1. Deterministic Finite Automata. Homeworks are p. 78: 2.1.2, 2.1.3, 2.1.7; p. 79: 2.2.1, 2.2.3, 2.3.1, 2.3.2, 2.3.3, 2.4, 2.5, 2.6; p. 81: 2.9, 2.11, 2.12, 2.13; p. 82: 2.14.1, 2.15, 2.16, 2.21, 2.22

- parity of incoming ones:

- even count of zeros after rightmost one:

Definition: A **deterministic finite automaton** (DFA) is a 5-tuple $M = (Q, \Sigma, \delta, q, F)$, where

- $Q$ is a **finite** set of **states**
- $\Sigma$ is a **finite** alphabet of **symbols**
- $\delta : Q \times \Sigma \rightarrow Q$ is a function (the **transition** function)
- $q \in Q$ is the **start** state
- $F \subseteq Q$ is a collection of **accepting** states.

- The transition function is a **program**.

- Note tabular display of the transition function.

- For a set of symbols, $\Sigma$, we let $\Sigma^*$ denote the set of finite strings composed of symbols from $\Sigma$.

- A **language** over $\Sigma$ is a subset of $\Sigma^*$.

- Definition: Let $M = (Q, \Sigma, \delta, q, F)$ be an DFA, and let $w = w_1, w_2, \ldots, w_n \in \Sigma^*$. Let $r_0, r_1, \ldots, r_n$ be a sequences of states from $Q$ such that $r_0 = q$ (from the machine definition), and $r_i = \delta(r_{i-1}, w_i)$ for $i = 1, 2, \ldots, n$.

  If $r_n \in F$, we say that $M$ **accepts** $w$.

  If $r_n \notin F$, we say that $M$ **rejects** $w$.

- Definition: Let $M = (Q, \Sigma, \delta, q, F)$ be an DFA. Then, $L(M) = \{w \in \Sigma^* : w$ is accepted by $M\}$ is the language **accepted** by $M$.

- Definition: A language $A \subseteq \Sigma^*$ is a **regular language** if there exists an DFA $M$ such that $A = L(M)$.

  Note two examples of regular languages at top: “even count of ones” and “even count of zeros after rightmost one.”

- Given DFA $M = (Q, \Sigma, \delta, q, F)$, define $\overline{\delta} : Q \times \Sigma^* \rightarrow Q$ via

  $$\overline{\delta}(r, w) = \begin{cases} r, & w = \epsilon \\ \delta(\overline{\delta}(r, v), a), & w = va, \text{ for } v \in \Sigma^* \text{ and } a \in \Sigma \end{cases}$$

- In terms of $\overline{\delta}$, $L(M) = \{w \in \Sigma^* : \overline{\delta}(q, w) \in F\}$, where $q$ is the starting state of $M$. 
• Binary string contains 101 as substring:
• Binary string that contains a one in the third position from the right. Use \(q_{000}, \ldots, q_{111}\) to track last three symbols at any given point in the scan...
• Operations on languages. For languages \(A, B \subseteq \Sigma^*\)
  – (union) \(A \cup B = \{w : w \in A \text{ or } w \in B\}\)
  – (concatenation) \(AB = \{w : w = uv \text{ for } u \in A, v \in B\}\)
  – (Kleen closure, star) \(A^* = \{w_1, w_2, \ldots, w_k : k \geq 0, w_i \in A \text{ for } 1 \leq i \leq k\}\)
• Kleene closure notation consistent with \(\Sigma^*\) notation, where we regard \(\Sigma\) as a language containing a finite number of one-character strings.
• Note Kleene closure of any language contains \(\epsilon\). So, if \(A = \phi\), then \(A^* = \{\epsilon\}\)
• Examples using \(A = \{0, 01\}, B = \{1, 10\}\)
• For any language \(A \subseteq \Sigma^*\), define
  \[A^0 = \{\epsilon\}\]
  \[A^1 = A\]
  \[A^2 = AA\]
  and so forth. In general \(A^k = AA^{k-1}\) and \(A^* = \bigcup_{k=0}^{\infty} A^k\).
• Theorem: The set of regular languages is closed under union operations. That is, if \(A, B\) are regular languages over the same alphabet, then \(A \cup B\) is a regular language.
  Proof: Note: cannot “run” on \(A\) then, having failed, run on “B.” A machine gets a single pass over the input string.
  But, can construct a Cartesian product machine...
• Regular languages closed under intersection. Use Cartesian product machine...
• Regular languages closed under complementation: Exchange accepting and non-accepting status for states in accepting DFA for \(L\). New machine accept \(L^c\).
• Note problems in proving closure under concatenation and Kleene closure...
• In a non-deterministic machine \(NFA\), a machine has several possibilities (including none) at each state
  – The machine may choose to move to one or more states on consuming a given input symbol
  – The machine may choose to move to one or more states without consuming the input symbol
  – The machine may have no choices for a particular input symbol (computation “dies” on that input)
• Binary string contain 101 or 11:
• Trace clones of NFA above for input 010110.
• Binary string with 1 in third place from right:
DFA had eight states.

