第一章

1.1 图给出两台 DFA M_1 和 M_2 的状态图. 回答下述有关问题.

a. M_1 的起始状态是 q_1
b. M_1 的接受状态集是{q_2}
c. M_2 的起始状态是 q_1
d. M_2 的接受状态集是 {q_1，q_4}
e. 对输入 aabb,M_1 经过的状态序列是 q_1，q_2，q_3，q_1，q_1
f. M_1 接受字符串 aabb 吗？否
g. M_2 接受字符串 ε 吗？是

1.2 给出练习 2.1 中画出的机器 M_1 和 M_2 的形式描述.

$M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ 其中

1）$Q_1=\{q_1,q_2,q_3,\}$;
2）$\Sigma=\{a, b\}$;
3）δ_1 为：

	a	b
q_1	q_2	q_1
q_2	q_3	q_3
q_3	q_2	q_1

4）q_1 是起始状态
5）$F_1=\{q_2\}$

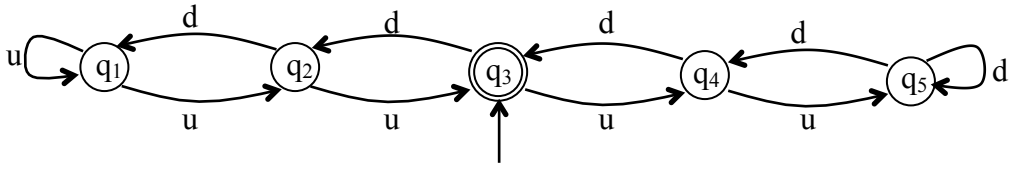
$M_2=(Q_2, \Sigma, \delta_2, q_2, F_2)$ 其中

1）$Q_2=\{q_1,q_2,q_3,q_4\}$;
2）$\Sigma=\{a, b\}$;
3）δ_2 为：

	a	b
q_1	q_1	q_2
q_2	q_3	q_4
q_3	q_2	q_1
q_4	q_3	q_4

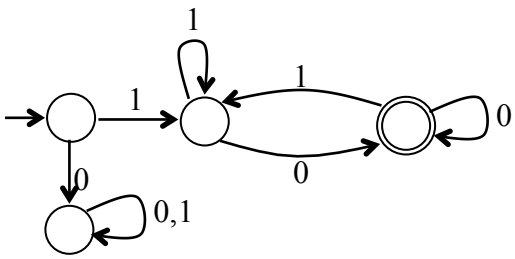
3）q_2 是起始状态
4）$F_2=\{q_1,q_4\}$

1.3 DFA M 的形式描述为 ({q_1，q_2，q_3，q_4，q_5}，{u,d}, δ ,q_3，{q_3}),其中 δ 在表 2-3 中给出。试画出此机器的状态图。

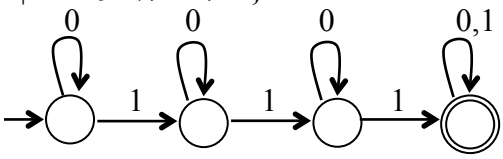


1.6 画出识别下述语言的 DFA 的状态图。

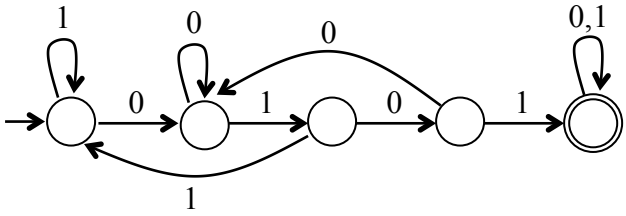
a){w | w 从 1 开始以 0 结束}



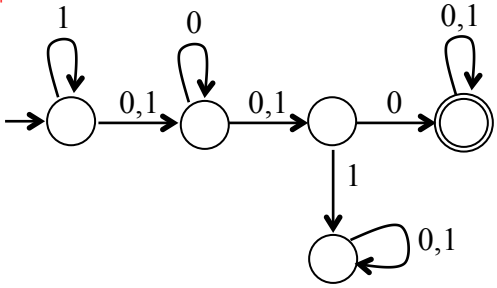
b){w | w 至少有 3 个 1}



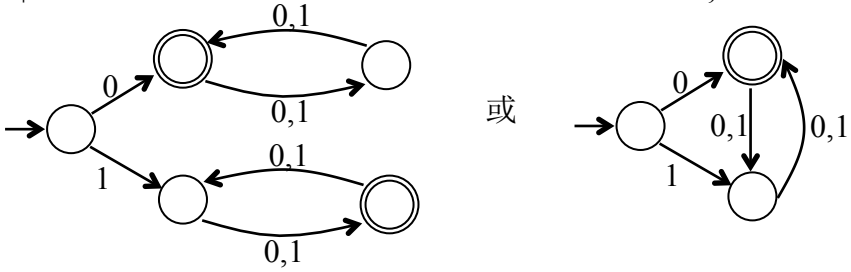
c) {w | w 含有子串 0101}



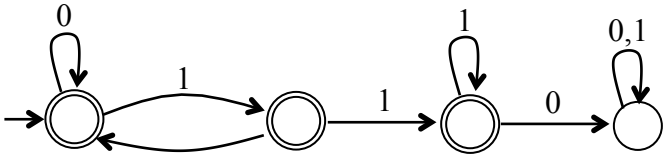
d) {w | w 的长度不小于 3，且第三个符号为 0}



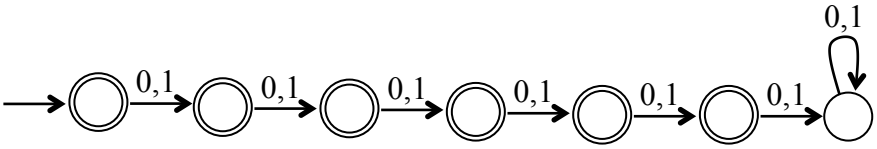
e) {w | w 从 0 开始且为奇长度，或从 1 开始且为偶长度}



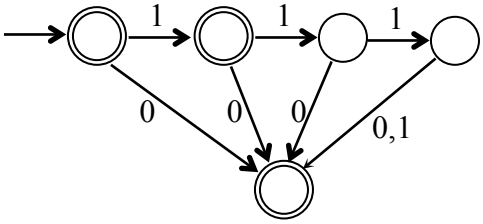
f) {w | w 不含子串 110}



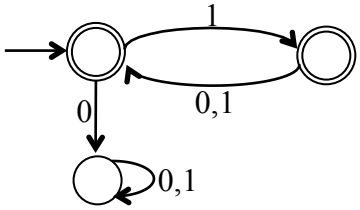
g) {w | w 的长度不超过 5}



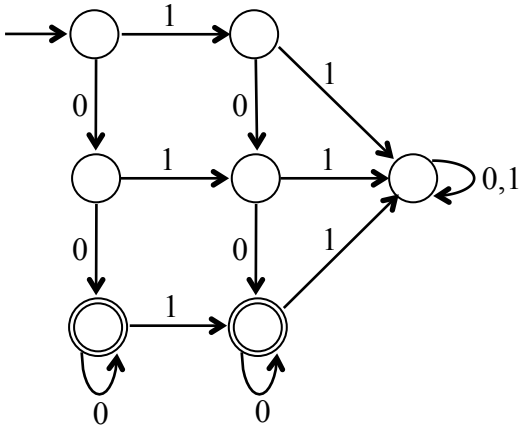
h){w | w 是除 11 和 111 以外的任何字符}



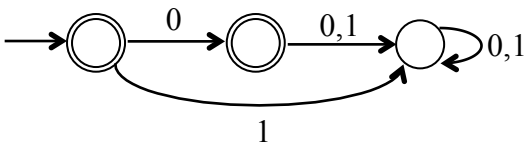
i){w | w 的奇位置均为 1}



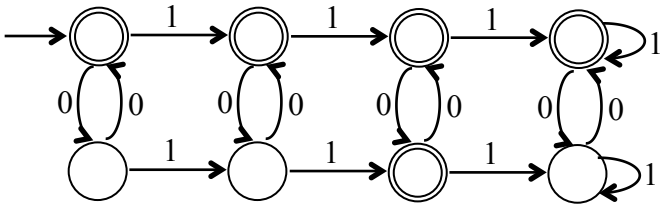
j) {w | w 至少含有 2 个 0，且至多含有 1 个 1}



k) {ε,0}

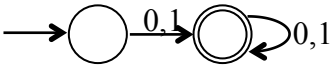


l) {w | w 含有偶数个 0，或恰好两个 1}



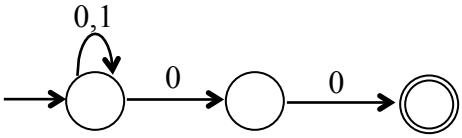
m) 空集

n) 除空串外的所有字符串

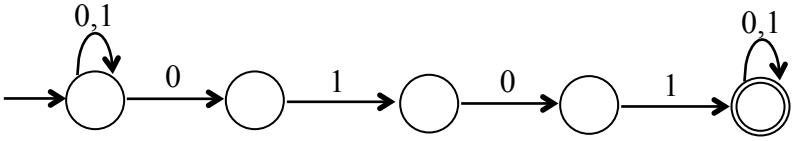


1.7 给出识别下述语言的 NFA，且要求符合规定的状态数。

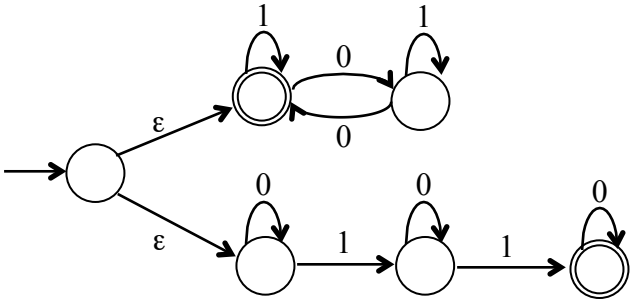
a. {w | w 以 00 结束}，三个状态



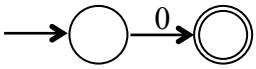
b. 语言 {w | w 含有子串 0101，即对某个 x 和 y，w=x0101y}，5 个状态.



