Chapter 2 Automata, Grammars, and Formal Languages

2-1 Finite-state Automata and Sequential Logic Circuits

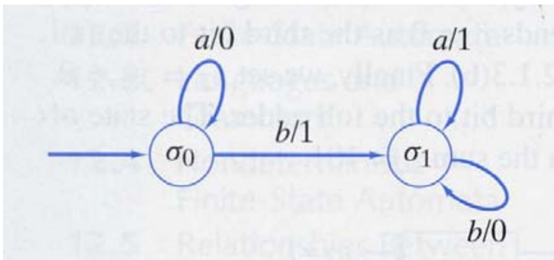
Transition diagram (or State diagram): It describes the relation of inputs/outputs and the transitions between the states.

TABLE

$S \backslash I$	f: a	f: b	g: a	g: b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

Eg. According to the left table, draw the corresponding transition diagram.

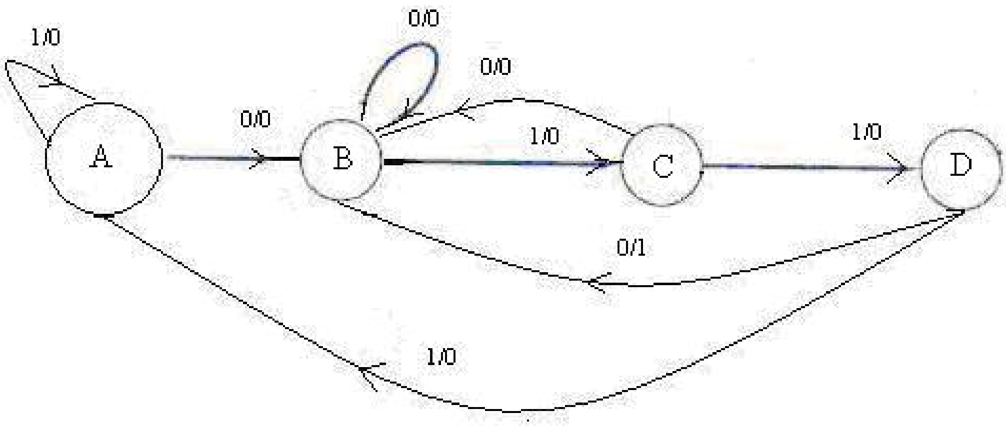
(Sol.)



Finite-state machine, *M*: It consists of a set I of input symbols, a set O of output symbols, a finite set S of states, a next-state function f, an output function g, and an initial state σ.

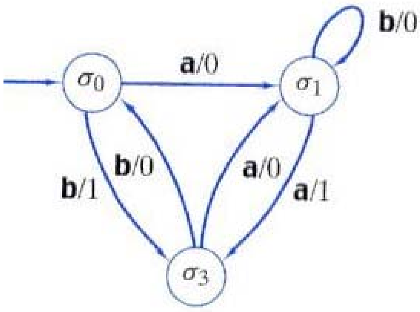
Eg. Draw the transition diagram of the finite-state machine which accepts a serial 0110 contained in a long string over [0, 1].

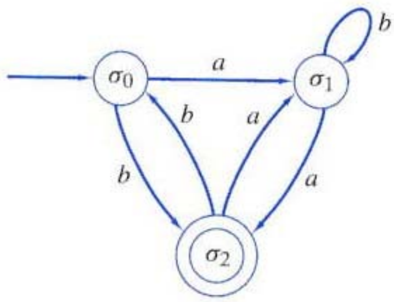
(Sol.)



Finite-state automaton, *A*: It is a finite-state machine in which the set of output function is {0,1} and the current state determine the last output. Those states foe which the last output was 1 are called **accepting states**. Let $\alpha=x_1x_2x_3\ldots x_n$ be a string, If there exist states σ_0, σ_1, σ_2, σ_3, …, σ_n satisfying (a) $\sigma_0=\sigma$, (b) $f(\sigma_{i-1},x_i)=\sigma_i$ for i=1, …, n, (c) $\sigma_n \in$ the set of the accepting states, then α is accepted by the finite automaton.

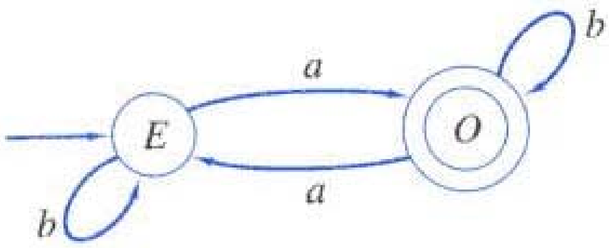
Eg. The transition diagrams of a finite-state machine and its finite-state automaton.