• Binary string with zero count = 0 mod 2 or mod 3:

Definition: A nondeterministic finite automaton (NFA) is a 5-tuple $M = (Q, \Sigma, \delta, q, F)$ with
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite set (alphabet) of symbols
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$
  - $q \in Q$ is the start state
  - $F \subseteq Q$ is a set of accepting states.

Earlier example

Definition: Let $M = (Q, \Sigma, \delta, q, F)$ be an NFA and let $w = w_1w_2 \ldots w_m \in \Sigma^*$. We say that $M$ accepts $w$ if there exists a state sequence $r_0, r_1, \ldots, r_m \in Q$ with
  - $r_0 = q$
  - For $0 \leq i < m$, $r_{i+1} \in \delta(r_i, y_{i+1})$
  - $r_m \in F$.
Otherwise, we say $M$ rejects $w$.

Definition: $M = (Q, \Sigma, \delta, q, F)$ be an NFA. Then the language $L(M)$ accepted by $M$ is $L(M) = \{w \in \Sigma^* : M \text{ accepts } w\}$.

Definition: Let $M = (Q, \Sigma, \delta, q, F)$ be an NFA. For $q \in Q$, the $\epsilon$-closure of $q$ is $C_\epsilon(q) = \{r \in Q : r \text{ can be reached from } q \text{ by one or more } \epsilon\text{-transitions}\}$.

Example: $C_\epsilon(q_1) = \{q_1, q_2\}$.

Theorem: If $N = (Q, \Sigma, \delta, q, F)$ is an NFA, then there exists a DFA $M$ such that $L(M) = L(N)$. Also, if $M = (Q, \Sigma, \delta, q, F)$ is a DFA, then there exists an NFA $N$ such that $L(N) = L(M)$.
Proof: Note DFA to NFA is trivial: convert $\delta$ by replacing all entries in the tabulation with singleton sets and add an $\epsilon$ columns with all $\phi$ entries.

To convert an NFA to a DFA, we will construct a DFA whose state space is the power set of the NFA state space.

Intuition: The constructed DFA, after reading a prefix, say $w$, of the input will be a subset, which is precisely those states that the NFA could occupy after reading the same prefix...

Here is the construction:

Given $NFA = (Q, \Sigma, \delta, q, F)$, we construct $DFA = (Q', \Sigma', \delta', q', F')$, where

- $Q' = \mathcal{P}(Q)$
- $q' = C_\epsilon(q)$
- $F' = \{A \in Q': A \cap F \neq \phi\}$
- For $A \in \mathcal{P}(Q), \text{define } \delta'(A, a) = \cup_{r \in A} C_\epsilon(\delta(r, a))$

Trace accepting path in NFA as accepting path in DFA — ditto for non-accepting paths.

Trace accepting path in DFA as accepting path in NFA — ditto for non-accepting paths.

• Theorem: Language $A$ is regular if and only if there exists an NFA $N$ with $A = L(N)$.