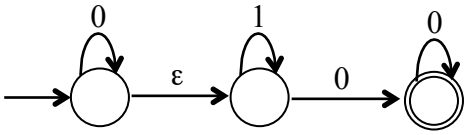
c. 语言 {w | w 含有偶数个 0 或恰好两个 1}，6 个状态。



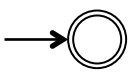
d. 语言 {0}，2 个状态。



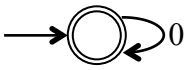
e. 语言 $0^*1^*0^*0$，3 个状态。



f. 语言{ε}，1 个状态。



g. 语言 0^*，1 个状态。



1.11 证明每一台 NFA 都能够转换成等价的只有一个接受状态的 NFA。

证明:设 NFA M={Q，Σ, δ , q_0,F}，F={r_{i1}, ……, r_{ik}}.添加一个状态 p 后，r_{i1}, ……, r_{ik} 分别向 p 引 ε 箭头，将 r_{i1}, ……, r_{ik} 变为非接受状态，p 变为接受状态。又因为添加 ε 箭头不影响 NFA 识别语言，所以命题成立。

1.11 Let $N = (Q, \Sigma, \delta, q_0, F)$ be any NFA. Construct an NFA N' with a single accept state that recognizes the same language as N. Informally, N' is exactly like N except it has ϵ-transitions from the states corresponding to the accept states of N, to a new accept state, q_{accept}. State q_{accept} has no emerging transitions. More formally, $N' = (Q \cup \{q_{\text{accept}}\}, \Sigma, \delta', q_0, \{q_{\text{accept}}\})$, where for each $q \in Q$ and $a \in \Sigma_\epsilon$

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } a \neq \epsilon \text{ or } q \notin F \\ \delta(q,a) \cup \{q_{\text{accept}}\} & \text{if } a = \epsilon \text{ and } q \in F \end{cases}$$

and $\delta'(q_{\text{accept}}, a) = \emptyset$ for each $a \in \Sigma_\epsilon$.

1.14

a. Show that if M is a DFA that recognizes language B, swapping the accept and nonaccept states in M yields a new DFA recognizing the complement of B. Conclude that the class of regular languages is closed under complement.

b. Show by giving an example that if M is an NFA that recognizes language C, swapping the accept and nonaccept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain your answer.

a. Let M' be the DFA M with the accept and non-accept states swapped. We will show that M' recognizes the complement of B, where B is the language recognized by M. Suppose M' accepts x. If we run M' on x we end in an accept state of M'. Because M and M' have swapped accept/non-accept states, if we run M on x, we would end in a non-accept state. Therefore, $x \notin B$. Similarly, if x is not accepted by M', then it would be accepted by M. So M' accepts exactly those strings not accepted by M. Therefore, M' recognizes the complement of B.
Since B could be any arbitrary regular language and our construction shows how to build an automaton to recognize its complement, it follows that the complement of any regular language is also regular. Therefore, the class of regular languages is closed under complement.

b. Consider the NFA in exercise 1.12(a). The string a is accepted by this automaton. If we swap the accept and reject states, the string a is still accepted. This shows that swapping the accept and non-accept states of an NFA doesn't necessarily yield a new NFA recognizing the complement of the original one. The class of languages recognized by NFAs is, however, closed under complement. This follows from the fact that the class of languages recognized by NFAs is precisely the class of languages recognized by DFAs which we know is closed under complement from part (a).

1.14 a 证明：设 M 是一台语言 B 的 DFA，交换 M 的接状态与非接受状态得到一台新的 DFA，则这台新的 DFA 是识别 B 的补集，因此，正则语言类受在补运算下封闭。

b 举例说明：设 M 是一台识别语言 B 的 NFA，交换 M 的接受状态与非接受状态得到一台新的 NFA，这台新的 NFA 不一定识别 B 的补集。NFA 识别的语言类在补集下封闭吗？解释你的回答。

解：

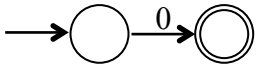
a. M 是 DFA, M 是$\{Q,\Sigma,\delta,q_0,F\}$,令 $N=\{Q,\Sigma,\delta,q_0,Q-F\}$,设 $\omega=\omega_1\omega_2\ldots\omega_n$ 是字母表上任意字符串，因为 M 与 N 均为 DFA,所以必然存在 Q 中状态序列 $r_0,r_1,\ldots,r_n$,使得：$r_0=q_0$, $\delta(r_i,\omega_{i+1})=r_{i+1}$, i=0,…,n-1

1)若 $r_n\in F$ 则 $\omega\in B$;

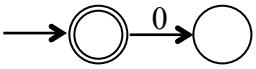
2)若 $r_n\notin F$,则 $r_n\in Q-F$,即 N 接受 ω，若 $\omega\notin B$,

所以 N 接受 B 的补集，即 B 的补集正则。

所以，正则语言类在补运算下封闭。

b. 设 B 为{0}。NFA M： 

可识别它。

依题对它作变换，得到 N： 

则 N 识别的语言{ε}不是 B 的子集。所以交换 M 的接受状态与非接受状态得到的新的 NFA 不一定识别 B 的补集。

但是由于 NFA 识别的语言类与 DFA 识别的语言类相同，即正则语言类。由 a 的证明，正则语言类在补运算封闭，可知，NFA 识别的语言类---正则语言类在补运算下封闭。

若 NFA 识别语言 A，必有 等价的 DFA 识别 A,从而由 a 知，可交换 DFA 的接受与非接受状态构造识别 A 的补集的 DFA,则必有等价的 NFA 识别 A 的补集。只是，该 NFA 未必有原 NFA 交换接受与非接受状态构造而成。

1.15 给出一个反例，说明下述构造不能证明定理 2.24，即正则语言类在星号运算下封闭。设 N=（$Q_1,\Sigma,\delta_1,q_1,F_1$）识别 A_1。如下构造 N=（$Q_1,\Sigma,\delta_1,q_1,F_1$）。N 应该识别 A_1^*。

a. N 的状态集是 N_1 的状态集。

b. N 的起始状态是 N_1 的起始状态相同。

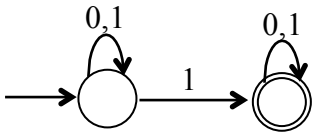
c. $F=\{q_1\}\cup F_1$。F 的接受状态是原来的接受状态加上它的起始状态。

d. 定义 δ 如下：对每一个 q 属于 Q 和每一个 a 属于 Σ。

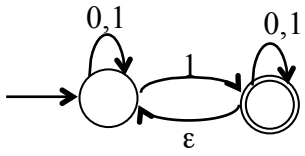
$$\delta(q,a)=\begin{cases}\delta_1(q,a), & \text{若} q\notin F_1 \text{ 或 } a\neq\varepsilon\\ \delta_1(q,a)\cup\{q_1\}, & \text{若} q\in F_1 \text{ 且 } a\neq\varepsilon\end{cases}$$

解：设 N_1 识别语言 A={至少含有一个 1}，其中输入字母表为{0，1}，可知 A^*={空串或至少含有一个 1}。

N_1:



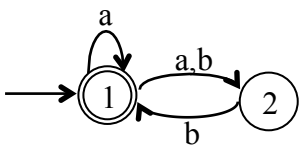
N:



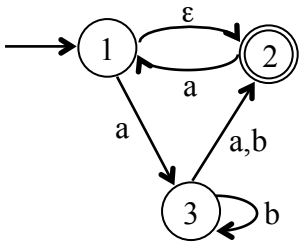
按上述规定构造 N 的状态图如上。可以看出 L(N)={0,1}*不等于 A*. 所以如此构造的 N 不一定识别 A*.

1.16 使用定理 2.19 中给出的构造，把下图中的两台非确定型有穷自动机转换成等价的确定型有穷自动机。

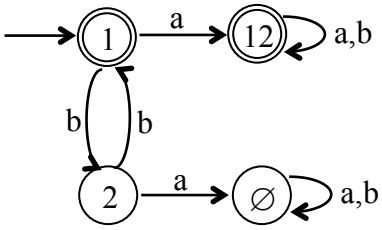
a)，



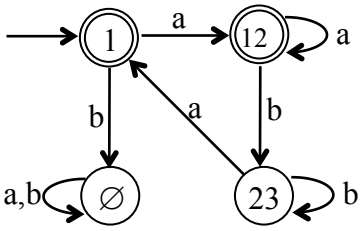
b)，



解：a),



b)



1.18 给出生成练习 1.6 中语言的正则表达式。(注: 答案不唯一)

a. {w | w 从 1 开始以 0 结束} $1\Sigma^*0$.
b. {w | w 至少有 3 个 1} $\Sigma^*1\Sigma^*1\Sigma^*1\Sigma^*$.
c. {w | w 含有子串 0101} $\Sigma^*0101\Sigma^*$.
d. {w | w 的长度不小于 3，且第三个符号为 0} $\Sigma\Sigma0\Sigma^*$.
e. {w | w 从 0 开始且为奇长度，或从 1 开始且为偶长度} $0(\Sigma\Sigma)^*\cup1\Sigma(\Sigma\Sigma)^*$.
f. {w | w 不含子串 110} $(0\cup10)^*1^*$.
g. {w | w 的长度不超过 5} $\varepsilon\cup\Sigma\cup\Sigma\Sigma\cup\Sigma\Sigma\Sigma\cup\Sigma\Sigma\Sigma\Sigma\cup\Sigma\Sigma\Sigma\Sigma\Sigma$.
h. {w | w 是除 11 和 111 以外的任何字符} $0\Sigma^*\cup10\Sigma^*\cup110\Sigma^*\cup111\Sigma\Sigma^*$.
i. {w | w 的奇位置均为 1} $(1\Sigma)^*(\varepsilon\cup1)$.
j. {w | w 至少含有 2 个 0，且至多含有 1 个 1} $0^*(00\cup010\cup001\cup100)\,0^*$.
k. {ε,0}. $\varepsilon\cup0$.
l. {w | w 含有偶数个 0，或恰好两个 1} $(1^*01^*0)^*1^*\cup0^*10^*10^*$.
m. 空集. $\varnothing$.
n. 除空串外的所有字符串 $\Sigma\Sigma^*$.

