ab|c)(d|ef)" denota el conjunto {"ab", "c"}{"d", "ef"} = {"abd", "abef", "cd", "cef"}.
3. (*clausura de Kleene*) *r** que denota el más pequeño conjunto que extiende a *R*, contiene ε y está cerrado por concatenación de palabras, donde *R* es el conjunto denotado por la expresión *r*. *r** es también el conjunto de todas las palabras que pueden construirse por concatenación de cero o más ocurrencias de *R*. Por ejemplo, "ab|cc*" contiene las palabras ε, "ab", "c", "abab", "abc", "cab", "cc", "abababab", etc.

Para reducir al mínimo el número de paréntesis necesarios para escribir una expresión regular, se asume que la clausura de Kleene es el operador de mayor prioridad, seguido de concatenación y luego la unión de conjuntos. Los paréntesis solo se incluyen para eliminar ambigüedades. Por ejemplo, "(ab)c" se escribe igualmente como "abc" y "U(b(c*))" puede escribirse "Ubc*".

Aplicaciones

Numerosos editores de texto y otras utilidades (especialmente en el sistema operativo UNIX), como por ejemplo sed y awk, utilizan expresiones regulares para, por ejemplo, buscar palabras en el texto y remplazarlas con alguna otra cadena de caracteres.

Autómata

Un **autómata** es una máquina auto-operada y en ocasiones la palabra es utilizada para describir a un robot.

En informática, (Teoría de los lenguajes formales) se describen tres tipos de autómatas que reconocen tipos diferentes de lenguajes: autómatas finitos, autómatas a pila y máquinas de Turing.

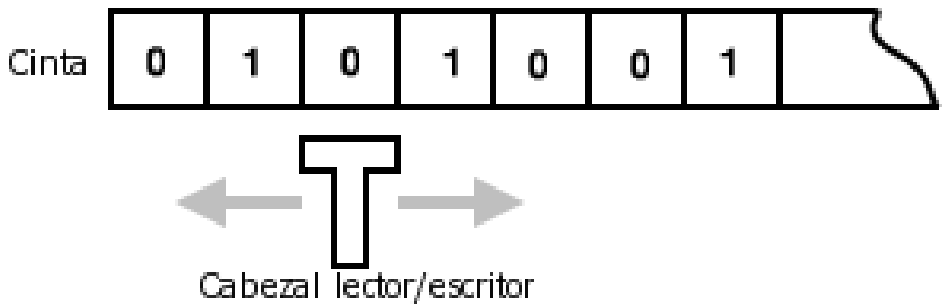
Los autómatas programables aparecieron en los Estados Unidos de América en los años 1969-70 y más particularmente en la industria del automóvil; fueron empleados en Europa dos años más tarde. Su fecha de creación coincide pues, con el comienzo de la era del microprocesador y con la generación de la lógica cableada modular. El autómata es la primera máquina con lenguaje es decir, un calculador lógico cuyo juego de instrucciones se orienta hacia los sistemas de evolución secuencial. Podríamos definirlo como un aparato electrónico, programado por un usuario, y destinado a gobernar, dentro de un entorno industrial, máquinas o procesos lógicos secuenciales.

Máquina de Turing

La **máquina de Turing** es un modelo computacional creado por Alan Turing con el cual él afirmaba que se podía realizar cualquier cómputo.
La máquina de Turing, como modelo matemático, consta de un cabezal lector/escritor y una cinta infinita en la que el cabezal lee el contenido, borra el contenido anterior y escribe un nuevo valor.

Las operaciones que se pueden realizar en esta máquina se limitan a:

- avanzar el cabezal lector/escritor para la derecha;
- avanzar el cabezal lector/escritor para la izquierda.



El cómputo es determinado a partir de una tabla de estados de la forma:

(estado,valor) $\longrightarrow$ (\nuevo estado, \nuevo valor, dirección)

Esta tabla toma como parámetros el estado actual de la máquina y el carácter leído de la cinta, dando la dirección para mover el cabezal, el nuevo estado de la máquina y el valor a ser escrito en la cinta.

Con este aparato extremadamente sencillo es posible realizar cualquier cómputo que un computador digital sea capaz de realizar.

Mediante este modelo teórico y el análisis de complejidad de algoritmos, fue posible la categorización de problemas computacionales de acuerdo a su comportamiento, apareciendo así, el conjunto de problemas denominados P y NP, cuyas soluciones en tiempo polinómico son encontradas según el determinismo y no determinismo respectivamente de la máquina de Turing.

De hecho, se puede probar matemáticamente que para cualquier programa de computadora es posible crear una máquina de Turing equivalente. Esta prueba resulta de la Tesis de Church - Turing, formulada por Alan Turing y Alonzo Church, de forma independiente a mediados del siglo XX.

Alfabeto

El **alfabeto** es la agrupación de símbolos con un orden determinado utilizado en el lenguaje escrito que sirve como sistema de comunicación. La palabra «alfabeto» se deriva del nombre de las dos primeras letras griegas *alfa* (α) y *beta* (β).

En matemáticas, un **alfabeto** es un conjunto finito y ordenado de símbolos.

Ejemplos de alfabetos

- Alfabeto árabe
- Alfabeto batak
- Alfabeto cirílico
- Alfabeto georgiano (Mxedruli)
- Alfabeto gótico
- Alfabeto griego
- Alfabeto hebreo
- Alfabeto romano (o latino)
- Alfabeto español

Alfabetos especiales

- Alfabeto Morse
- Alfabeto por palabras

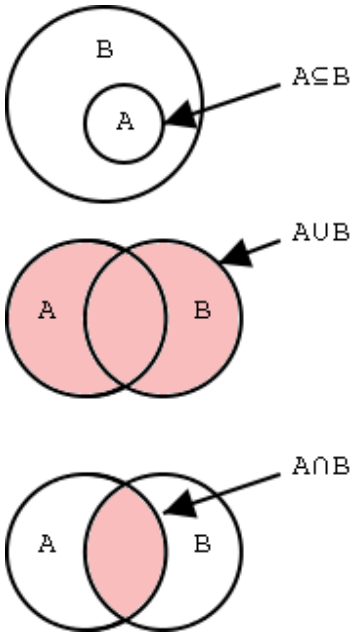
Conjunto

En matemáticas, un **conjunto** es una colección de objetos, tales que dos conjuntos son iguales si, y sólo si, contienen los mismos objetos. Se puede obtener una descripción mas detallada en la Teoría de conjuntos.