![Diagram of DFA conversion](image)

• Example: Convert to DFA:

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${3}$</td>
<td>$\phi$</td>
<td>${2}$</td>
</tr>
<tr>
<td>2</td>
<td>${1}$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>3</td>
<td>${2}$</td>
<td>${2,3}$</td>
<td>$\phi$</td>
</tr>
</tbody>
</table>

Short conversion:

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ast{1,2}$</td>
<td>${1,2,3}$</td>
<td>$\phi$</td>
<td></td>
</tr>
<tr>
<td>$\ast{1,2,3}$</td>
<td>${1,2,3}$</td>
<td>$\phi$</td>
<td></td>
</tr>
<tr>
<td>$\ast{1,2}$</td>
<td>${1,2}$</td>
<td>${2,3}$</td>
<td></td>
</tr>
<tr>
<td>$\ast{2,3}$</td>
<td>${1,2}$</td>
<td>${2,3}$</td>
<td></td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td></td>
</tr>
</tbody>
</table>
Theorem: Regular languages are closed under union. That is, \( A, B \) regular implies \( A \cup B \) regular.
Proof: New start state with \( \epsilon \)-transitions to start states of \( A, B \) NFAs. Keep accepting states in both.

Theorem: Regular languages are closed under concatenation. That is, \( A, B \) regular implies \( AB \) regular.
Proof: New NFA has start state from \( A \). All accepting states from \( A \) get \( \epsilon \)-transitions to start state of \( B \). Remove accepting status from all such states in \( A \). Keep accepting status for all such states in \( B \).

Theorem: Regular languages are closed under Kleene closure.
Proof: Modify NFA \( N \) that accepts language \( L \). New starting/accepting states — to accept \( \epsilon \) string. New start gets \( \epsilon \)-transition to start state of \( M \). All accepting states of \( M \) get \( \epsilon \)-transition to start state of \( M \).

Given an alphabet \( \Sigma = \{a_1, a_2, \ldots, a_m\} \), a **regular expression** over \( \Sigma \) is a constrained string over a related alphabet containing \( \{a_1, a_2, \ldots, a_m, (, \cup, *, \epsilon, \phi\} \). Each well-formed regular expression describes a language in \( \Sigma^* \).

Examples with \( \Sigma = \{0, 1\} \).
- \( L = \{w : |w| \text{ is even}\} \) via \(((0 \cup 1)(0 \cup 1))^*\)
- \( L = \{w : |w| \text{ is odd}\} \) via \(((0 \cup 1)(0 \cup 1))^*(0 \cup 1)\)
- Finite language \( L = \{1011, 0\} \) via \( 1011 \cup 0 \)
- \( L = \{w : \text{the first and last symbols of } w \text{ are the same}\} \) via \( 0 \cup 1 \cup 0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \)

Recursive definition of a regular expression over alphabet \( \Sigma \).
Base cases: \( \epsilon, \phi, a \in \Sigma \) are regular expressions.
Inductive cases: If \( R_1, R_2 \) are regular expressions, then \( R_1 \cup R_2, R_1R_2 \) are regular expressions.
If \( R \) is a regular expression, then \( R^* \) is a regular expression.

Show how to construct \( (0 \cup 1)^*101(0 \cup 1)^* \) from the recursive definition.

Definition, assuming non-empty alphabet \( \Sigma \)
- Regular expression \( \epsilon \) represents (describes) the language \( \{\epsilon\} \)
- Regular expression \( \phi \) represents the language \( \phi \)
– Regular expression \(a\), for \(a \in \Sigma\), represents the language \(\{a\}\)
– If regular expressions \(R_1, R_2\) represent languages \(L_1, L_2\) respectively, then \(R_1 \cup R_2\) represents language \(L_1 \cup L_2\)
– If regular expressions \(R_1, R_2\) represent languages \(L_1, L_2\) respectively, then \(R_1 R_2\) represents language \(L_1 L_2\)
– If regular expression \(R\) represents language \(L\), then \(R^*\) represents \(L^*\).

• Write \(R_1 = R_2\) if \(R_1\) and \(R_2\) represent the same language.

Example \((0 \cup \epsilon)1^* = 01^* \cup 1^*\).