1.20 对下述每一个语言，给出 4 个字符串，2 个是这个语言的成员，2 个不是这个语言的成员。这里假设字母表 Σ={a,b}.

a. a^*b^*　　成员：ab，aab　　非成员：aba，ba

b. $a(ba)^*$　　成员：ab，abab　　非成员：abb，aa

c. $a^* \cup b^*$　　成员：aaa，bbb　　非成员：ab，ba

d. $(aaa)^*$　　成员：aaa，aaaaaa　　非成员：a，aa

e. $\Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$　　成员：aba，aaba　　非成员：aa，abb

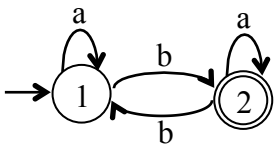
f. aba∪bab　　成员：aba，bab　　非成员：a，b

g. (ε∪a)b　　成员：b，ab　　非成员：a，bb

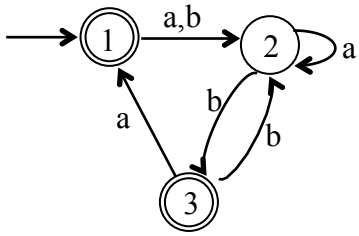
h. (a∪ba∪bb) Σ^*　　成员：a，bb　　非成员：ε，b

1.21 使用引理 2.32 中叙述的过程，把图 2-38 中的有穷自动机转换成正则表达式。

a),



b),



解：　a) $a^*b(a \cup ba^*b)^*$

b) $\varepsilon \cup (a \cup b)a^*b[(aa \cup ab \cup b)a^*b]^*(a \cup \varepsilon)$.

(注：答案不唯一)

A1.23 Let B be any language over the alphabet Σ. Prove that $B = B^+$ iff $BB \subseteq B$.

1.23 We prove both directions of the "iff."

($\rightarrow$) Assume that $B = B^+$ and show that $BB \subseteq B$.

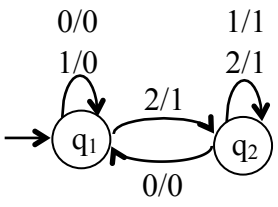
For every language $BB \subseteq B^+$ holds, so if $B = B^+$, then $BB \subseteq B$.

($\leftarrow$) Assume that $BB \subseteq B$ and show that $B = B^+$.

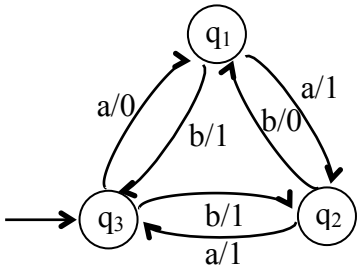
For every language $B \subseteq B^+$, so we need to show only $B^+ \subseteq B$. If $w \in B^+$, then $w = x_1 x_2 \cdots x_k$ where each $x_i \in B$ and $k \geq 1$. Because $x_1, x_2 \in B$ and $BB \subseteq B$, we have $x_1 x_2 \in B$. Similarly, because $x_1 x_2$ is in B and x_3 is in B, we have $x_1 x_2 x_3 \in B$. Continuing in this way, $x_1 \cdots x_k \in B$. Hence $w \in B$, and so we may conclude that $B^+ \subseteq B$.

The latter argument may be written formally as the following proof by induction.
Assume that $BB \subseteq B$.
Claim: For each $k \geq 1$, if $x_1, \ldots, x_k \in B$, then $x_1 \cdots x_k \in B$.
Basis: Prove for $k = 1$. This statement is obviously true.
Induction step: For each $k \geq 1$, assume that the claim is true for k and prove it to be true for $k + 1$.
If $x_1, \ldots, x_k, x_{k+1} \in B$, then by the induction assumption, $x_1 \cdots x_k \in B$. Therefore, $x_1 \cdots x_k x_{k+1} \in BB$, but $BB \subseteq B$, so $x_1 \cdots x_{k+1} \in B$. That proves the induction step and the claim. The claim implies that if $BB \subseteq B$, then $B^+ \subseteq B$.

1.24 有穷状态转换器(FST)是确定性有穷自动机的一种类型。它的输出是一个字符串，而不仅仅是接受或拒绝。图 2—39 是两台有穷状态状态转换器 T_1 和 T_2 的状态图。



T_1



T_2

FST 的每一个转移用两个符号标记，一个指明该转移的输入符号，另一个指明输出符号。两个符号之间用斜杠“/”把它们分开。在 T_1 中，从 q_1 到 q_2 的转移有输入符号 2 和输出符号 1。某些转移可能有多对输入-输出，比如 T_1 中从 q_1 到它自身的转移。FST 在对输入串 w 计算时，从起始状态开始，一个接一个地取输入符号 $w_1 \ldots w_n$，并且比照输入标记和符号序列 $w_1 \ldots w_n = w$ 进行转移。每一次沿一个转移走一步，输出对应的输出符号。例如，对输入 2212011，机器 T_1 依次进入状态 $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ 和输出 1111000。对输入 abbb，T_2 输出 1001。给出在下述每一小题中机器进入的状态序列和产生的输出。

a. T_1 对输入串 011, 输出：000, 状态序列：q_1, q_1, q_1, q_1.

b. T_1 对输入串 211, 输出：111, 状态序列：q_1, q_2, q_2, q_2.

c. T_1 对输入串 0202, 输出：0101, 状态序列：q_1, q_1, q_2, q_1, q_2。

d. T_2 对输入串 b, 输出：1, 状态序列：q_1, q_3.

e. T_2 对输入串 bbab, 输出：1111, 状态序列：q_1, q_3, q_2, q_3, q_2.

f. T_2 对输入串 bbbbbb, 输出：110110, 状态序列：$q_1, q_3, q_2, q_1, q_3, q_2, q_1$。

g. T_2 对输入串ε, 输出：ε, 状态序列：q_1。

1.25 给出有穷状态转换器的形式定义。

解：有穷状态转换器 FST 是一个五元组$(Q, \Sigma, \Gamma, \delta, q0)$

1) Q 是一个由穷集合，叫做状态集
2) Σ 是一个由穷集合，叫做输入字母表
3) Γ 是一个由穷集合，叫做输出字母表
4) δ：$Q \times \Sigma \rightarrow Q \times \Gamma$ 是转移函数
5) q_0 是起始状态

FST 计算的形式定义：

$M=(Q, \Sigma, \Gamma, \delta, q_0)$是一台由穷状态转换器，$w=w_1w_2 \ldots w_n$ 是输入字母表上的一个字符串。若存在 Q 中的状态序列：$r_0, r_1, \ldots r_n$ 和输出字母表上的一个字符串 $s=s_1 \ldots s_n$, 满足下述条件：

1）$r_0=q_0$;

2）$\delta(r_i, w_{i+1})=(r_{i+1}, s_{i+1})$, i=0,1,…,n-1

则 M 在 W 的输入下输出 s.

1.26 利用你给练习 2.20 的答案，给出练习 2.19 中画出的机器 T_1 和 T_2 的形式描述。

解：有穷状态转换器 T_1 的形式描述为：

$T_1=(\{q_1, q_2\}, \{0，1，2\}, \delta, q_1, \{0,1\})$

其中转移函数为：

	0	1	2
q_1	$q_1/0$	$q_1/0$	$q_2/1$
q_2	$q_1/0$	$q_2/1$	$q_2/1$

有穷状态转换器 T_2 的形式描述为：

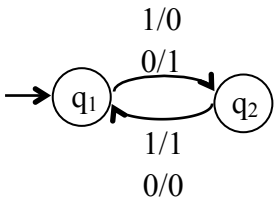
$T_2=(\{\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{0,1\})$

其中转移函数为：

	a	b
q_1	$q_2/1$	$q_3/1$
q_2	$q_3/1$	$q_1/0$
q_3	$q_1/0$	$q_2/1$

1.27 给出一台具有下述行为的 FST 的状态图。它的输入、输出字母表都是{0,1}。它输出的字符串与输入的字符串偶数为相同、奇数位相反。例如，对于输入 0000111，它应该输出 1010010。

解：



1.29 利用泵引理证明下述语言不是正则的。

a. $A_1=\{0^n1^n2^n \mid n\geq 0\}$。

证明：假设 A_1 是正则的。设 p 是泵引理给出的关于 A_1 的泵长度。

令 $S=0^p1^p2^p$,

∵S 是 A_1 的一个成员且 S 的长度大于 p，所以泵引理保证 S 可被分成 3 段 S=xyz 且满足泵引理的 3 个条件。根据条件 3，y 中只含 0，xyyz 中，0 比 1、2 多，xyyz 不是 A_1 的成员。违反泵引理的条件 1，矛盾。

∴A_1 不是正则的。

b. $A_2=\{www \mid w\in\{a,b\}^*\}$.

证明：假设 A_2 是正则的。设 p 是泵引理给出的关于 A_2 的泵长度。

令 $S=a^pba^pba^pb$,

∵S 是 A_2 的一个成员且 S 的长度大于 p，所以泵引理保证 S 可被分成 3 段 S=xyz 且满足泵引理的 3 个条件。根据条件 3，y 中只含 a，所以 xyyz 中第一个 a 的个数将比后两个 a 的个数多，故 xyyz 不是 A_2 的成员。违反泵引理的条件 1，矛盾。

∴A_2 不是正则的。

c. $A_3=\{a^{2^n} \mid n\geq 0\}$.(在这里，$a^{2^n}$ 表示一串 2^n 个 a .)