Los conjuntos son uno de los conceptos básicos de las matemáticas. Como ya se ha dicho, un conjunto es, más o menos, una colección de objetos, denominados **elementos**. La notación estándar utiliza llaves ('{' y '}') alrededor de la lista de elementos para indicar el contenido del conjunto, como por ejemplo:

{rojo, amarillo, azul} ; {rojo, azul, amarillo, rojo} ; {x: x es un color primario}

Las tres líneas anteriores denotan el mismo conjunto. Como puede verse, es posible describir el mismo conjunto de diferentes maneras: Bien dando un listado de sus elementos (lo mejor para conjuntos finitos pequeños) o bien dando una propiedad que defina todos sus elementos. Por otro lado, no importa el orden, ni cuantas veces aparezcan en la lista sus elementos.



Si *A* y *B* son dos conjuntos y todo elemento *x* de *A* está contenido también en *B*, entonces se dice que *A* es un subconjunto de *B*. Todo conjunto tiene como subconjunto a sí mismo y al conjunto vacío, {}.

La **unión** de una colección de conjuntos $S = \{S_1, S_2, S_3,...\}$ es el conjunto de todos los elementos contenidos en, al menos, uno de los conjuntos $S_1, S_2, S_3,....$

La **intersección** de una colección de conjuntos $T = \{T_1, T_2, T_3,...\}$ es el conjunto de todos los elementos contenidos en todos los conjuntos.
La *unión* y la *intersección* de conjuntos, $A_1, A_2, A_3,...$ se representa como $A_1 \cup A_2 \cup A_3 \cup ...$ y $A_1 \cap A_2 \cap A_3 \cap ...$ respectivamente.

Algunos ejemplos de conjuntos de número son:

1. Los números naturales utilizados para contar los elementos de un conjunto.
2. Los números enteros
3. Los números racionales
4. Los números reales, que incluyen a los números irracionales
5. Los números complejos que proporcionan soluciones a ecuaciones del tipo $x^2 + 1 = 0$.

La teoría estadística se construye sobre la base de la teoría de conjuntos y la teoría de la probabilidad.

Cadena de caracteres

En matemáticas, una **cadena de caracteres**, **palabra** o **frase** es una secuencia ordenada de longitud arbitraria (aunque finita) de elementos que pertenecen a un cierto alfabeto.

Habitualmente se usan las letras w, x, y,... para referirnos a cadenas. Por ejemplo, si tenemos un alfabeto $\Sigma = \{a, b, c\}$, una cadena podría ser: x = aacbbcba.

En general, una **cadena de caracteres** es una sucesión de caracteres (letras, números y/o determinados signos).

Se utilizan en programación, normalmente como un tipo de dato predefinido, para palabras, frases o cualquier otra sucesión de caracteres.

Es muy habitual que se delimiten mediante comillas superiores ("). Para poder mostrar, por ejemplo, una comilla (") dentro de la cadena y no tener problemas con las comillas que la delimitan, se usan secuencias de escape. Esto se aplica a otros caracteres reservados o no imprimibles como el retorno de carro. No obstante, las expresiones para producir estas secuencias de escape dependen del lenguaje de programación que se esté usando.

Una forma común, en muchos lenguajes, de escapar un carácter es anteponiéndole un «\» (sin comillas), p. e.: «\"» (sin comillas).

Algunas operaciones comunes
Concatenación: unir dos cadenas de caracteres.

```
$pareja = "Luis"." y "."Carmen"     # en Perl y PHP;
pareja = "Luis" & " y " & "Carmen"  # en Visual Basic;
pareja = "Luis" + " y " + "Carmen"; # en C++ con el tipo string.
```

Multiplicar una cadena: repetir una cadena un número de veces

```
$puntos ="." x 5 # pone 5 puntos en Perl
```

Máquinas de estado finito

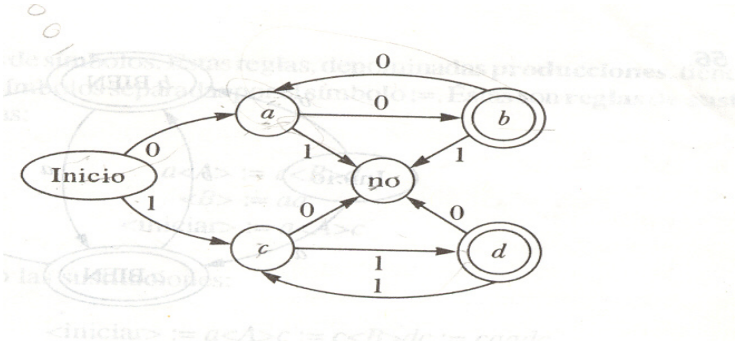
Autómatas

Cuando se traza un círculo adicional alrededor del vértice "continuar", debido a que este estado indica que la máquina ha aceptado el dinero y permitirá continuar. Cuando se desea indicar que una máquina ha logrado un objetivo cuando llega a cierto estado, a este estado se le denomina estado de aceptación y se traza un círculo alrededor de éste en la digráfica de la máquina.

Un autómata es una máquina con un conjunto selecto de estados denominados estados de aceptación y un solo estado denominado estado inicial.

