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**

<table>
<thead>
<tr>
<th>S</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Σ₀</td>
<td>σ₀</td>
<td>0</td>
</tr>
<tr>
<td>σ₁</td>
<td>σ₁</td>
<td>1</td>
</tr>
</tbody>
</table>

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 for which the last output was 1 are called accepting states. Let \( \alpha=x₁x₂x₃…xₙ \) be a string. If there exist states \( σ₀, σ₁, σ₂, σ₃, …, σₙ \) satisfying (a) \( σ₀=σ \), (b) \( f(σᵢ,xᵢ)=σᵢ \) for \( i=1, …, n \), (c) \( σₙ ∈ \) 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:

<table>
<thead>
<tr>
<th>( Q(t) )</th>
<th>( Q(t+\tau) )</th>
<th>( J )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( d )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( d )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( d )</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( d )</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>( A(t) )</th>
<th>( B(t) )</th>
<th>( x )</th>
<th>( A(t+\tau) )</th>
<th>( B(t+\tau) )</th>
<th>( y )</th>
<th>( J_A )</th>
<th>( K_A )</th>
<th>( J_B )</th>
<th>( K_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( d )</td>
<td>1</td>
<td>( d )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( d )</td>
<td>0</td>
<td>( d )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( d )</td>
<td>( d )</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( d )</td>
<td>( d )</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( d )</td>
<td>1</td>
<td>1</td>
<td>( d )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( d )</td>
<td>0</td>
<td>1</td>
<td>( d )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( d )</td>
<td>1</td>
<td>( d )</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( d )</td>
<td>1</td>
<td>( d )</td>
<td>1</td>
</tr>
</tbody>
</table>

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:
The sequential logic circuit is as shown 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 $\sigma$.

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: $N \cap T = \emptyset$), a starting symbol $\sigma$, 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 $N=\{\sigma,S\}$, $T=\{a,b\}$, $P=\{\sigma \rightarrow b\sigma$, $\sigma \rightarrow aS$, $S \rightarrow bS$, $S \rightarrow b\}$. Show that $bbbabb$ is in agreement with the grammar $G$, but $aab$ is not in agreement with the grammar $G$.

(Proof) $\sigma \Rightarrow b\sigma \Rightarrow bbb \Rightarrow bbbabb$, \therefore $bbbabb$ is in agreement with the grammar $G$.

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

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

(Proof) $\sigma \Rightarrow 狗 A \Rightarrow 狗咬 B \Rightarrow 狗咬人$
\therefore “狗咬人” is in agreement with the grammar $G$.

$\sigma \Rightarrow 狗 A \Rightarrow 狗咬 B$, \therefore “狗人咬” is not in agreement with the grammar $G$.

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

(Proof) $\sigma \Rightarrow women A \Rightarrow women like B \Rightarrow women like men.$
\therefore “women like men.” is in agreement with the grammar $G$.

$\sigma \Rightarrow women A \Rightarrow women like B$, \therefore “women men like.” is not in agreement with the grammar $G$. 
**Context-sensitive grammar:** Its production is of the form \( aA\beta \rightarrow a\delta\beta \), where \( a, \beta \in (\Sigma \cup T)^* \), \( A \in N \), \( \delta \in (\Sigma \cup T)^*-\{\lambda}\).

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

\( \therefore \) \( \sigma \Rightarrow aAB \Rightarrow aaACB \Rightarrow aaaACCB \Rightarrow aaaaaACCCDc \Rightarrow aaaaaACCCCDc \Rightarrow \ldots \)

\( \Rightarrow a^nA^{n-2}DC \Rightarrow a^nA^{n-1}Cc \Rightarrow a^nDC^{n-2}Cc \Rightarrow a^nDC^{n-1}Cc \Rightarrow a^nDC^nCc \)

\( \Rightarrow \ldots \Rightarrow a^nD^{n-1}Cc^{n-2} \Rightarrow a^nD^nCc^{n-2} \Rightarrow a^nD^nC^n \Rightarrow a^nD^nC^n \), and \( L(G) = \{a^nD^nC^n|n=1,2,\ldots\} \) is a context-sensitive language.