证明：假设 A_3 是正则的。设 p 是泵引理给出的关于 A_3 的泵长度。

令 $S= a^{2^p}$，

∵S 是 A_2 的一个成员且 S 的长度大于 p，所以泵引理保证 S 可被分成 3 段 S=xyz 且满足泵引理的 3 个条件。即对任意的 i≥0，xy^iz 都应在 A_3 中，且 xy^iz 与 $xy^{i+1}z$ 的长度都应是 2 的幂. 而且 $xy^{i+1}z$ 的长度应是 xy^iz 的长度的 2^n 倍(n≥1)。于是可以选择足够大的 i，使得 $|xy^iz|=2^n>p$. 但是 $|xy^{i+1}z|-|xy^iz|=|y|\leq p$. 即 $|xy^{i+1}z|<2^{n+1}$, 矛盾。

∴A_3 不是正则的。

1.30 下面“证明”0^*1^*不是正则语言，指出这个“证明”中的错误。(因为 0^*1^* 是正则的，所以一定错误。) 采用反证法证明。假设 0^*1^*是正则的。令 P 是泵引定理给出的关于 0^*1^*的泵长度。取 S 为字符串 0^p1^p。S 是 0^*1^*的一个成员，但是例 2.38 已证明 S 不能被抽取。于是得到矛盾，所以 0^*1^*不是正则的。

解：在例 2.38 中的语言是$\{0^n1^n \mid n\geq 0\}$，取 S 为字符串 0^p1^p，S 确实不能被抽取；但针对语言 0^*1^*，S 是能被抽取的。将 S 分成三段 S=xyz，由泵引理的条件 3，y 仅包含 0，而 xy^iz 属于语言 0^*1^*，即 S 能被抽取，故题设中的“证明”不正确。

1.31 对于任意的字符串 $w=w_1w_2\ldots w_n$，w 的反转是按相反的顺序排列 w 得到的字符串，记作 w^R，即 $w^R=w_n\ldots w_2w_1$。对于任意的语言 A，记 $A^R=\{w^R \mid \in A\}$ 证明：如果 A 是正则的，则 A^R 也是正则的。

证明：

因为 A 是正则语言，所以可以用 NFA 来表示该语言，现在来构造 A^R 的 NFA，将 NFA A 中的接受态变为中间态，起始态变为接受态，再添加一新的起始态，并用 ε 箭头连接至原来的接受态，其它所有的箭头反向。 经过变换后得到 NFA 变成描述 A^R 的 NFA.

1.32 令

$$\Sigma_3=\left\{\begin{pmatrix}0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\ldots,\begin{pmatrix}1\\1\\1\end{pmatrix}\right\}$$

$\sum_3$ 包括所有高度为 3 的 0 和 1 的列向量。$\sum_3$ 上的字符串给出三行 0 和 1 的字符串。把每一行看作一个二进制数，令

$$B=\{ w\in\sum_3{}^* \mid w \text{ 最下面一行等于上面两行的和}\}$$

例如，

$$\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\in B,\qquad \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}\notin B$$

证明 B 是正则的。

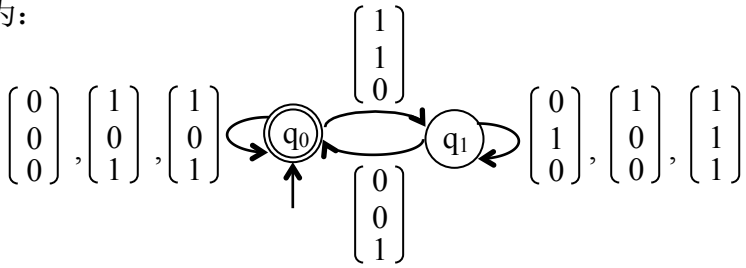
证明：由题设 B 的定义可知，w 上面两行之和减去下面一行结果为零，由此可设计 NFA M (Q, Σ, δ, q0, F)，其中$\sum=\sum_3$。$Q=\{q_0,q_1\}$。q_0 状态表示上一次运算的进位为 0，q_1 状态表示上一次运算的进位为 1。

δ 由下表给出：

	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\1\end{bmatrix}$	$\begin{bmatrix}1\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\1\end{bmatrix}$	$\begin{bmatrix}1\\1\\0\end{bmatrix}$	$\begin{bmatrix}1\\1\\1\end{bmatrix}$
q_0	$\{q_0\}$	∅	∅	$\{q_0\}$	∅	$\{q_0\}$	$\{q_1\}$	∅
q_1	∅	$\{q_0\}$	$\{q_1\}$	∅	$\{q_1\}$	∅	∅	$\{q_1\}$

$F=\{q_0\}$

状态图为：



所以可知自动机 M 识别的是语言 B^R，因此 B^R 是正则的。由题 2-24 的结论可知 B 也是正则的。

1.33 令

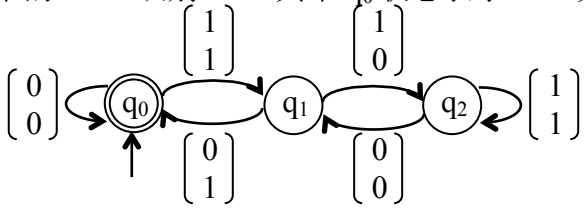
$$\Sigma_2=\left\{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}\right\}$$

Σ_2 包含所有高度为 2 的 0 和 1 的列。Σ_2 上的字符串给出两行 0 和 1 的字符串。把每一行看作一个二进制数，令

$$C=\{\ w\in\sum_2^*\mid w\text{ 下面一行等于上面一行的 3 倍 }\}。$$

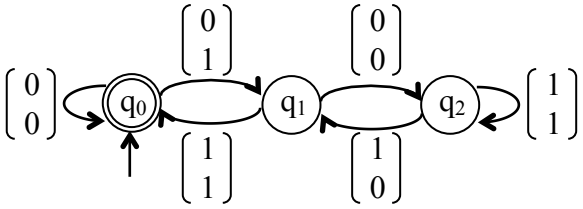
证明 C 是正则的。可以假设已知问题 2.24 中的结果。

证明：如下的 NFA 识别 C^R：其中 q_0 状态表示上一次运算的进位为 0，



q_1 状态表示上一次运算的进位为 1, q_2 状态表示上一次运算的进位为 2。

如下的 NFA 识别 C：其中状态 q_0,q_1,q_2 分别表示目前读到的下面的数减上面



的数的 3 倍余 0,1,2 的情况。

1.34 令 $\Sigma_2=\left\{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}\right\}$

Σ_2 包含所有高度为 2 的 0 和 1 的列。Σ_2 上的字符串给出两行 0 和 1 的字符串。把每一行看作一个二进制数，令

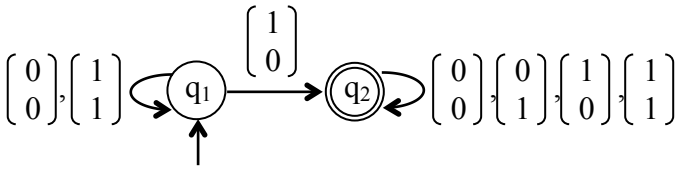
$$D=\{\ w\in\sum_2{}^*\mid w \text{ 上一行大于下一行 }\}。$$

证明 D 是正则的。

证明：由题设可设计自动机 $M=(Q,\Sigma,\delta,q,F)$，其中 $Q=\{q_1,q_2\}$，$F=\{q_2\}$，

转移函数与状态图为：

	$\begin{matrix}0\\0\end{matrix}$	$\begin{matrix}0\\1\end{matrix}$	$\begin{matrix}1\\0\end{matrix}$	$\begin{matrix}1\\1\end{matrix}$
q_1	$\{q_1\}$	$\varnothing$	$\{q_2\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$



1.35 设 Σ_2 与问题 2.26 中的相同。把每一行看作 0 和 1 的字符串，令 $E=\{w\in\sum_2{}^*\mid$ w 的下一行是上一行的反转$\}$，证明 E 不是正则的。

证明：假设 E 是正则的，令 p 是有泵引理给出的泵长度。

选择字符串 $s=\begin{bmatrix}1\\0\end{bmatrix}^p\begin{bmatrix}0\\1\end{bmatrix}^p$。于是 s 能够被划分为 xyz 且满足泵引理的条件。

根据条件 3，y 仅能取包含 $\begin{bmatrix}1\\0\end{bmatrix}$ 的部分，当添加一个 y 时，xyyz 不属于 E. 所以 E 不是正则的.

1.36 令 $B_n=\{a_k\mid k$ 是 n 的整数倍$\}$。证明对于每一个 $n\geq1$, B_n 是正则的。

证明：设字母表Σ为{a}，则 a^n 是一正则表达式。所以，$(a^n)^*$也是正则表达式。由题意 $B_n=(a^n)^*$, 即 B_n 可以用正则表达式表示。所以，B_n 也是正则的。

1.37 令 $C_n=\{x\mid x$ 是一个二进制数，且是 n 的整数倍$\}$。证明对于每一个 $n\geq1$, C_n 是正则的。

证明：下面的 DFA 识别 C_n：(正向读)

M=(Q, {0,1} , δ , q_0 , F), 其中 Q＝{0,1,2,…，n-1},
δ(i,1)=2i+1 mod n, δ(i,0)=(2i mod n), i=0,1,2,…，n-1,
起始状态为 0, F＝{0}.
这里 i 表示当前数 mod n 余 i.

下面的 DFA 识别 C_n^R：(反向读)
M=(Q, {0,1} , δ , q_0 , F), 其中 Q＝{(i,j)|i,j=0,1,2,…，n-1},
δ((i,j),1)=(2i mod n, (2i+j)mod n), δ((i,j),0)=(2i mod n, j), i,j=0,1,2,…，n-1
起始状态为(1,0), F＝{ (i,0) | i=0,1,…，n-1}.
这里(i,j)表示当前数 mod n 余 j, 而下一位所表示单位数 mod n 余 i(例如，若读下一位将达到 k 位，则下一位所表示单位数为 10^{k-1}).