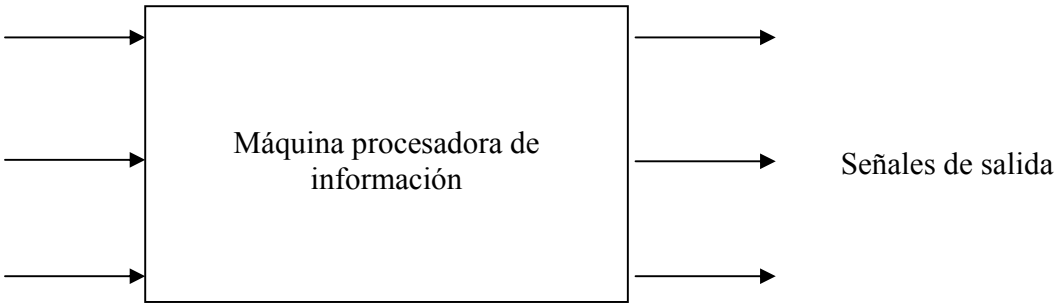
Una máquina de estado finito o autómata finito es aquella que tiene un número finito de estados. Así como se puede introducir una serie de monedas a una máquina recaudadora, también se puede dar a cualquier autómata una serie de caracteres de su alfabeto de entrada. Una serie finita de caracteres se llama **cadena de caracteres**. Se dice que el autómata *acepta* una cadena de caracteres si, cuando empieza en su estado inicial y recibe esta serie de entradas, la maquina llega a un estado de aceptación.

EJEMPLO: Describa el conjunto de estados, el alfabeto de entrada y el conjunto de cadenas de caracteres aceptados por el autómata determinista de la figura 55.



Solución: El conjunto de estados es {empezar, *no, a, b, c, d}.* Los estados de aceptación son *b* y *d.* El alfabeto de entrada es {O, 1}, ya que los bordes que salen de cada vértice tienen la etiqueta O o la etiqueta 1. Una serie que conduce al estado *b* solamente puede tener ceros. Se puede demostrar por inducción sobre el número de ceros que una serie de ceros conduce a un estado *a* si tiene un número impar de ceros y a un estado *b* si tiene un número par de ceros. Por tanto, una cadena de caracteres que consiste en ceros es aceptada si y sólo si tiene un número par de ceros. De manera semejante, una cadena que consiste en unos es aceptada si y sólo si tiene un número par de unos. Así, una cadena es aceptada si y sólo si consiste en un número par de ceros o en un número par de unos.

Una *máquina procesadora de información* es un dispositivo que recibe un conjunto de señales de entrada, y produce en correspondencia un conjunto de señales de salida. Como se ilustra en la siguiente figura.



Por tanto, podemos considerar que una lámpara de mesa es una máquina procesadora de información: la señal de entrada es la posición *ENCENDIDO* y la posición *APAGADO* del interruptor, y la señal de salida es la *ILUMINACIÓN* o la *OSCURIDAD* Otro ejemplo de máquina

procesadora de información es un sumador, cuyas señales de entrada son dos números decimales y la señal de salida su suma. En un automóvil. Que es una máquina procesadora de información, las señales de entrada son la presión sobre el acelerador y la posición angular del volante, y las señales de salida corresponden a la velocidad y dirección del vehículo. En una máquina vendedora, también una procesadora de información las señales de entrada son las monedas depositadas y la selección de la mercancía, las señales de salida son la mercancía y, posiblemente, el cambio. Por ultimo una computadora digital es una máquina procesadora de información; el programa del usuario y los datos son las señales de entrada, y los resultados impresos del cálculo son las señales de salida.

En general, las señales de entrada para una máquina procesadora de información cambian con el tiempo. En ese caso, las señales de salida también variarán con el tiempo en la forma correspondiente. Así, una máquina procesadora de información recibe una serie (temporal) de señales de entrada y produce una correspondiente serie (temporal) de señales de salida.

Consideremos el ejemplo de una lámpara de mesa: la señal de entrada es una las dos posibles posiciones del interruptor, *ENCENDIDO* o *APAGADO,* y la señal de salida es.

Una de las dos posibles condiciones, *ILUMINACIÓN* u *OSCURIDAD.* Por tanto, en correspondencia con la serie de señales de entrada

ENCENDIDO	*APAGADO*	*APAGADO*	*ENCENDIDO*
APAGADO	*ENCENDIDO*	*ENCENDIDO*	

existe la serie de señales de salida

ILUMINACIÓN	*OSCURIDAD*	*OSCURIDAD*	*ILUMINACIÓN*
OSCURIDAD	*ILUMINACIÓN*	*ILUMINACION*	

En el ejemplo del sumador, donde las señales de entrada son dos números de un digito y la señal de salida es un número de dos dígitos, a las series de señales de entrada son

3	5	0	3	3	9	2….
4	4	6	1	4	5	5….

Y la correspondiente de señales de salida es

7	9	6	4	7	14	7….

Consideremos el ejemplo de la máquina vendedora, donde las señales de entrada son monedas de cinco, diez o veinticinco centavos, y la señal de salida es un paquete de goma de mascar o nada. Suponga que un paquete de goma de mascar cuesta 30 centavos.

Entonces, como corresponde a la serie de señales de entrada

Diez	*Diez*	*Diez*	*Veinticinco*	*Veinticinco*	*Cinco*	*Veinticinco*	*Cinco…*

existe la serie de señales de salida

nada	nada	goma de mascar	nada	goma de mascar	nada	goma de mascar	nada…

Advirtamos que existe una diferencia significativa entre las máquinas de este ejemplo. En el caso de la lámpara de mesa, siempre que la señal de entrada es *ENCENDIDO,* la señal de salida es *ILUMINACIÓN,* y siempre que la señal de entrada es *APAGADO,* la señal de salida es *OSCURIDAD.* Es decir, la señal de salida en depende en cualquier momento sólo de u señal de entrada activada en tal instante, y no de las señales de entrada anteriores a dicho instante. En el caso del sumador sucede lo mismo, la señal de salida en cualquier momento siempre es la suma de los dos números dados como entrada en tal instante, y es independiente por completo de los números que fueron sumados con anterioridad.
Por otro lado, en el caso de la máquina vendedora, la señal de salida obtenida en cualquier instante depende no solo de la señal de entrada dada en tal instante, sino, además, de las señales de entrada precedente.