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

Eg. The grammar \( G=\{N, T, P, \sigma\} \) is defined by \( N=\{\sigma\} \), \( T=\{a, b\} \), \( P=\{\sigma \rightarrow a\sigma b, \sigma \rightarrow ab\} \). \( \sigma \Rightarrow a\sigma b \Rightarrow a\sigma\sigma b \Rightarrow aaaS\sigma b \Rightarrow \ldots \Rightarrow a^{n-1}a\sigma b^{n-1} \Rightarrow a^{n-1}abb^{n-1} \Rightarrow a^n b^n, \) and \( L(G) = \{a^n b^n|n=1,2,\ldots\} \) 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 \( N=\{\sigma, S\} \), \( T=\{a, b\} \), \( P=\{\sigma \rightarrow b\sigma, \sigma \rightarrow aS, S \rightarrow bS, S \rightarrow b\} \). \( \sigma \Rightarrow b\sigma \Rightarrow b\sigma \Rightarrow bbb\sigma \Rightarrow \ldots \Rightarrow b^n\sigma \Rightarrow b^n\sigma \Rightarrow b^nabS \Rightarrow b^nabS \)

\( \Rightarrow b^nabbbS \Rightarrow \ldots \Rightarrow b^nab^n \), \( L(G) = \{b^nab^n|m=1,2,\ldots\} \) 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.

\[
\text{<starting symbol>::=<integer>}
\]

\[
\text{<digit>::=0|1|2|3|4|5|6|7|8|9}
\]

\[
\text{<integer>::=<signed integer>|<unsigned integer>}
\]

\[
\text{<signed integer>::=+<unsigned integer>}
\]

\[
\text{<unsigned integer>::=<digit>|<digit><unsigned integer>}
\]

(Sol.)

\[
\text{<integer>::=<signed integer>::=+<unsigned integer>}
\]

\[
::=<\text{digit}<\text{unsigned integer}>::=9<\text{unsigned integer}>
\]

\[
::=9<\text{digit}<\text{unsigned integer}>::=-90<\text{unsigned integer}>::=-90<\text{digit}>::=-901
\]

\( \therefore \) -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 \( G=(\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{D}) \) is defined by \( \mathcal{N}=\{D\} \), \( \mathcal{T}=\{d,+,-\} \), \( \mathcal{P}=\{D \rightarrow D-D++D-D, \ D \rightarrow d, \ + \rightarrow +, \ - \rightarrow -\} \). Generate the curve by the grammar.

(Sol.) \( D \Rightarrow D-D++D-D \Rightarrow d-d++d-d \in L(G) \).

1°:

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

2°:

\[
D \Rightarrow D-D++D-D \Rightarrow D-D++D-D++D-D++D-D++D-D
\Rightarrow d-d++d-d++d-d++d-d++d-d++d-d++d-d
\]

The string \( d-d++d-d++d-d++d-d++d-d++d-d++d-d++d-d++d-d \) 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

Example. Draw the corresponding finite automaton or the nondeterministic finite automaton of the grammar $G=(N, T, P, \sigma)$ defined by $N=\{\sigma, C\}$, $T=\{a, b\}$, $P=\{\sigma \rightarrow b\sigma, \sigma \rightarrow aC, C \rightarrow bC, C \rightarrow b\}$.

(Sol.)

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

Construct a grammar

Example. Give a grammar that specifies the language $\{(ab)^k\ c^j \mid k, j \geq 1\}$. [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 $G=(N, T, P, \sigma)$ that is defined by $N=\{\sigma, a, b\}$, $T=\{c\}$, $P=\{\sigma \rightarrow aA, A \rightarrow bB, B \rightarrow aA \mid cC, C \rightarrow cD, D \rightarrow cC \mid \phi\}$.

Example. Describe the language $\{(A, B, S), \{a, b, c\}, S, \{S \rightarrow Sa \mid AB, A \rightarrow aA \mid a, B \rightarrow b \mid cS\}\}$. [交大資工所]