1.38 考虑一种叫做全路径 NFA 的新型有穷自动机。一台全路径 NFA M 是 5 元组 (Q, ∑, δ, q_0, F). 如果 M 对 x∈$\sum^*$的每一个可能的计算都结束在 F 中的状态，则 M 接受 x。注意，相反的，普通的 NFA 只需有一个计算结束在接受状态，就接受这个字符串。证明：全路径 NFA 识别正则语言。

证明：一个 DFA 一定是一个全路径 NFA。所以下面只需证明，任意全路径 NFA，都有一个与之等价的 DFA。

设有一全路径 NFA N=(Q, Σ, δ, q_0, F)，构造一个新与 N 等价的全路径 NFA N_1＝(Q_1, Σ, δ_1, q_0, F), 其中 Q_1=Q∪{s}, s∉Q。对于任意 q∈Q_1, a∈Σ_ε

$$\delta_1(q,a)=\begin{cases}\delta(q,a), & q\neq s, \text{ 且 } \delta(q,a)\neq\varnothing;\\ \{s\}, & q\neq s, \text{ 且 } \delta(q,a)=\varnothing;\\ \{s\}, & q=s.\end{cases}$$

现在来构造一个与全路径 NFA N_1 等价的 DFA M＝(Q_2, Σ, δ_2, q_1, F_2). 其中

1) Q_2=Power(Q_1), 即 Q_1 的所有子集组成的集合(也即 Q_1 的幂集)；

2) 定义函数 E: Q_2→Q_2 为：对任意 R∈Q_2, $E(R)=\bigcup_{r\in R}\delta_1(r,\varepsilon)$；

3) q_1=E(q_0);

4) 对于任意的 R 属于 Q_2, a 属于 Σ, $\delta_2(R,a)=E\left(\bigcup_{r\in R}\delta_1(r,a)\right)$.

5) F_2={ R∈Q_2 | R⊆F}。

综上所述，DFA 等价于全路径 NFA。

1.39 The construction in Theorem 1.54 shows that every GNFA is equivalent to a GNFA with only two states. We can show that an opposite phenomenon occurs for DFAs. Prove that for every $k > 1$, a language $A_k \subseteq \{0,1\}^*$ exists that is recognized by a DFA with k states but not by one with only $k-1$ states.

Solution: Consider $A_k = \{0^{k-2}\}$, $k > 2$. Then A_k is recognized by M_k with states $q_0, \ldots, q_{k-1}, q_k$, where q_{k-1} is the only accepting state with an arrow pointing to q_k and q_k self-loops. So A_k is recognized with k many states. Now suppose that we have a DFA with $1 \leq l < k$ many states. Using a Pumping Lemma-like argument, we can show that after reading the first 0^l many zeros, there are $0 \leq i < j \leq l$ such that the state after reading 0^i is the same as the state after reading 0^j, so the DFA "looses count", and does no longer "know" if it has seen i or j zeros, so it cannot decide A_k.

1.39 Let $A_k = \Sigma^*0^{k-1}0^*$. A DFA with k states can recognize A_k. In addition to the start state, it has one state for each number of 0s it has read since the last 1. After $k-1$ 0s the machine enters a accept state whose transitions are self-loops.

In any DFA with fewer than k states, two of the k strings 1, 10, ... , 10^{k-1} must cause the machine to enter the same state, by the pigeon hole principle. But then, if we add to both of these strings enough 0s to cause the longer of these two strings to have exactly $k-1$ 0s, the two new strings will still cause the machine to enter the same state, but one of these strings is in A_k and the other is not. Hence, the machine must fail to produce the correct accept/reject response on one of these strings.

1.39 证明：设对于每一个 k>1，A_k={ w | w 包含子串 1^{k-1}}⊆{1}*，下面证明 A_k 能被一台 k 个状态的 DFA 识别，而不能被只有 k－1 个状态的 DFA 识别。

显然，A_k 能被 k 个状态的 DFA

$$M=(\{q_1,q_2,\cdots,q_k\}, \{1\}, \delta, q_1, \{q_k\}).$$

识别，其中$\delta(q_i,1)=q_{i+1}(i=1,2,\cdots,k-1)$, $\delta(q_k,1)=q_k$。

假设 A_K 可以被只有 k-1 个状态的 DFA M_1 识别。

考虑这样一个输入串 $s=1^{k-1}$，设 M 识别 s 的状态序列是 $r_1, r_2,\ldots r_k$,由于 M 的状态只有 k-1 个，根据鸽巢原理，$r_1,r_2,\ldots,r_k$ 中必有两个重复的状态，假设是 $r_i=r_j$ $(0\le i<j\le k)$，此时可知，字符串 $x=1^{j-i}$ 把 M 从 ri 带到了 rj。由于 ri,rj 是同一个状态，所以去掉 s 中的 x 子串，M 仍可识别 s 的剩余部分，即 M 识别 $1^{k-1-(j-i)}$，这与 M_1 识别 A_k 相矛盾。故 A_k 不能被只有 k-1 个状态的 DFA M 识别。

所以，对于每一个 k>1，存在 A_K，使得 AK 能被 K 个状态的 DFA 识别，而不能被只有 k-1 个状态的 DFA 识别。

1.40 如果存在字符串 z 使得 xz=y,则称字符串 x 是字符串 y 的前缀。如果 x 是 y 的前缀且 x≠y,则称 x 是 y 的真前缀。下面每小题定义一个语言 A 上的运算。证明：正则语言类在每个运算下封闭。

a) NOPREFIX(A)={ω∈A|ω 的任意真前缀都不是 A 的元素}

b)NOEXTEND(A)={ ω∈A|ω 不是 A 中任何字符串的真前缀}

证明：假设 DFA M=(Q, Σ, δ, q_0, F)识别语言 A。

a) 构造 NFA N_1=(Q, Σ, δ_1, q_0, F)识别语言 NOPREFIX(A)。其中，

对任意 q∈F, a ∈Σ$_\varepsilon$，
$$\delta_1(q,a)=\begin{cases}\{\delta(q,a)\}, q\in Q-F, a\neq\varepsilon;\\ \varnothing, \quad q\in F;\\ \varnothing, \quad q\in Q, a=\varepsilon;\end{cases}$$

所以，即 NOPREFIX(A)为正则语言，亦即正则语言类 A 在 NOPREFIX(A)运算下封闭。■

b) 构造 NFA N_2=(Q, Σ, δ, q_0, F_2)识别语言 NOEXTEND(A)。F_2 构造如下：对 M 中的任一接受状态 q_i，若存在一条从它出发到达某接受状态(含本身)的路径，则将状态 q_i 改为非接受状态；最后剩下的接受状态集记为 F_2. 所得的 NFA N_2 即识别 NOEXTEND(A)。所以，NOEXTEND(A)为正则语言，亦即正则语言类 A 在运算 NOEXTEND(A)下封闭。■

1.41 For languages A and B, let the ***perfect shuffle*** of A and B be the language

$$\{w | \ w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}.$$

Show that the class of regular languages is closed under perfect shuffle.

Answer: Let $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two DFAs that recognize A and B, respectively. Here, we shall construct a DFA $D = (Q, \Sigma, \delta, q, F)$ that recognizes the perfect shuffle of A and B.

The key idea is to design D to alternately switch from running D_A and running D_B after each character is read. Therefore, at any time, D needs to keep track of (i) the current states of D_A and D_B and (ii) whether the next character of the input string should be matched in D_A or in D_B. Then, when a character is read, depending on which DFA should match the character, D makes a move in the corresponding DFA accordingly. After the whole string is processed, if both DFAs are in the accept states, the input string is accepted; otherwise, the input string is rejected.

Formally, the DFA D can be defined as follows:

(a) $Q = Q_A \times Q_B \times \{A, B\}$, which keeps track of all possible current states of D_A and D_B, and which DFA to match.

(b) $q = (q_A, q_B, A)$, which states that D starts with D_A in q_A, D_B in q_B, and the next character read should be in D_A.

(c) $F = F_A \times F_B \times \{A\}$, which states that D accepts the string if both D_A and D_B are in accept states, and the next character read should be in D_A (i.e., last character was read in D_B).

(d) δ is as follows:

i. $\delta((x, y, A), a) = (\delta_A(x, a), y, B)$, which states that if current state of D_A is x, the current state of D_B is y, and the next character read is in D_A, then when a is read as the next character, we should change the current state of A to $\delta_A(x, a)$, while the current state of B is not changed, and the next character read will be in D_B.

ii. Similarly, $\delta((x, y, B), b) = (x, \delta_B(y, b), A)$.

1.42 For languages A and B, let the ***shuffle*** of A and B be the language

$$\{w | \ w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma^*\}.$$

Show that the class of regular languages is closed under shuffle.

Answer: Let $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be two DFAs that recognize A and B, respectively. Similar to the previous question, we shall prove by construction. However, the key difference is that D *may* now switch from running D_A and running D_B after each character is read. To allow this flexibility and simplify the construction, we design an NFA $N = (Q, \Sigma, \delta, q, F)$ that recognizes the shuffle of A and B instead of directly designing a DFA.

At any time, N needs to keep track of the current states of D_A and D_B. Then, when a character is read, N may make a move in D_A or D_B accordingly. After the whole string is processed, if both DFAs are in the accept states, the input string is accepted; otherwise, the input string is rejected. In addition, N should also accept the empty string.

Formally, the NFA N can be defined as follows:

(a) $Q = (Q_A \times Q_B) \cup \{q_0\}$, where $Q_A \times Q_B$ keeps track of all possible current states of D_A and D_B, and q_0 denotes the state when nothing is read.

(b) $q = q_0$.

(c) $F = (F_A \times F_B) \cup \{q_0\}$, which states that N accepts the string if both D_A and D_B are in accept states, or N accepts the empty string.

(d) δ is as follows:

i. $\delta(q_0, \varepsilon) = (q_A, q_B)$, which states that at the start state q_0, N can make D_A in q_A and D_B in q_B without reading anything.

ii. $(\delta_A(x, a), y) \in \delta((x, y), a)$, which states that if current state of D_A is x, the current state of D_B is y, then when a is read as the next character, we can change the current state of A to $\delta_A(x, a)$, while the current state of B is not changed.

iii. Similarly, $(x, \delta_B(y, a)) \in \delta((x, y), a)$.

[A]**1.44** Let B and C be languages over $\Sigma = \{0, 1\}$. Define

$$B \overset{1}{\leftarrow} C = \{w \in B \mid \text{ for some } y \in C, \text{ strings } w \text{ and } y \text{ contain equal numbers of 1s}\}.$$

Show that the class of regular languages is closed under the $\overset{1}{\leftarrow}$ operation.