Eg. Draw the transition diagram of the finite-state automaton which accepts a string over [*a*,*b*] that contain an odd number of *a*'s.

(Sol.)

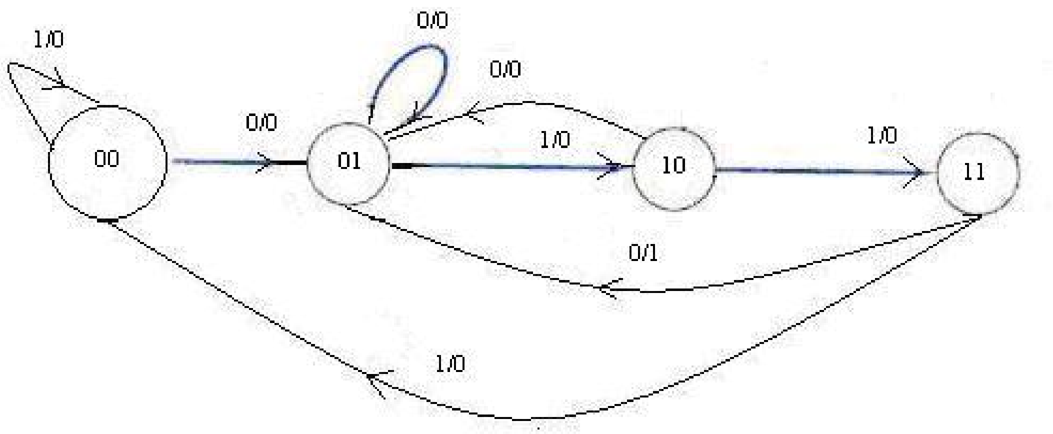


Application of finite-state machines and finite-state automata: Designing sequential logic circuits

Eg. Use *J-K* Flip-flops and other logic gates to design a digital circuit that accepts a serial 0110 contained in a long string over [0, 1].

(Sol.)

Encoding the transition diagram:



Excitation table of *J-K* Flip-flop:

$Q(t)$	$Q(t+\tau)$	J	K
0	0	0	*d*
0	1	1	*d*
1	0	*d*	1
1	1	*d*	0

According to the transition diagram and the excitation table of *J-K* flip-flop, we have the following truth table:

$A(t)$	$B(t)$	x	$A(t+\tau)$	$B(t+\tau)$	y	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	*d*	1	*d*
0	0	1	0	0	0	0	*d*	0	*d*
0	1	0	0	1	0	0	*d*	*d*	0
0	1	1	1	0	0	1	*d*	*d*	1
1	0	0	0	1	0	*d*	1	1	*d*
1	0	1	1	1	0	*d*	0	1	*d*
1	1	0	0	1	1	d	1	d	0
1	1	1	0	0	0	d	1	d	1

Let $A=Q_A$, $B=Q_B$ for the two *J-K* flip-flops, and then we can obtain the relation of J_A, K_A, Q_A, J_B, K_B, Q_B and A, B, x, and y by the Karnaugh map:

	$\overline{A}\overline{B}$	$A\overline{B}$	AB	$\overline{A}B$
x		d	d	1
$\overline{x}$		d	d	

$J_A = Bx$

	$\overline{A}\overline{B}$	$A\overline{B}$	AB	$\overline{A}B$
x	d		1	d
$\overline{x}$	d	1	1	d

$K_A = \overline{x} + B$

	$\overline{A}\overline{B}$	$A\overline{B}$	AB	$\overline{A}B$
x		1	d	d
$\overline{x}$	1	1	d	d

$J_B = \overline{x} + A$

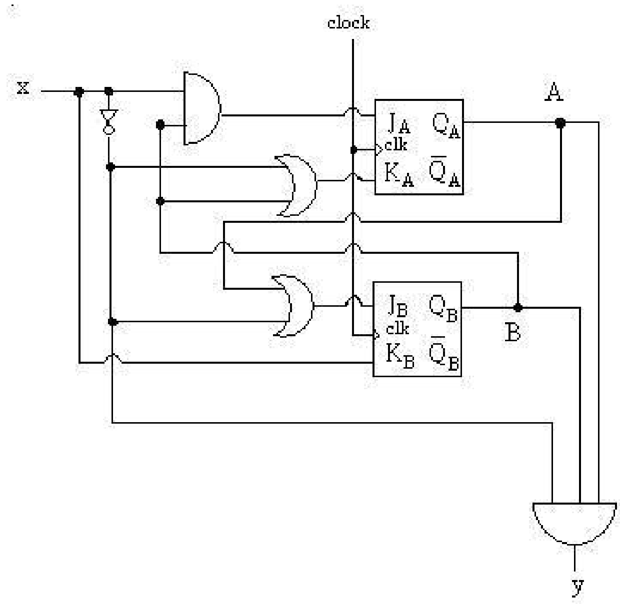
	$\overline{A}\overline{B}$	$A\overline{B}$	AB	$\overline{A}B$
x	d	d	1	1
$\overline{x}$	d	d		