Así, para las tres señales de entrada sucesivas

DIEZ *DIEZ* *DIEZ*

Las correspondientes señales de salida son

Nada *Nada* *Goma de mascar*

En forma más específica, en el primer instante la entrada es *DIEZ* y la correspondiente salida es *NADA;* en el segundo instante la entrada es *DIEZ* y la salida correspondiente es NADA pero en el tercer instante, la entrada es *DIEZ y* la salida correspondiente es *GOMA DEMAR.*
Es por esto que dividimos a las máquinas en dos clases -unas con memoria y otras sin memoria. Para que una máquina sin memoria su salida en cualquier instante, sólo depende de la entrada en tal instante. Tanto la lámpara de mesa como el sumador analizados, son ejemplos de máquinas que no tienen memoria. Para una máquina con memoria, su salida en cualquier instante depende de la entrada en dicho instante como de las entradas en instantes previos debido a que la máquina puede recordar "qué ha sucedido en el pasado". Es claro que uno máquina vendedora puede recordar qué ha sucedido en el pasado. No obstante, ésta no recuerda (o no puede recordar) *todo* lo que ha sucedido en el pasado. En un instante la máquina recuerda la cantidad total que ha sido depositada hasta entonces. Es decir, mientras la cantidad total- sea, digamos, 25 centavos, la máquina no hace diferencia alguna acerca que si se depositaron cinco monedas de cinco centavos, dos monedas de diez centavos y una de cinco centavos, una moneda de veinticinco centavos, u otras combinaciones posibles. Para describir los pasados eventos, introducimos el concepto de *estado.* **Un estado representa resumen de la historia pasada de una máquina.**

En el ejemplo de la máquina vendedora existen siete estados diferentes, que corresponden al depósito total acumulado a saber, 0, 5, 10, 15, 20, 25 Y 30 o más centavos. En consecuencia, el estado de la máquina junto con las señales de entrada en un instante particular, determinarán las

señales de salida correspondientes a dicho instante. Siguiendo con el ejemplo de esta máquina, en un instante cualquiera, el estado en que se encuentra en la máquina más el nuevo depósito, permitirán a la máquina determinar si deberá tener *NADA* o *GOMA DE MASCAR* como salida. Además conforme llega otra señal de entrada, la máquina pasará desde un estado a otro, puesto que necesita actualizar el resumen de su historia.

Deposito Total	Deposito nuevo		
	5c	**10c**	**25c**
0c	5c	10c	25c
5c	10c	15c	30c o mas
10c	15c	20c	30c o mas
15c	20c	25c	30c o mas
20c	25c	30c o mas	30c o mas
25c	30c o mas	30c o mas	30c o mas
30c o más	5c	10c	25c

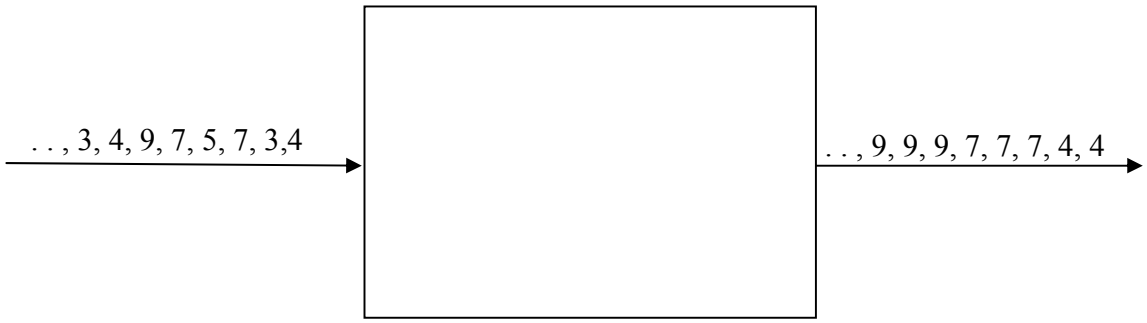
Si iniciamos con una entrada de 5 centavos llevaría a la máquina al estado de 5 centavos, una entrada de 10 centavos llevaría a la máquina al estado de 15 centavos, una entrada de 10 centavos llevaría a la máquina al estado de 25 centavos o una entrada de 25 centavos la llevaría al estado de 30 centavos o más, etcétera.

El comportamiento de la máquina vendedora puede resumirse como se muestra en la tabla siguiente.

Depósito total	Mercancía entregada
0c	Nada
5c	Nada
10c	Nada
15c	Nada
20c	Nada
25c	Nada
30 o mas o más	Goma de mascar

Como otro ejemplo, consideremos una máquina que acepta una sucesión de enteros positivos entre 1 y 100, Y produce como salida en cualquier instante el entero más grande que ha recibido hasta ese momento, como se ilustra en la siguiente figura. Observemos que en tanto la máquina pueda recordar el mayor entero que ha recibido, cuando llega una nueva entrada, la máquina puede comparar el mayor entero recibido hasta el momento con la nueva entrada, y determinar la

salida correspondiente, la cual es el "nuevo" entero más grande recibido hasta el momento. Así, para esta máquina, un resumen de la historia pasada puede ser representado mediante un entero igual al mayor entero que haya recibido. Por tanto, la máquina deberá tener 101 estados correspondientes a los enteros 0, 1,. . ., 100, que representan el mayor entero que la máquina ha recibido (es claro que el estado 0 significa que ningún entero ha sido recibido)



.