1.44 Let $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ and $M_C = (Q_C, \Sigma, \delta_C, q_C, F_C)$ be DFAs recognizing B and C, respectively. Construct NFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $B \overset{1}{\leftarrow} C$ as follows. To decide whether its input w is in $B \overset{1}{\leftarrow} C$, the machine M checks that $w \in B$, and in parallel nondeterministically guesses a string y that contains the same number of 1s as contained in w and checks that $y \in C$.

1. $Q = Q_B \times Q_C$.
2. For $(q, r) \in Q$ and $a \in \Sigma_\varepsilon$, define

$$\delta((q, r), a) = \begin{cases} \{(\delta_B(q, 0), r)\} & \text{if } a = 0 \\ \{(\delta_B(q, 1), \delta_C(r, 1))\} & \text{if } a = 1 \\ \{(q, \delta_C(r, 0))\} & \text{if } a = \varepsilon. \end{cases}$$

3. $q_0 = (q_B, q_C)$.
4. $F = F_B \times F_C$.

1.46 证明：

a) 假设$\{0^n1^m0^n|m,n\geq 0\}$是正则的，p 是由泵引理给出的泵长度。取 $s=0^p1^q0^p$，q>0。由泵引理条件 3 知，$|xy|\leq p$，故 y 一定由 0 组成，从而字符串 xyyz 中 1 前后 0 的数目不同，即 xyyz 不属于该语言，这与泵引理矛盾。所以该语言不是正则的。

b) 假设$\{0^n1^n|n\geq 0\}$的补集是正则的，则根据正则语言在补运算下封闭可得$\{0^n1^n|n\geq 0\}$是正则的，这与已知矛盾，故假设不成立。所以该语言不是正则的。

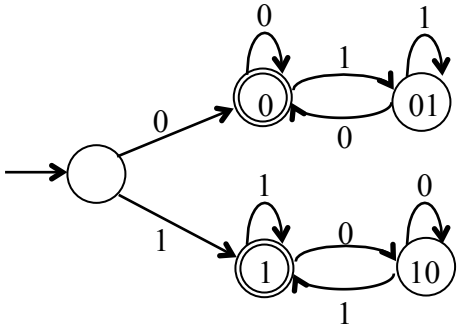
记 $c=\{0^m1^n|m\neq n\}$，¬ c 为 c 的补集，¬ $c\cap 0^*1^*=\{0^n1^n|n\geq 0\}$，已知$\{0^n1^n|n\geq 0\}$不是正则的。若 ¬ c 是正则的，由于$0^*1^*$是正则的，那么¬ $c\cap 0^*1^*$也应为正则的。这与已知矛盾，所以 ¬ c 不是正则的。由正则语言在补运算下的封闭性可知 c 也不是正则的。

c) $\{w \mid w\in\{0,1\}^*$不是一个回文$\}$的补集是$\{w \mid w\in\{0,1\}^*$是一个回文$\}$，设其是正则的，令 p 是由泵引理给出的泵长度。取字符串 $s=0^p1^q0^p$，显然 s 是一个回文且长度大于 p。由泵引理条件 3 知$|xy|\leq p$，故 y 只能由 0 组成。而 xyyz 不再是一个回文，这与泵引理矛盾。所以$\{w \mid w\in\{0,1\}^*$是一个回文$\}$不是正则的。由正则语言在补运算下的封闭性可知$\{w \mid w\in\{0,1\}^*$不是一个回文$\}$也不是正则的。

$\{wtw|w,t\in\{0,1\}^+\}$

Solution: The proof is by contradiction using the pumping lemma. Let $p>0$ (pumping length) and $w=0^p10^p\in L$. Then is is the case that $|w|=2p+1>p$. Using the condition that $|xy|\leq p$, it follows that xy (and y) consists of only 0s. Choose any i, say $i=2$, then $xyyz\notin L$ (there will be more zeros in the left side on the string). Thus, the language is not regular.

1.48 证明：构造 NFA N 如下：



由于该 NFA 识别 D，故 D 是正则语言。

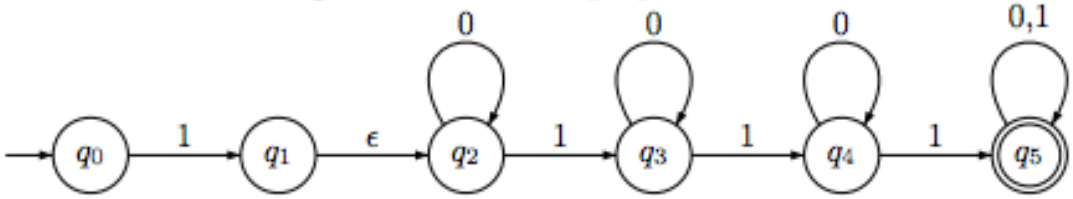
1.49 **a.** Let $B = \{1^k y \mid y \in \{0,1\}^*$ and y contains at least k 1s, for $k \geq 1\}$. Show that B is a regular language.

b. Let $C = \{1^k y \mid y \in \{0,1\}^*$ and y contains at most k 1s, for $k \geq 1\}$. Show that C isn't a regular language.

Solution:
To prove that B is regular, we can construct an automaton that accepts B. But before doing that, it may be useful to work with an equivalent definition of B that is easier to map to an automaton.

$1^k y = 11^{k-1}y$ Taking one 1 out of 1^k
Let $x = 1^{k-1}y$, then $1^k y = 1x$

Now we just need to construct the machine $1x$, which we can do easily via the closure properties of the class of regular languages, particulary the concatenation of 1 and x. To be able to do this, we need to know what x represent: $x = 1^{k-1}y$, where $y \in \{0,1\}^*$ and y contains at least k 1s, for $k > 1$. Therefore, x contains at least $k - 1 + k = 2k - 1$ ones. But by definition, $k > 1 \iff k \geq 2$, and thus, x contains at least $2(2) - 1 = 4 - 1 = 3$ ones. Now we can work with the following definition $B = \{1x | x \in \{0,1\}^*$and x contains at least 3 ones$\}$. For which we can build the following NFA, via closure properties:



(b) Let B be the set of strings over $\{0,1\}$ that can be written in the form $1^k 0y$ where y contains at least k 1s, for some $k \geq 1$. Show that B is not a regular language.

SOLUTION: Assume to the contrary that B is regular. Let p be the pumping length given by the pumping lemma. Consider the string $s = 1^p 0^p 1^p \in B$. The pumping lemma guarantees that s can be split into 3 pieces $s = abc$, where $|ab| \leq p$. Hence, $y = 1^i$ for some $i \geq 1$. Then, by the pumping lemma, $ab^2c = 1^{p+i}0^p1^p \in B$, but cannot be written in the form specified, a contradiction.

(c) Let C be the set of strings over $\{0,1\}$ that can be written in the form $1^k z$ where z contains at most k 1s, for some $k \geq 1$. Show that C is not a regular language.

SOLUTION: Assume to the contrary that C is regular. Let p be the pumping length given by the pumping lemma. Consider the string $s = 1^p 0^p 1^p \in B$. The pumping lemma guarantees that s can be split into 3 pieces $s = abz$, where $|ab| \leq p$. Hence, $b = 1^i$ for some $i \geq 1$. Then, by the pumping lemma, $ac = 1^{p-i}0^p1^p \in C$, but cannot be written in the form specified, a contradiction.

1.50 参见练习 1.24 中给出的有穷状态转换器的非形式定义。证明不存在 FST 对每一个输入 w 能够输出 w^R，其中输入和输出的字母表为{0,1}。

证明：假设存在一台 FST 对每个输入 w 能够输出 w^R。

则对于输入串 w_1 =100, w_2=101.

该 FST 可分别输出 w_1^R=001，w_2^R=101，于是对于它的起始状态和输入字符 1，会输出 1 和 0 两个不同字符，这与 FST 是确定性有穷自动机相矛盾。

所以，不存在一台 FST 对每个输入 w 能够输出 w^R。

Assume to the contrary that some FST T outputs w^R on input w. Consider the input strings 00 and 01. On input 00, T must output 00, and on input 01, T must output 10. That is impossible, because in both cases the first input bit is a 0, but in one case the first output bit is a 0 and in the other case the first output is a 1, and the first output bit depends only on the first input bit. Hence no such FST can exist.

1.51 Let x and y be strings and let L be any language. We say that x and y are ***distinguishable by L*** if some string z exists whereby exactly one of the strings xz and yz is a member of L; otherwise, for every string z, we have $xz \in L$ whenever $yz \in L$ and we say that x and y are ***indistinguishable by L***. If x and y are indistinguishable by L, we write $x \equiv_L y$. Show that $\equiv_L$ is an equivalence relation.

证明: 1) 自反性。即对任意字符串 x，$x \equiv_L x$。这是因为对于每个字符串 z 均有 xz 和 xz 同时是或不是 L 的成员。

2) 对称性。即对任意字符串 x 和 y, 若 $x \equiv_L y$，则 $y \equiv_L x$。这是因为若 $x \equiv_L y$，则对于每个字符串 z, xz 和 yz 同时是或不是 L 的成员，那么 yz 和 xz 同时是或不是 L 的成员, 于是 $y \equiv_L x$。

3) 传递性。即对任意字符串 x,y 和 z, 若 $x \equiv_L y$ 且 $y \equiv_L z$，则 $x \equiv_L z$。这是因为对任意字符串 u, 由 $x \equiv_L y$ 知 xu 和 yu 同时是或不是 L 的成员, 由 $y \equiv_L z$ 知 yu 和 zu 同时是或不是 L 的成员, 所以 xu 和 zu 同时是或不是 L 的成员, 此即 $x \equiv_L z$。

综上所述，$\equiv_L$ 是自反的，对称的，传递的，所以是一个等价的关系。

To show that $\equiv_L$ is an equivalence relation we show it is reflexive, symmetric, and transitive. It is reflexive because no string can distinguish x from itself and hence $x \equiv_L x$ for every x. It is symmetric because x is distinguishable from y whenever y is distinguishable from x. It is transitive because if $w \equiv_L x$ and $x \equiv_L y$, then for each z, $wz \in L$ iff $xz \in L$ and $xz \in L$ iff $yz \in L$, hence $wz \in L$ iff $yz \in L$, and so $w \equiv_L y$.