$K_B = x$

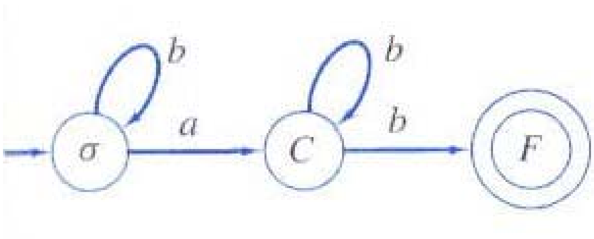
	$\overline{A}\overline{B}$	$A\overline{B}$	AB	$\overline{A}B$
x				
$\overline{x}$			1	

$y = AB\overline{x}$

The sequential logic circuit is as shown as in the following diagram:

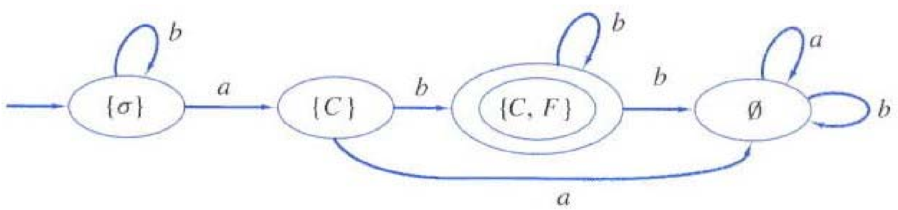


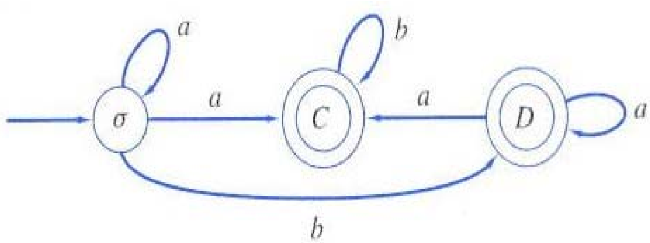
Nondeterministic finite-state automaton: It is a finite-state automaton consisting of a set I of input symbols, a finite S of states, a subset A of S of accepting states, a next-state function f, and an initial state σ.



Eg. The left finite-state automaton is a nondeterministic finite-state automaton. Vertex *C* has no outgoing edge labeled *a*, and it has 2 outgoing edges labeled *b*. In state *C*, if *b* is input, we have 2 choices of next states. It can remain in state *C* or go to state *F*. On the other hand, vertex *F* has no outgoing edges at all. In state *F*, null string is input and is accepted by the nondeterministic finite-state automaton.

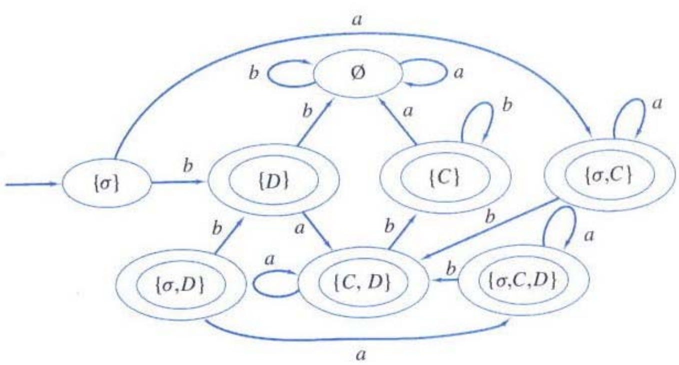
This nondeterministic finite-state automaton is equivalent to a finite-state automaton as shown in the follow figure.





Eg. Vertex *D* of the left nondeterministic finite-state automaton has 2 outgoing edges labeled *a*. In state *F*, if *a* is input, we have 2 choices of next states. It can remain in state *D* or go to state *C*.

This nondeterministic finite-state automaton is equivalent to a finite-state automaton as shown in the follow figure.