MÁQUINAS DE ESTADO FINITO

Ahora introducimos un modelo abstracto de una máquina de estado finito, la cual está especificada por:

1. Un conjunto finito de estados S= *{s_0,* s_1, *s_2, . . .}*.
2. Un elemento especial del conjunto S, *s_0,* es conocido como *estado inicial.*
3. Un conjunto finito de caracteres de entrada I = {i_1, *i_2...}*
4. Un conjunto finito de caracteres de salida O = *{o_1, 0_2. . .}*.
5. Una función f de S x / a S, conocida como *función de transición.*
6. Una función g de S a O, conocida como *función de salida.*

En un instante cualquiera, una máquina de estado finito se encuentra en uno de sus estados. Al llegar un carácter de entrada, la máquina pasará a otro estado de acuerdo con la función de transición. Además, en cada estado la máquina produce un carácter de salida de acuerdo con la función de salida. Al principio, la máquina se encuentra en su estado inicial. Una forma conveniente para describir una máquina de estado finito es la forma tabular usada en la figura siguiente. Para la máquina de estado finito de la tabla (a), el conjunto de estados es *{s_0, s_1, s_2, s_3, s_4, s_5, s_6},* el conjunto de caracteres de entrada es *{a, b, c},* y el conjunto de caracteres de salida es {0, 1}. La doble flecha que apunta hacia S_0 indica que S_0 es el estado inicial. La función de transición f está especificada en la figura (a), donde el estado en 1a intersección de una fila (correspondiente al estado *sp)* y una columna correspondiente al carácter de entrada *iq),* es el valor *f(sp, iq)*

Estado	Entrada		
	a	***b***	***c***
=> s_0	s_1	s_2	s_5
s_1	s_2	s_3	s_6
s_2	s_3	s_4	s_6
s_3	s_4	s_5	s_6
s_4	s_5	s_6	s_6
s_5	s_6	s_6	s_6
s_6	s_1	s_2	s_5

(a)

Estado	Salida
s_0	0
s_1	0
s_2	0
s_3	0
s_4	0
s_5	0
s_6	1

(b)

La función de salida se especifica en la tabla (b). Por lo regular, las tablas (a) y (b) pueden combinarse en una sola tabla, como en la figura siguiente.

Estado	Entrada			Salida
	a	***b***	***c***	
s_0	s_1	s_2	s_5	0
s_1	s_2	s_3	s_6	0
s_2	s_3	s_4	s_6	0
s_3	s_4	s_5	s_6	0
s_4	s_5	s_6	s_6	0
s_5	s_6	s_6	s_6	0
s_6	s_1	s_2	s_5	1

De hecho, la máquina de estado finito de la figura anterior es precisamente la máquina vendedora mostrada de la figura, 7.6. En específico, los estados *so, s1, s2, s3, s4, s5* y *s6,* corresponden a los estados 0,5, 10, 15, 20,25 Y 30 o más centavos. Los caracteres de entrada *a, b* y c corresponden a las monedas de *CINCO, DIEZ* Y *VEINTICINCO centavos* caracteres de salida 0 y 1 corresponden al momento en que la máquina entregará; NADA O *GOMA DE MASCAR.*

Estado	Entrada		
	a	***c***	***b***
=> s_0	s_1	s_2	s_5
s_1	s_2	s_3	s_6
s_2	s_3	s_4	s_6
s_3	s_4	s_5	s_6
s_4	s_5	s_6	s_6
s_5	s_6	s_6	s_6
s_6	s_1	s_2	s_5