• List of 16 “identities” that can be used to show regular expressions are equivalent. Some examples:

  – \(R \phi = \phi R = \phi\)
  – \(R \epsilon = \epsilon R = R\)
  – \(R \cup \phi = \phi \cup R = R\)
  – \(R \cup R = R\)
  – \(R_1 \cup R_2 = R_2 \cup R_1\)
  – \(R_1 (R_2 \cup R_3) = R_1 R_2 \cup R_1 R_3\)
  – \(\phi^* = \epsilon\)
  – \(\epsilon^* = \epsilon\)

• Theorem: Every regular language expression describes a regular language.

Proof: (Induction) There are three base cases, corresponding to \(R = \epsilon, R = \phi,\) or \(R = a,\) for some \(a \in \Sigma\). Here are the corresponding DFAs.

There are three inductive steps: \(R_1, R_2 \Rightarrow R_1 \cup R_2, R_1, R_2 \Rightarrow R_1 R_2,\) and \(R \Rightarrow R^*\).

But, assuming \(R_1, R_2, R\) are regular, we can construct NFAs for \(R_1 \cup R_2, R_1 R_2,\) and \(R^*\) as follows.
• Example: construct a machine for regex \((ab \cup a)^*\):

• Want DFA to regular expression construction. Following recurrence equation will help.

Lemma: Let \(B, C \subseteq \Sigma^*\), with \(\epsilon \not\in B\), and consider the language equation \(L = BL \cup C\). Then, \(L = B^*C\).

Proof: (Side note: for numbers, \(x = 4x + 6 \Rightarrow -3x = 6 \Rightarrow x = -2\.)

Let \(u \in B^*C\). Want to show \(u \in L\).

\(u\) is the concatenation of \(k\) strings in \(B\) followed by a string in \(L\), for some \(k \geq 0\).

Use induction on \(k\).

Base case: \(k = 0 \Rightarrow u \in C \Rightarrow u \in BL \cup C = L\).

Inductive case: \(k > 0 \Rightarrow u = vwc\), with \(v \in B\), \(w\) is the concatenation of \(k-1\) strings in \(B\), and \(c \in C\).

Let \(y = wc\), which is the concatenation of \(k-1\) strings from \(B\) with a string from \(C\).

By the induction hypothesis, \(y \in L \Rightarrow u = v \in BL \subseteq BL \cup C = L\). We conclude \(B^*C \subseteq L\).

Now, let \(u \in L\). Want to show \(u \in B^*C\).

Use induction on \(|u|\).

Base case: \(|u| = 0 \Rightarrow u = \epsilon \in L = BL \cup C\)

But, \(u = \epsilon \not\in BL\) because \(\epsilon \not\in B\). Therefore \(u \in C \subseteq B^*C\).

Inductive case: \(|u| = k > 0\).

Case (a) \(u \in C \Rightarrow u \in B^*C \Rightarrow \) done.

Case (b) \(u \not\in C\)

Now, since \(u \in L = BL \cup C\), we have \(u \in BL\).

Then, \(u = bv\), for \(b \in B\), \(v \in L\).

Then \(\epsilon \not\in B \Rightarrow |b| \geq 1 \Rightarrow |v| < |u| \Rightarrow \) (induction) \(v \in B^*C \Rightarrow u = bv \in BB^*C \subseteq B^*C\)

• DFA to regular expression — via an example.

Given DFA \(M = (Q, \Sigma, \delta, q, F)\), define, for each \(r \in Q\), a language \(L_r\) via

\(L_r = \{w \in \Sigma^* : \text{the path starting at state } r \text{ and consuming } w \text{ ends in an accepting state}\}\).

Note \(L(M) = L_q\), since \(q\) is the start state of \(M\).
We proceed by overkill. That is, we obtain a regular expression for each $L_r$, which includes $L_q = L(M)$.