2-2 Formal Languages and Grammars

Formal language: Let A be a finite set. A formal language L over A is a subset of A^*, the set of all strings over A.

Eg. Let A={a,b,c}, then *cbabb*, *aab*, *abcab*, *bcba*, *cb*, …, are all formal languages.

Eg. Let A={狗,咬,人}, then "狗咬人" and "人咬狗" are both formal languages.

Eg. Let A={women, like, men}, then "women like men" and "men like women" are both formal languages.

Grammar G: It consists of a finite set N of non-terminal symbols, a finite set T of terminal symbols (Note: $N \cap T = \phi$), a starting symbol σ, and a finite set P of productions $[(N \cup T)^* - T^*] \times (N \cup T)^*$.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by $N=\{\sigma, S\}$, $T=\{a,b\}$, $P=\{\sigma \to b\sigma, \sigma \to aS, S \to bS, S \to b\}$. Show that *bbbabb* is in agreement with the grammar G, but *aab* is not in agreement with the grammar G.

(Proof) $\sigma \Rightarrow b\sigma \Rightarrow bb\sigma \Rightarrow bbb\sigma \Rightarrow bbbaS \Rightarrow bbbabS \Rightarrow bbbabb$, ∴ *bbbabb* is **in agreement with** the grammar G.

$\sigma \Rightarrow aS \Rightarrow abS$ or ab, ∴ ***aab*** is **not in agreement with** the grammar G.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by $N=\{\sigma, A, B\}$, T={狗,咬,人}, P={σ→狗 A, σ→人 A, A→咬 B, B→狗, B→人}. Show that "狗咬人" is in agreement with the grammar, but "狗人咬" is not in agreement with the grammar.

(Proof) σ⇒狗 A⇒狗咬 B⇒狗咬人

∴ "狗咬人" is **in agreement with** the grammar G.

σ⇒狗 A⇒狗咬 B, ∴ "狗人咬" is **not in agreement with** the grammar G.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by N={σ,women,like,men,A,B}, T={.}, P={σ→women A, σ→men A, A→like B, B→women., B→men.}. Show that "women like men." is in agreement with the grammar, but "women men like." is not in agreement with the grammar.

(Proof) σ⇒women A⇒women like B⇒women like men.

∴ "**women like men.**" is **in agreement with** the grammar G.

σ⇒women A⇒women like B, ∴"**women men like.**" is **not in agreement with** the grammar G.

Context-sensitive grammar: Its production is of the form $\alpha A\beta\rightarrow\alpha\delta\beta$, where α, $\beta\in(N\cup T)^*$, $A\in N$, $\delta\in(N\cup T)^*-\{\lambda\}$.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by $N=\{\sigma,A,B,C,D,E\}$, $T=\{a,b,c\}$, $P=\{\sigma\rightarrow aAB, \sigma\rightarrow aB, A\rightarrow aAC, A\rightarrow aC, B\rightarrow Dc, D\rightarrow b, CD\rightarrow CE, CE\rightarrow DE, DE\rightarrow DC, Cc\rightarrow Dcc\}$. $\because$ $CD\rightarrow CE\rightarrow DE\rightarrow DC$,

$\therefore$ $\sigma\Rightarrow aAB\Rightarrow aaACB\Rightarrow aaaACCB\Rightarrow aaaaACCCDc\Rightarrow aaaaaACCCCDc\Rightarrow\ldots$ $\Rightarrow a^{n-1}AC^{n-2}Dc\Rightarrow a^{n-1}aC^{n-1}Dc\Rightarrow a^nDC^{n-1}c\Rightarrow a^nDC^{n-2}Cc\Rightarrow a^nDC^{n-2}Dcc$ $\Rightarrow a^nD^2C^{n-2}cc\Rightarrow a^nD^2C^{n-3}Ccc\Rightarrow a^nD^2C^{n-3}Dccc\Rightarrow a^nD^3C^{n-3}ccc\Rightarrow a^nD^3C^{n-4}Cccc$ $\Rightarrow\ \ldots\ \Rightarrow\ a^nD^{n-1}Ccc^{n-2}\ \Rightarrow\ a^nD^{n-1}Dccc^{n-2}\ \Rightarrow\ a^nD^nc^n\ \Rightarrow\ a^nb^nc^n$, and $L(G)=\{a^nb^nc^n|n=1,2,\ldots\}$ is a context-sensitive language.