Estado	Salida
s_0	0
s_1	0
s_2	0
s_3	0
s_4	0
s_5	0
s_6	1

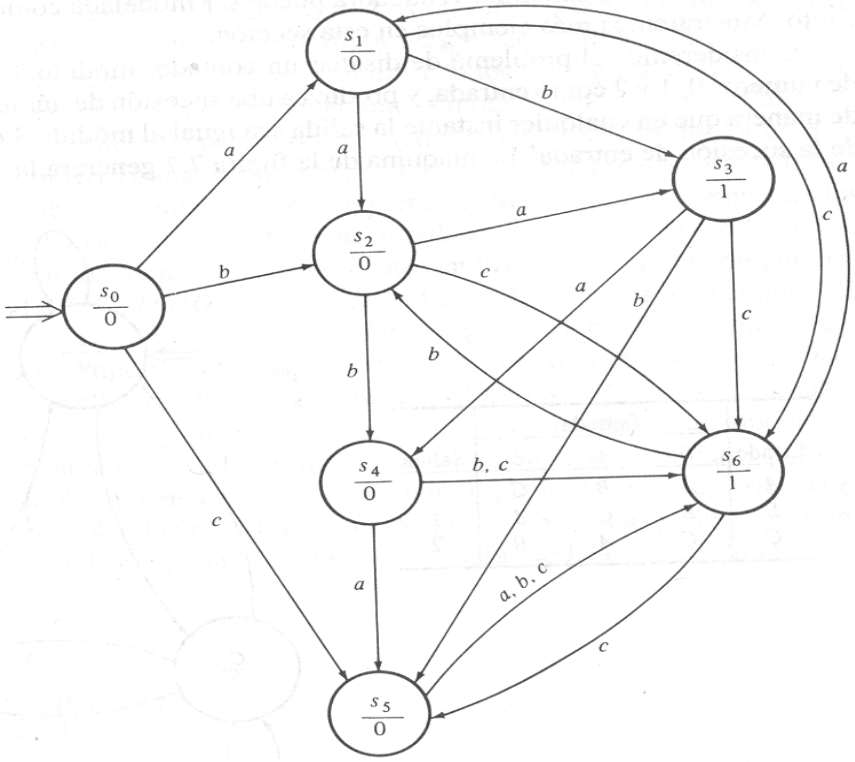


Figura 7.6

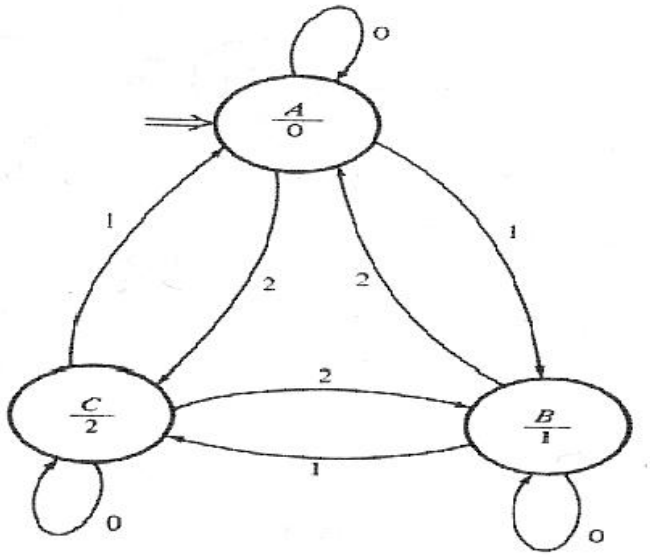
Podemos describir una máquina de estado finito en forma gráfica, como la figura 7.6. En el grafo dirigido de la figura 7.6, cada vértice corresponde a un estado de la máquina. De nuevo, el estado inicial se identifica por una doble flecha que apunta hacia él. La salida asociada con un estado se coloca bajo el nombre del estado, separado por una barra horizontal. La transición desde un estado a otro se indica mediante una arista dirigida y etiquetada con el(los) correspondiente(s) carácter(es) de entrada. Efectivamente las tablas anteriores describen la misma máquina de estado finito, como puede verificar rápidamente el lector.

MÁQUINAS DE ESTADO FINITO COMO MODELOS DE SISTEMAS FÍSICOS

Una máquina de estado finito puede ser útil para modelar un sistema físico. Anteriormente vimos cómo una máquina vendedora puede ser modelada como una máquina finita. Mostraremos más ejemplos.

Consideremos el problema de diseñar un contador módulo 3 que reciba de números 0, 1 Y 2 como entrada, y produzca una sucesión de números 0, 1 Y 2 como salida de manera que en cualquier instante la salida sea igual al módulo 3 de la suma de los dígitos de la sucesión de entrada. La máquina de la figura siguiente generará la sucesión de salida como:

Estado	Entrada			Salida
	O	1	2	
=>*A*	*A*	*B*	C	O
B	*B*	C	*A*	1
C	C	*A*	*B*	2



Se especifica. Observe que *A* es un estado correspondiente a la situación en que el módulo 3 de la suma de todos los dígitos de entrada es 0; *B* es un estado que corresponde a la situación en que el módulo 3 de la suma de todos los dígitos de entrada es 1, Y C es un estado que corresponde a la situación en que el módulo 3 de la suma de todos los dígitos de entrada es 2.

Ahora analicemos otro ejemplo, en el cual se dispositivo que compara dos números binarios para determinar si éstos son iguales, o cuál de los dos es mayor. Suponemos que los dígitos de los dos números llegan uno por uno, empezándose con los dígitos de orden inferior. Así, el alfabeto de entrada es {00, 01, 10, 11}, donde los dos dígitos de cada pareja son los dígitos correspondientes en los números a ser comparados. El alfabeto de salida es *{IGUAL, MAYOR, MENOR}.* La figura a continuación muestra la máquina.

Estado	**Entrada**				
	00	01	10	11	**Salida**
=>*A*	*A*	C	*B*	*A*	IGUAL
B	*B*	C	*B*	*B*	MAYOR
C	C	C	*B*	C	MENOR

Por último, la figura siguiente muestra un ejemplo de una máquina de estado finito que modela el comportamiento de un estudiante. El conjunto de estados es *{FELIZ, ENOJADO, DEPRIMIDO},* el conjunto de entradas es *{TAREA, FIESTA, MAL EXAMEN},* Y el conjunto de salidas es *{CANTAR, MALDECIR, DORMIR}.* Aunque éste es un modelo muy simplificado, de hecho, abarca un cierto aspecto de cómo reacciona un estudiante bajo diversas condiciones (debemos señalar que éste no es un ejemplo vano incluido aquí para impresionar al lector. En realidad, los psicólogos, sociólogos, economistas y científicos de diversas disciplinas hacen uso de máquinas de estado finito para modelar los sistemas que ellos estudian).

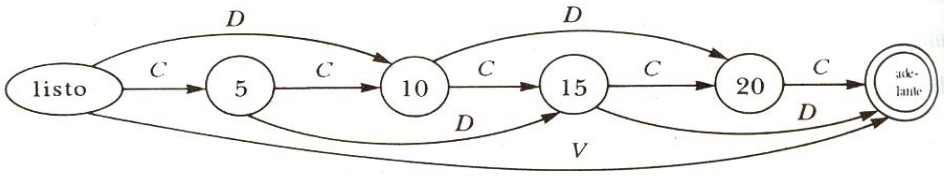
Estado	Entrada			Salida
	Tarea	Fiesta	Mal examen	
=> *A* (feliz)	*A*	*A*	*B*	cantar
B (enojado)	C	*A*	*B*	maldecir
C (deprimido)	C	*A*	C	dormir

Digráficas y máquinas

Una aplicación importante de las digráficas es obtener un modelo del funcionamiento interno de una máquina "inteligente" como una computadora. Una máquina así es el dispositivo utilizado en el carril de "cambio exacto" en la caseta de derecho de tránsito de una carretera o un puente. Esta máquina suma los valores de las monedas depositadas hasta que el valor total es la cuota requerida y luego indica al conductor que puede continuar su camino. En la figura 53 se muestra una máquina que colecta exactamente 25 centavos.

Digráfica de una máquina colectora de "cambio exacto".

Figura 53 Digráfica de una máquina colectora de "cambio exacto".



Manera en *que las digráficas pueden representar máquinas*

Cuando para representar una máquina se usa una digráfica, los vértices de ésta se llaman estados de la máquina. Así, la máquina representada en la figura 53 tiene un estado de alerta, uno de cinco centavos, uno de diez centavos, uno de quince centavos, uno de van: te centavos y un estado de avanzar. Cuando un conductor llega a la máquina, ésta se encuentra en su estado de alerta (también se dice que está "lista" o "preparada") esperando que alguien introduzca moneda". Las entradas legales (las monedas que el conductor puede introducir) son de cinco, diez y veinticinco centavos. Cada vez que la máquina recibe una entrada, puede modificar su estado con base en la entrada. Por ejemplo, si se encuentra en el estado de cinco centavos y recibe una entrada de una moneda de diez centavos, cambiará al estado de quince centavos. Cada borde de la digráfica representa este cambio de estado; la etiqueta sobre el borde indica la entrada que conduce a ese cambio de estados.