A*1.52 **Myhill–Nerode theorem.** Refer to Problem 1.51. Let L be a language and let X be a set of strings. Say that X is ***pairwise distinguishable by L*** if every two distinct strings in X are distinguishable by L. Define the ***index of L*** to be the maximum number of elements in any set that is pairwise distinguishable by L. The index of L may be finite or infinite.

a. Show that if L is recognized by a DFA with k states, L has index at most k.

b. Show that if the index of L is a finite number k, it is recognized by a DFA with k states.

c. Conclude that L is regular iff it has finite index. Moreover, its index is the size of the smallest DFA recognizing it.

1.52 **(a)** We prove this assertion by contradiction. Let M be a k-state DFA that recognizes L. Suppose for a contradiction that L has index greater than k. That means some set X with more than k elements is pairwise distinguishable by L. Because M has k states, the pigeonhole principle implies that X contains two distinct strings x and y, where $\delta(q_0, x) = \delta(q_0, y)$. Here $\delta(q_0, x)$ is the state that M is in after starting in the start state q_0 and reading input string x. Then, for any string $z \in \Sigma^*$, $\delta(q_0, xz) = \delta(q_0, yz)$. Therefore, either both xz and yz are in L or neither are in L. But then x and y aren't distinguishable by L, contradicting our assumption that X is pairwise distinguishable by L.

(b) Let $X = \{s_1, \ldots, s_k\}$ be pairwise distinguishable by L. We construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ with k states recognizing L. Let $Q = \{q_1, \ldots, q_k\}$, and define $\delta(q_i, a)$ to be q_j, where $s_j \equiv_L s_i a$ (the relation $\equiv_L$ is defined in Problem 1.51). Note that $s_j \equiv_L s_i a$ for some $s_j \in X$; otherwise, $X \cup s_i a$ would have $k + 1$ elements and would be pairwise distinguishable by L, which would contradict the assumption that L has index k. Let $F = \{q_i |\ s_i \in L\}$. Let the start state q_0 be the q_i such that $s_i \equiv_L \epsilon$. M is constructed so that for any state q_i, $\{s |\ \delta(q_0, s) = q_i\} = \{s |\ s \equiv_L s_i\}$. Hence M recognizes L.

(c) Suppose that L is regular and let k be the number of states in a DFA recognizing L. Then from part (a), L has index at most k. Conversely, if L has index k, then by part (b) it is recognized by a DFA with k states and thus is regular. To show that the index of L is the size of the smallest DFA accepting it, suppose that L's index is *exactly* k. Then, by part (b), there is a k-state DFA accepting L. That is the smallest such DFA because if it were any smaller, then we could show by part (a) that the index of L is less than k.

1.53 令 Σ={0，1，+，=}和

ADD={ x=y+z | x,y,z 是二进制整数，且 x 是 y 与 z 的和}

证明 ADD 不是正则的。

证明：假设 ADD 是正则的。设 P 是泵引理给出的关于 ADD 的泵长度。令 s 为 $1^P=10^{P-1}+1^{P-1}$。于是 s 能够被划分成 xyz，且满足泵引理的条件。根据条件 3，$y=1^i$, i>0. 所以 xyyz 为 $1^{P+i}=10^{P-1}+1^{P-1}\notin$ADD。故 ADD 不是正则的。

1.54 Consider the language $F = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}$.

a. Show that F is not regular.

b. Show that F acts like a regular language in the pumping lemma. In other words, give a pumping length p and demonstrate that F satisfies the three conditions of the pumping lemma for this value of p.

c. Explain why parts (a) and (b) do not contradict the pumping lemma.

(a) Show that F acts like a regular language in the pumping lemma i.e. give a pumping length p and show that F satisfies the conditions of the lemma for this p.

SOLUTION: The pumping lemma says that for any string s in the language, with length greater than the pumping length p, we can write $s = xyz$ with $|xy| \leq p$, such that xy^iz is also in the language for every $i \geq 0$.

For the given language, we can take $p = 2$. Consider any string $a^ib^jc^k$ in the language. If $i = 1$ or $i > 2$, we take $x = \epsilon$ and $y = a$. If $i = 1$, we must have $j = k$ and adding any number of a's still preserves the membership in the language. For $i > 2$, all strings obtained by pumping y as defined above, have two or more a's and hence are always in the language.

For $i = 2$, we can take $x = \epsilon$ and $y = aa$. Since the strings obtained by pumping in this case always have an even number of a's, they are all in the language. Finally, for the case $i = 0$, we take $x = \epsilon$, and $y = b$ if $j > 0$ and $y = c$ otherwise. Since strings of the form b^jc^k are always in the language, we satisfy the conditions of the pumping lemma in this case as well.

(1 point for handling each of the cases.)

(b) Show that F is not regular.

SOLUTION: We claim all strings of the form ab^i must be in distinct equivalence classes for all $i \geq 0$. This is because any two strings ab^{i_1} and ab^{i_2} can be distinguished by c^{i_1}, since $ab^{i_1}c^{i_1} \in F$, while $ab^{i_2}c^{i_1} \notin F$. Since there are infinitely many equivalence classes of the indistinguishability relation, we conclude by the Myhill-Nerode theorem that no DFA can recognize F.

(2 points for giving a set of distinguishable strings and 2 points for arguing the distinguishability.)

(c) Why is this not a contradiction?

SOLUTION: The pumping lemma only says that if a language is regular, then it must satisfy the conditions of the lemma. However, this does not necessarily mean that no non-regular language can satisfy these conditions. (2 points)

1.55 The pumping lemma says that every regular language has a pumping length p, such that every string in the language can be pumped if it has length p or more. If p is a pumping length for language A, so is any length $p' \geq p$. The ***minimum pumping length*** for A is the smallest p that is a pumping length for A. For example, if $A = 01^*$, the minimum pumping length is 2. The reason is that the string $s = 0$ is in A and has length 1 yet s cannot be pumped; but any string in A of length 2 or more contains a 1 and hence can be pumped by dividing it so that $x = 0$, $y = 1$, and z is the rest. For each of the following languages, give the minimum pumping length and justify your answer.

[A]a. 0001^*
[A]b. 0^*1^*
c. $001 \cup 0^*1^*$
[A]d. $0^*1^+0^+1^* \cup 10^*1$
e. $(01)^*$
f. ϵ
g. $1^*01^*01^*$
h. $10(11^*0)^*0$
i. 1011
j. Σ^*

1.55 **(a)** The minimum pumping length is 4. The string 000 is in the language but cannot be pumped, so 3 is not a pumping length for this language. If s has length 4 or more, it contains 1s. By dividing s into xyz, where x is 000 and y is the first 1 and z is everything afterward, we satisfy the pumping lemma's three conditions.

(b) The minimum pumping length is 1. The pumping length cannot be 0 because the string ϵ is in the language and it cannot be pumped. Every nonempty string in the language can be divided into xyz, where x, y, and z are ϵ, the first character, and the remainder, respectively. This division satisfies the three conditions.

(d) The minimum pumping length is 3. The pumping length cannot be 2 because the string 11 is in the language and it cannot be pumped. Let s be a string in the language of length at least 3. If s is generated by $0^*1^+0^+1^*$ and s begins either 0 or 11, write $s = xyz$ where $x = \epsilon$, y is the first symbol, and z is the remainder of s. If s is generated by $0^*1^+0^+1^*$ and s begins 10, write $s = xyz$ where $x = 10$, y is the next symbol, and z is the remainder of s. Breaking s up in this way shows that it can be pumped. If s is generated by 10^*1, we can write it as xyz where $x = 1$, $y = 0$, and z is the remainder of s. This division gives a way to pump s.

(b) $001 \cup 0^*1^*$

Solution: if we let $s = 0$, then $x = \epsilon$ and $y = 0$, then s can be pumped and still be in A. It follows that $p = 1$

(c) $1^*01^*01^*$

Solution: the strings in this languages may or may not have ones, but they most have exactly two 0s. To be able to pump a string it must be of the form, $s = 1010$, where $x = 10$, $y = 1$ and $z = 0$. Therefore, $p = 4$. To see why this is the case, let us try to pump a string of less length $s' = 101$. There is no choice of x, y and z for this string (or even a smaller one) that would produce a string in the language for all $i \in N$.

e. $(01)^*$最小泵长度为 2.

若|s|≥2 时, 令 x=ε, y=01, z 为其余, 则 $xy^iz \in (01)^*$.

d. 01，其最小泵长度为 3。
因为语言中只有有限个字符串时，任何一个字符串都不能被抽取。所以有限语言的泵长度为其最长字符串的长度加 1。此时没有比泵长度长的字符串，前提假所以命题真。

f. ε，其最小泵长度为 1.
理由类似于 d 中所述。

*1.57 If A is any language, let $A_{\frac{1}{2}\_}$ be the set of all first halves of strings in A so that

$$A_{\frac{1}{2}\_} = \{x \mid \text{for some } y,\ |x| = |y| \text{ and } xy \in A\}.$$

Show that if A is regular, then so is $A_{\frac{1}{2}\_}$.

Answer: Given an NFA recognizing A we construct an NFA N accepting $A_{\frac{1}{2}\_}$ taking the following actions. M keeps tracks two states in N using two “fingers”. As it reads each input symbol N uses one finger to simulate M on that symbol. As the it from an accept state on a symbol. Simultaneously, M uses the other finger to run M from an accept state on a sumblol. N accepts whenever the forward simulation and the backward simulation are in the same states, that is, whenever the two fingers are together. At those we are that N has found a string

where another string of the same length can be appended to yield a member of A, precisely the definition of $A_{\frac{1}{2}\_}$.
Formally, we assume for simplicity that the NFA M recognizing A has a single accept state as guaranteed by Exercise 1.9. Let $M = (Q, \Sigma, \delta, q_0, q_{\text{accept}})$. Construct NFA $N = (Q', \Sigma, \delta', q_0', F')$ recognizing the first halves of the strings in A as follows:

i) $Q' = Q \times Q$.
ii) For $q, r \in Q$ define $\delta'((q,r),a) = \{(u,v) \mid u \in \delta(q,a) \text{ and } r \in \delta(v,b) \text{ for some } b \in \Sigma\}$.
iii) $q_0' = (q_0, q_{\text{accept}})$.
iv) $F' = \{(q,q) \mid q \in Q\}$.