Context-free grammar: Its production is of the form $A\rightarrow\delta$, where $A\in N$, $\delta\in(N\cup T)^*$.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by $N=\{\sigma\}$, $T=\{a,b\}$, $P=\{\sigma\rightarrow a\sigma b, \sigma\rightarrow ab\}$. $\sigma\Rightarrow a\sigma b\Rightarrow aa\sigma bb\Rightarrow aaa\sigma bbb\Rightarrow\ldots\Rightarrow a^{n-1}\sigma b^{n-1}\Rightarrow a^{n-1}abb^{n-1}\Rightarrow a^nb^n$, **and $L(G)=\{a^nb^n|\ n=1,2,\ldots\}$ is a context-free language.**

Regular grammar: Its production is of the form $A\rightarrow a$ or $A\rightarrow aB$ or $A\rightarrow\lambda$, where A, $B\in N$, $a\in T$.

Eg. The grammar $G=(N, T, P, \sigma)$ is defined by $N=\{\sigma,S\}$, $T=\{a,b\}$, $P=\{\sigma\rightarrow b\sigma, \sigma\rightarrow aS, S\rightarrow bS, S\rightarrow b\}$. $\sigma\Rightarrow b\sigma\Rightarrow bb\sigma\Rightarrow bbb\sigma\Rightarrow\ldots\Rightarrow b^n\sigma\Rightarrow b^naS\Rightarrow b^nabS\Rightarrow b^nabbS$ $\Rightarrow b^nabbbS\Rightarrow\ldots\Rightarrow b^nab^m$, $L(G)=\{b^nab^m|n, m=1,2,\ldots\}$ is a regular language.

Backus-Naur form (BNF): Rewrite the forms such as $S\rightarrow T$ into $S::=T$, and $S\rightarrow T$, $S\rightarrow U$, $S\rightarrow V$, $S\rightarrow W$ into $S::=T|U|V|W$.

Eg. Determine whether or not -901 is an integer by the following (BNF) grammar generates all decimal integers.

<starting symbol>::=<integer>

<digit>::=0|1|2|3|4|5|6|7|8|9

<integer>::=<signed integer>|<unsigned integer>

<signed integer>::=+<unsigned integer>|-<unsigned integer>

<unsigned integer>::=<digit>|<digit><unsigned integer>

(Sol.)

<integer>::=<signed integer>::=-<unsigned integer>

::=-<digit><unsigned integer>::=-9<unsigned integer>

::=-9<digit><unsigned integer>::=-90<unsigned integer>::=-90<digit>::=-901

$\therefore$ -901 is an integer.

Application of Grammars: Generating fractal curves to model the growth of plants

Fractal curves: A part of the whole curve resembles the whole.

Eg. Let *d* be a command to draw a straight line of a fixed length in the current direction, and + be a command to turn right by 60°, and - be a command to turn left by 60°. The context-free grammar *G*=(*N*, *T*, *P*, *D*) is defined by *N*={*D*}, *T*={*d*,+,-}, *P*={*D*→*D*-*D*++*D*-*D*, *D*→*d*, +→+, -→-}. Generate the curve by the grammar.

(Sol.) $D \Rightarrow D\text{-}D\text{++}D\text{-}D \Rightarrow d\text{-}d\text{++}d\text{-}d \in L(G)$.

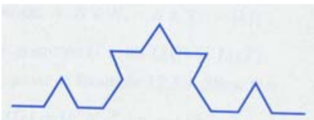
1°:

The string *d-d++d-d* is interpreted as the first-order von Koch snowflake as shown as in the left figure.

2°:

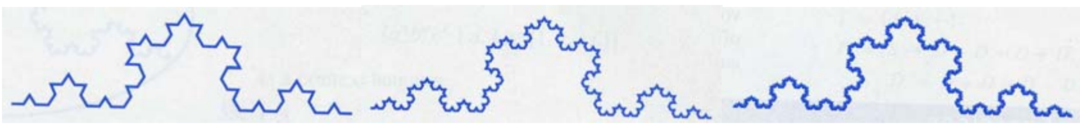
$D \Rightarrow D\text{-}D\text{++}D\text{-}D \Rightarrow D\text{-}D\text{++}D\text{-}D\text{-}D\text{-}D\text{++}D\text{-}D\text{++}D\text{-}D\text{++}D\text{-}D\text{-}D\text{-}D\text{++}D\text{-}D$

$\Rightarrow d\text{-}d\text{++}d\text{-}d\text{-}d\text{-}d\text{++}d\text{-}d\text{++}d\text{-}d\text{++}d\text{-}d\text{-}d\text{-}d\text{++}d\text{-}d$