En vez de intentar definir de manera precisa el término *máquina*. Se definirá el término *digráfica de máquina*.

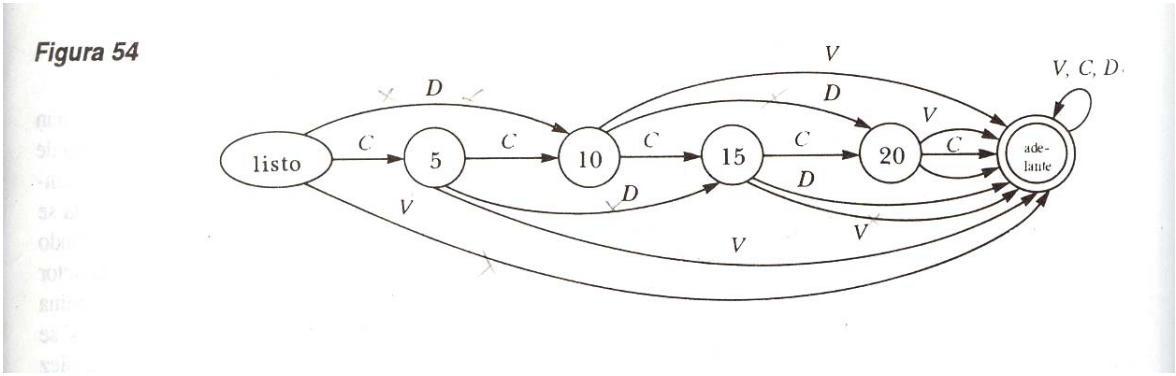
Una digráfica de máquina con conjunto de estados **S** y alfabeto de entrada ***I*** se define como una digráfica cuyo conjunto de vértices es **S** y cuyos bordes están identificados con elementos elegidos de ***I***.

(Es válido que un borde tenga más de una identificación.) En la figura 53 se observa una digráfica de máquina con conjunto de estados {alerta, 5, 10,15, 20, avanzar} y alfabeto de entrada *{C, D, V}*.
La digráfica no representa perfectamente una máquina recaudadora. Puede ser posible que una persona deposite una moneda de veinticinco centavos en la máquina inclusive si ésta se encuentra en el estado 5, 10, 15 ó 20. Como la acción de la máquina con una entrada así no está determinada se dice que este es un ejemplo de maquina ***no determinista***.(Más tarde se proporcionará una definición más formal.) El siguiente: ejemplo hace más realista el modelo de la máquina, aunque también más complicado.

EJEMPLO 29 Trace una digráfica de máquina para una máquina recaudadora que permita avanzar al conductor tan pronto como deposite 25 centavos () *Tnlís.*

Solución: Es necesario agregar flechas a la figura 53, para indicar que la máquina pasa al estado "avanzar" si en el estado 20 se introduce una moneda de 10 centavos o en los estados 5, 10, 15 ó 20 se introduce una moneda de 25 centavos. Además, si en el estado "avanzar" se deposita más dinero, el estado de la máquina ya no cambia. Esto se muestra en la figura 54. .

Figura 54



Observe en la figura 54 que de cada vértice salen tres bordes, con uno y sólo un borde para cada entrada posible. Se dice que una digráfica de máquina es una digráfica de una máquina determinista si para cada vértice hay un correspondencia uno a uno entre las etiquetas en los bordes que salen del vértice y el alfabeto de entrada. Así, la máquina de la figura 54 es determinista.

Maquinas de turing

Las maquinas de turing son mas generales que cualquier autómata finito y cualquier autómata de pila, debido a que ellas pueden reconocer tanto como los lenguajes regulares como los lenguajes independientes del contexto y, además, muchos otros tipos de lenguajes.

Una máquina de turing es una 7-tupla M= (Q, Σ, Γ, s, Ъ, F, δ), donde

Q es un conjunto de estados
Σ es un alfabeto de entrada
Γ es un alfabeto llamado alfabeto de la cinta
S ε Q es el estado inicial
Ъ ε Γ es el símbolo blanco (y no esta en Σ)
F C Q es el conjunto de estados finales o de aceptación
δ : Q x Γ → Q x Γ x {L, R} es una función parcial que se llama función de transición

En esta definición se supone que el valor inicial de todas las celdas de la cinta es el símbolo Ъ. La definición requiere que Ъ no elemento de Σ. Generalmente, permitimos que Σ C Γ – {Ъ}.La función de transición δ transforma pares (q, σ) formados por el estado actual y los símbolos de la cinta en ternas de la forma (p, t, X), donde p es el estado siguiente, t es el símbolo escrito en la cinta y X es un movimiento de lectura/escritura de la cabeza, que puede ser L o R, según que el movimiento sea hacia la izquierda o hacia la derecha (nos imaginamos que la cinta se extiende de izquierda a derecha).

Consideremos la máquina de Turíng definida mediante:

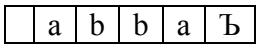
$Q = \{q_1, q_2\}$
$\sum = \{a, b\}$
$\Gamma = \{a, b, Ъ\}$
$F = \{q_2\}$

Y δ dado por

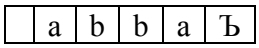
$\delta (q_1,a) = (q_1,a, R)$
$\delta (q_1,b) = (q_1,a, R)$
$\delta (q_1,b) = (q_2,b, L)$

Esta máquina empieza sus operaciones en el estado q_1. Si el contenido de la celda de la cinta sobre la que se encuentra la cabeza de lectura/escritura es a, la transición que se puede aplicar es $\delta (q_1, a) = (q_1, a, R)$.

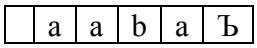
La máquina de Turing sobrescribirá la a que está en la cinta con otra a.