*1.58 If A is any language, let $A_{\frac{1}{3}\_\frac{1}{3}}$ be the set of all strings in A with their middle thirds removed so that

$$A_{\frac{1}{3}\_\frac{1}{3}} = \{xz \mid \text{for some } y,\ |x| = |y| = |z| \text{ and } xyz \in A\}.$$

Show that if A is regular, then $A_{\frac{1}{3}\_\frac{1}{3}}$ is not necessarily regular.

Let $A = \{0^*\#1^*\}$. Thus, $A_{\frac{1}{3}\_\frac{1}{3}} \cap \{0^*1^*\} = \{0^n1^n \mid n \geq 0\}$. Regular sets are closed under intersection, and $\{0^n1^n \mid n \geq 0\}$ is not regular, so $A_{\frac{1}{3}\_\frac{1}{3}}$ is not regular.

1.60 Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. For each $k \geq 1$, let C_k be the language consisting of all strings that contain an a exactly k places from the right-hand end. Thus $C_k = \Sigma^*\mathtt{a}\Sigma^{k-1}$. Describe an NFA with $k+1$ states that recognizes C_k in terms of both a state diagram and a formal description.

1.61 Consider the languages C_k defined in Problem 1.60. Prove that for each k, no DFA can recognize C_k with fewer than 2^k states.

给出一族语言 E_n，其中每一个 E_n 可以被一台 n 个状态的 NFA 识别，但要求对常数 $c>1$，DFA 至少有 c^n 个状态。证明你给出的语言具有这种性质。

Let $E_n = \{x \in \{0,1\}^* |$ the $(n-1)$st symbol from the right in x is 1$\}$ for $n \geq 2$. Constructing an n-state NFA for E_n is easy.
Let M be a DFA recognizing E_n. We claim that M has at least 2^{n-1} states. Let S be the set of all strings of length $n-1$. For each y in S let q_y be the state that M is in after starting at the start state and the reading y. If $y \neq z$, then $q_y \neq q_z$, otherwise M fails to recognize the E_n for the following reason. Say that y and z differ on the ith bit. Then one of the strings $y0^{i-1}$ and $z0^{i-1}$ is in E_n and the other one isn't. But M has the same behavior on both, so M fails to recognize E_n. Hence, M has at least as many states a S has strings, and so M has at least 2^{n-1} states.

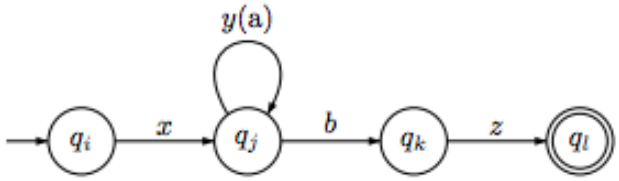
***1.63**

a. Let A be an infinite regular language. Prove that A can be split into two infinite disjoint regular subsets.

b. Let B and D be two languages. Write $B \Subset D$ if $B \subseteq D$ and D contains infinitely many strings that are not in B. Show that if B and D are two regular languages where $B \Subset D$, then we can find a regular language C where $B \Subset C \Subset D$.

Solution: By the pumping lemma, we know that any string in A of length at least p (the pumping length) can be divided into three pieces x, y and z. The following is an schematic of that situation. Please note that this diagram is intended to represent an abstract machine (as in diagram on page 79 of Sipser), and that transition from state q_i to q_j is a transition where there can be intermediate states in between, but that the machine upon consuming the piece x of the word arrives from q_i to q_j. A

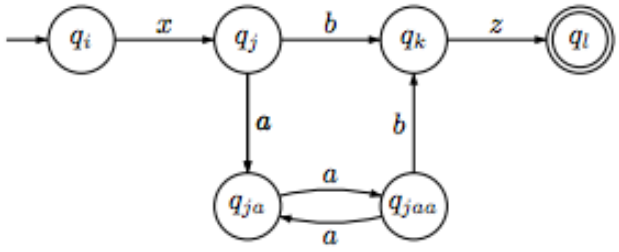
similar case occurs with the other transitions. In particular, the loop over q_j represent all of the states that the piece y goes through so that it can be pumped and still be on the language.



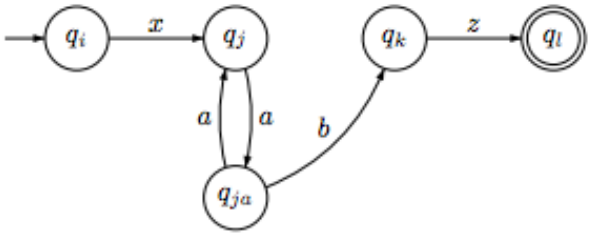
Letters a and b above represent the consumption over some finite alphabet. The alphabet can be changed and the argument will still hold. I prove this only for the case with alphabet $\Sigma = \{a, b\}$, which is the simpler. A more complicated alphabet can be reduced to this one via binary representation, which I not show here.

Now, I will modify this abstract description to yield two NFA that hold the necessary conditions.

A_1



A_2



We have that:
$L(A_1) = \{w|$ the words that are in A with an even number of a's or no a in the middle$\}$
$L(A_2) = \{w|$ the words that are in A with an odd number of a's in the middle$\}$.
By this definition, it holds that $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \emptyset$. Also, it is obvious that this regular subsets are infinite. In summary, the idea is to use the pumping lemma and rearrange the states where the string can

be pumped such that we can yield to NFAs which together represent the same language as the original NFA. Now, what happens if there are some final states between q_i and q_j? In that case, make those final states not final in one of A_1 or A_2, such that only one of them accepts them. All properties still hold. Q.E.D

1.64 Let N be an NFA with k states that recognizes some language A.

a. Show that if A is nonempty, A contains some string of length at most k.

b. Show, by giving an example, that part (a) is not necessarily true if you replace both A's by $\overline{A}$.

c. Show that if $\overline{A}$ is nonempty, $\overline{A}$ contains some string of length at most 2^k.

d. Show that the bound given in part (c) is nearly tight; that is, for each k, demonstrate an NFA recognizing a language A_k where $\overline{A_k}$ is nonempty and where $\overline{A_k}$'s shortest member strings are of length exponential in k. Come as close to the bound in (c) as you can.

Problem 1.64

Answer:

(a) If A is non-empty, there must be a path from the start state to a final state in the machine M recognizing A. This means that there must be a simple path whose length is at most k.

(b) We can construct an NFA (even over a unary alphabet) that accepts strings whose lengths are not multiples of $2, 3$ or 5. (We also make sure that this machine accepts ϵ whose length, 0, is indeed a multiple of all 3 of these numbers. The machine works as follows. The start state q_0 is accepting. From q_0 there are ϵ-transitions to q_2, q_3 and q_5, which are all non-accepting. There is a disjoint 2-cycle, 3-cycle, and 5-cycle containing q_2, q_3 and q_5 respectively. All other states in these cycles are accepting. The shortest string that is not accepted is a string of length $2 * 3 * 5 = 30$. On the other hand, the number of states in the machine is $1 + 2 + 3 + 5 = 11$. This proves the result. This construction can be generalized to more primes showing that the shortest non-accepted string for an NFA could be exponential in the number of states.

(c) The subset construction allows us to convert the NFA to a DFA with at most 2^k states. We can then complement the language accepted by the DFA by interchanging accepting and non-accepting states. Then since $\bar{A}$ has a DFA with at most 2^k, if it is non-empty, there must be a string of length at most 2^k.

(d) As we said in part (b) we can construct an NFA with $1+\sum_i p_i$ states where the shortest non-accepted string is of length $\prod_i p_i$. If we choose n primes whose values are close to n each, then the machine has approximately n^2 states, while the shortest non-accepted string is of length approximately n^n, which is the number of states raised to the power $n/2$. There may be even tighter constructions. If you think of one, please let me know!

1.66 A ***homomorphism*** is a function $f: \Sigma \longrightarrow \Gamma^*$ from one alphabet to strings over another alphabet. We can extend f to operate on strings by defining $f(w) = f(w_1)f(w_2)\cdots f(w_n)$, where $w = w_1 w_2 \cdots w_n$ and each $w_i \in \Sigma$. We further extend f to operate on languages by defining $f(A) = \{f(w) | \ w \in A\}$, for any language A.

a. Show, by giving a formal construction, that the class of regular languages is closed under homomorphism. In other words, given a DFA M that recognizes B and a homomorphism f, construct a finite automaton M' that recognizes $f(B)$. Consider the machine M' that you constructed. Is it a DFA in every case?

b. Show, by giving an example, that the class of non-regular languages is not closed under homomorphism.

- Let h(0) = ab; h(1) = ϵ.
- Let L be the language of regular expression **01*** + **10***.
- Then h(L) is the language of regular expression **abϵ*** + ϵ(**ab**)*.

Note: use parentheses to enforce the proper grouping.

- **abϵ*** + ϵ(**ab**)* can be simplified.
- ϵ* = ϵ, so **abϵ*** = **abϵ**.
- ϵ is the identity under concatenation.
  - That is, ϵE = Eϵ = E for any RE *E*.
- Thus, **abϵ*** + ϵ(**ab**)* = **abϵ** + ϵ(**ab**)* = **ab** + (**ab**)*.
- Finally, L(**ab**) is contained in L((**ab**)*), so a RE for h(L) is (**ab**)*.

True. Consider $L = \{0^n 1^n \mid n \geq 0\}$. Take $h_1(0) = a$ and $h_1(1) = \epsilon$ then $h_1(L) = L(a^*)$ which is regular. On the other hand, for $h_2(0) = 0$ and $h_2(1) = 1$, $h_2(L) = L$ which is non-regular.