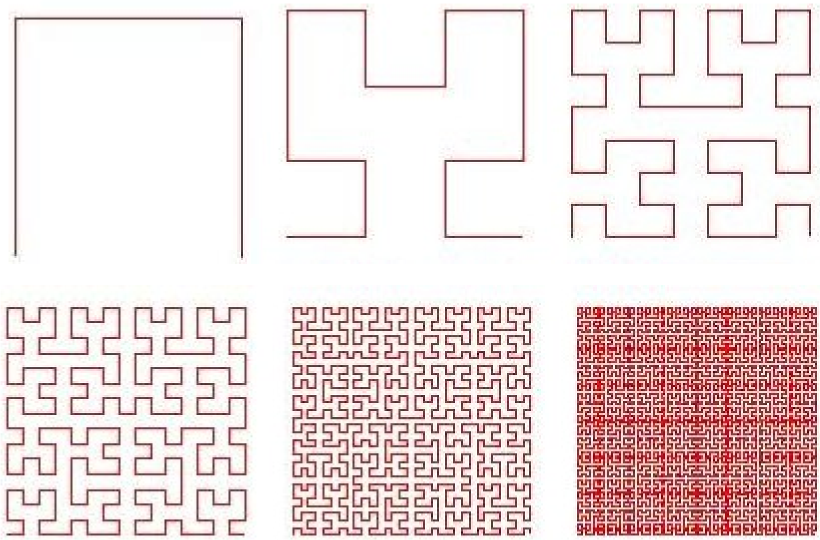


The string *d-d++d-d-d-d++d-d++d-d++d-d-d-d++d-d* is interpreted as the second-order von Koch snowflake as shown as in the left figure.

The other higher- (3rd-, 4th-, and 5th-) order of von Koch snowflakes are as shown as in the following figures.



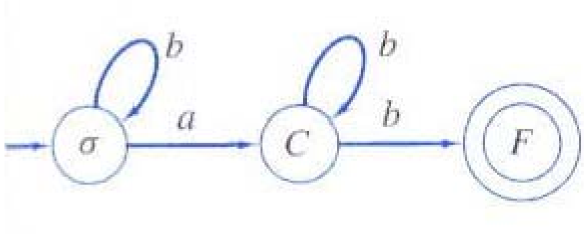
Eg. Some examples of the Hilbert curves can be generated by a special grammar.



Relation of finite automata and grammars

Eg. Draw the corresponding finite automaton or the nondeterministic finite automaton of the grammar G=(N, T, P, σ) is defined by N={σ,C}, T={a,b}, P={σ→bσ, σ→aC, C→bC, C→b}.

(Sol.)

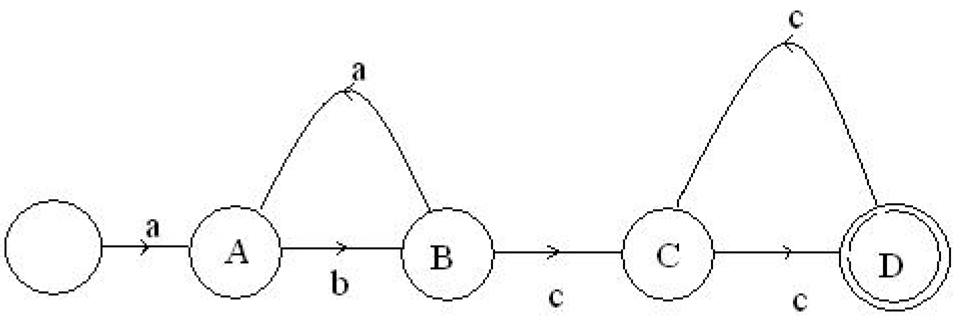


It can accept the strings over [a,b] containing precisely one a and ending with b, like $b^n ab^m$, $n \geq 0$, $m \geq 1$.

Construct a grammar

Eg. Give a grammar that specifies the language $\{(ab)^k c^{2j} \mid k, j \geq 1\}$. [交大資工所]

(Sol.) According to the above description, we draw a nondeterministic finite-state automaton as shown in the following figure.



And then we have G=(N, T, P, σ) that is defined by N={σ,a,b}, T={c}, P={σ→aA, A→bB, B→aA｜ cC, C→cD, D→cC｜ φ}.

Eg. Describe the language ({A,B,S}, {a,b,c}, S, {S→Sa｜ AB, A→aA｜ a, B→b｜ cS)) [交大資工所]