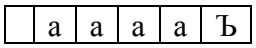
Estado q_1



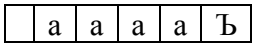
Estado q_1



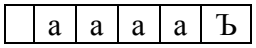
Estado q_1



Estado q_1



Estado q_1



Estado q_2

Sea cual sea la notación que utilicemos, denotaremos el paso de una configuración otra por medio del símbolo ya familiar ├. Por tanto, en el ejemplo anterior se tiene qué

$(q_1,abba)$ ├ $(q_1,abba)$ ├ $(q_1,aaba)$ ├ $(q1,aaaa)$
├ $(q_1,aaaa\ Ъ)$ ├ $(q_2,aaaa)$

$q_1,abba$ ├ $aq_1bba)$ ├ aaq_1ba ├ $aaaq_1a$ ├ $aaaq_1Ъ$
├ $aaaq_2a$

Veamos otro ejemplo en el que consideremos la siguiente maquina de turing.

$Q= \{q_1, q_2, q_3\}$
$\Sigma= \{a, b\}$
$\Gamma= \{a, b, Ъ\}$
$F= \{q_3\}$
$S= q_1$

$\delta\ (q_1,a)= (q_1,q,L)$
$\delta\ (q_1,b)= (q_1,b,L)$
$\delta\ (q_1,ъ)=(q_2,ъ,R)$
$\delta\ (q_2,a)=(a_3,a,L)$
$\delta\ (q_2,b)=(q_3,b,L)$
$\delta\ (q_2,ъ)=(q_3,ъ,L)$

Esta máquina de Turing examinará la cinta hacia la izquierda que se encuentre con la primera celda en blanco. Entonces parará y se colocará sobre el blanco. Por tanto, deberíamos tener:

$(q_1, aababb) \vdash (q_1,aababbb) \vdash (q_1,aababb)$
$\vdash (q_1,aababbb) \vdash (q_1,baababb)$
$\vdash (q_2,aababbb) \vdash (q_3,ъaababb)$

$aabq_1abb \vdash aaq_1babb \vdash aq_1ababb \vdash q_1aababb$
$\vdash q_1ъaababb \vdash q_2aababb \vdash q_3ъaababb$

Aplicaciones.

Máquina Finita Determinística.

Dada la tabla:

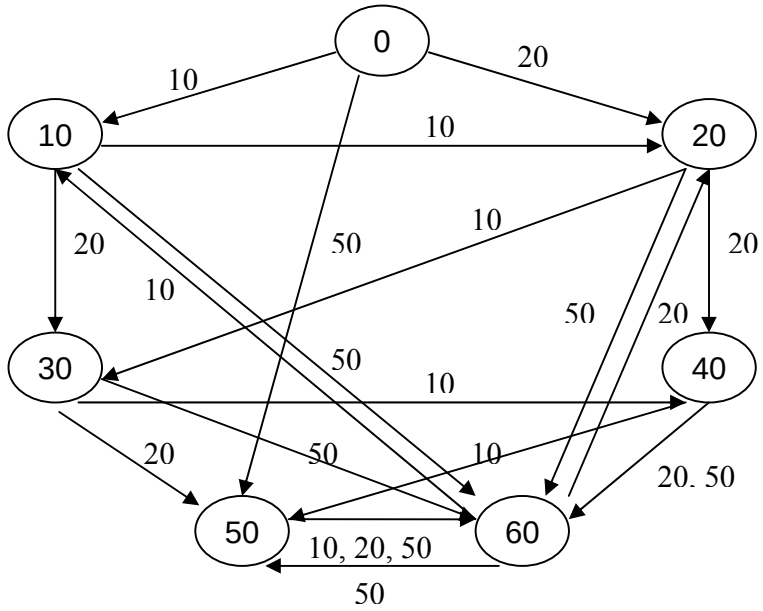
Entrada	**a**	**b**	**c**	**salida**
q_0	q_1	q_2	q_5	0
q_1	q_2	q_3	q_6	0
q_2	q_3	q_4	q_6	0
q_3	q_4	q_5	q_6	0
q_4	q_5	q_6	q_6	0
q_5	q_6	q_6	q_6	0
q_6	q_1	q_2	q_5	1

Y dado los valores:

a	10
b	20
c	50

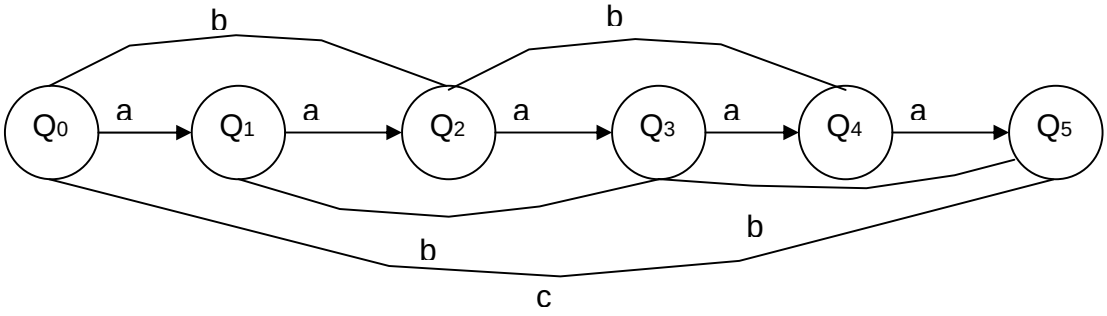
q_0	0
q_1	10
q_2	20
q_3	30
q_4	40
q_5	50
q_6	60

Represéntelo gráficamente:



Maquina Finita No Deterministica.

Dada la grafica:



Hacer su tabla BNF:

$<q_0>$	a $<q_1>$	b $<q_2>$	c $<q_5>$
$<q_1>$	a $<q_2>$	b $<q_3>$	
$<q_2>$	a $<q_3>$	b $<q_4>$	
$<q_3>$	a $<q_4>$	b $<q_5>$	
$<q_4>$	a $<q_5>$		
$<q_5>$			