Example:
We write $L_0$ for $L_{q_0}$ and so forth.

\[
\begin{align*}
L_0 &= aL_0 \cup bL_2 \\
L_1 &= aL_0 \cup bL_1 \\
L_2 &= \epsilon \cup aL_1 \cup bL_0
\end{align*}
\]

Substitute for $L_2$ in equation for $L_0$:

\[
\begin{align*}
L_0 &= aL_0 \cup b(\epsilon \cup aL_1 \cup bL_0) = aL_0 \cup b \cup baL_1 \cup bbL_0 = (a \cup bb)L_0 \cup baL_1 \cup b \\
L_1 &= aL_0 \cup bL_1 = bL_1 \cup aL_0 \Rightarrow L_1 = b^*aL_0
\end{align*}
\]

Then,

\[
L_0 = (a \cup bb)L_0 \cup bab^*aL_0 \cup b = (a \cup bb \cup bab^*a)L_0 \cup b = (a \cup bb \cup bab^*a)^*b.
\]

General proof involves induction on the number of states.

- Theorem: (Pumping Lemma for Regular Languages). Let $A$ be a regular language. Then there exists an integer $p \geq 1$ (the pumping length) such that every string in $s \in A$ with $|s| \geq p$ can be written as $s = xyz$, in which the segments $x, y, z$ have the following properties:
  - $y \neq \epsilon$
  - $|xy| \leq p$
  - $xy^iz \in A$, for $i = 0, 1, 2, 3, \ldots$

Proof: Let $p$ be the number of states in a DFA that accepts $A$. Let $q_0$ be the starting state of the DFA. Consider $s \in A$ with $|s| \geq p$.

Then the accepting trajectory of $s$ involves at least $p$ transitions, which involves at least $p + 1$ states.

By the pigeonhole principle, some state, say $q_1$, is repeated within those first $p + 1$ states.
• Over $\Sigma = \{0, 1\}$, $A = \{0^n1^n : n \geq 0\}$ is not regular.  
  Proof: Suppose $A$ is regular. Let $p$ be the pumping length. Pump $0^p1^p$ for a contradiction. □

• Over $\Sigma = \{0, 1\}$, $A = \{w \in \Sigma^* :$ the number of 0s in $w$ equals the number of 1s in $w\}$.  
  Proof: Suppose $A$ is regular. Let $p$ be the pumping length. Pump $0^p1^p$ for a contradiction. □

• Over $\Sigma = \{0, 1\}$, $A = \{ww : w \in \Sigma^*\}$.  
  Proof: Suppose $A$ is regular. Let $p$ be the pumping length. Pump $s = 0^p1^p$ for a contradiction. Note that $s = 0^{2p}$ doesn’t work. If $y$ is of odd length, then $xz$ is of odd length and is therefore not in $A$. □

• Over $\Sigma = \{0, 1\}$, $A = \{0^m1^n : m > n \geq 0\}$ is not regular.  
  Proof: Suppose $A$ is regular. Let $p$ be the pumping length. Pump $s = 0^p1^{p+1}1$ to get $s = xyz$ with $xy$ containing all zeros. Then $xz \in A$ and $xz$ has at least one fewer zeros than $s$, which means the number of zeros is less than or equal to the number of ones, which places $xz \notin A$ — a contradiction. □

• Over $\Sigma = \{0, 1\}$, $A = \{1^{n^2} : n \geq 0\}$ is not regular.  
  Proof: Suppose $A$ is regular. Let $p$ be the pumping length. Pump $s = 1^{p^2}$.  
  $|s| = |xyz| = p^2$  
  $|xyyz| = |xyz| + |y| = p^2 + |y|$  
  $|xy| \leq p \Rightarrow |y| \leq p$  
  $y \neq \Rightarrow |y| \geq 1$  
  $1 \leq |y| \leq p$  
  $p^2 < p^2 + y = |xyyz| \leq p^2 + p < p^2 + 2p + 1 = (p + 1)^2$  
  The length of $xyyz$ lies strictly between to adjacent squares: $p^2$ and $(p + 1)^2$ and therefore cannot be a perfect square. We conclude that $xyyz \notin A$ — a contradiction. □

• Higman’s Theorem (Section 2.10 of the text): For arbitrary $L \in \Sigma^*$,  
  $\text{SUBSEQ}(L) = \{x \in \Sigma^* : \exists y \in L, x$ is a subsequence of $y\}$ is a regular language.  
  Proof: Skip... □