CS4124 Homework #3 Solution

2.3.1
If M is not deterministic, then interchanging its start and final states is not guaranteed to make it reject every string it formerly accepted. If there is any string w such that $(s, w) \vdash_M^* (q_1, e)$ and $(s, w) \vdash_M^* (q_2, e)$, where q_1 is final and q_2 is not, then $w \in L(M)$, but the automaton that results from carrying out this construction also accepts w, even though $w \notin \overline{L(M)}$.

2.3.3
Given two regular languages L_1 and L_2, we want to show $L_1 \cap L_2$ is a regular language. Let $M_1 = (K_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \delta_2, s_2, F_2)$ be two deterministic finite automaton such that $L(M_1) = L_1$ and $L(M_2) = L_2$. Define a new automaton $M' = (K', \Sigma', \delta', s', F')$, where

$$\begin{aligned} K' &= K_1 \times K_2 \\ \Sigma' &= \Sigma \\ s' &= (s_1, s_2) \\ F' &= F_1 \times F_2 \\ \delta'(\langle q_1, q_2 \rangle, \sigma) &= \langle \delta_1(q_1, \sigma), \delta_2(q_2, \sigma) \rangle. \end{aligned}$$

From the construction of M', it can be show that $L(M') = L_1 \cap L_2$. Thus, $L_1 \cap L_2$ is a regular language.

2.3.7
(a) $a^*b(ba^*b \cup a)^*$
(b) $((a \cup b)(a \cup b))^*$
(c) $(a \cup b)^*abaa(a \cup b)^*$
(d) $(a \cup ba^*a)(ba^*a)^*b(a \cup b)^*$

2.3.11
(a) Given a homomorphism $h : \Sigma_1 \to \Sigma_2^*$, one can also take a determintistic finite automaton $M = (K, \Sigma_1, \delta, s, F)$ accepting L and define a new automaton $M' = (K', \Sigma_2, \Delta', s', F')$ to accept $h[L]$. Let k be the maximum of $|h(\sigma)|$ for $\sigma \in \Sigma_1$. Then

$$
\begin{aligned}
K' &= K' \times \{w : w \in \Sigma_2^*, |w| \leq k\} \\
s' &= \langle s, e \rangle \\
F' &= F \times \{e\} \\
\Delta' &= \{((\langle p, e \rangle, e, \langle q, h(\sigma) \rangle)) : \delta(p, \sigma 0 = q, \sigma \in \Sigma_1\} \\
&\quad \cup \{((\langle p, \sigma w \rangle, \sigma, \langle p, w \rangle)) : \delta \in \Sigma_2\}
\end{aligned}
$$

In effect, on input x, M' nondeterministically guesses the string w for which $h(x) = w$. It computes $h(x)$ on symbol at a time, storing the results in a finite buffer, the second component of K' — this computation is the first set in the definition of Δ'. It then compares this buffer to its input, advancing only when the symbols match — the second set of transitions allowed in Δ'.

It can be proven that $L(M') = h[L(M)]$ by induction on the number of steps in a computation. One needs, however, to be careful about the nature of computations of M', which consist of a transition of the first type alternating with some number of transitions of the second type.

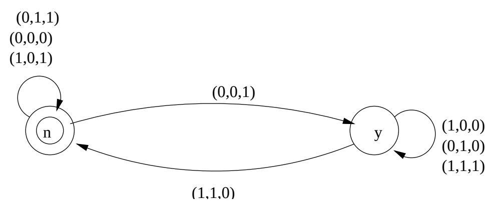
(b) Given a homomorphism $h : \Sigma_1 \rightarrow \Sigma_2^*$ and a deterministic finite automaton $M = (K, \Sigma_2, \delta, s, F)$, define a new nondeterministic finite automaton $M' = (K', \Sigma_1, \Delta', s', F')$ to accept $h^{-1}[L]$ as follows (where k is the maximum of $|h(\sigma)|, \sigma \in \Sigma_1$):

$$
\begin{aligned}
K' &= K \times \{w : w \in \Sigma_2^*, |w| \leq k\} \\
s' &= \langle s, e \rangle \\
F' &= F \times \{e\} \\
\Delta' &= \{((\langle q, e \rangle, \sigma, \langle q, h(\sigma) \rangle)) : \sigma \in \Sigma_1\} \\
&\quad \cup \{((\langle q, \sigma w \rangle, e, \langle p, w \rangle)) : \delta(q, \sigma) = p, \sigma \in \Sigma_2\}
\end{aligned}
$$

From each symbol σ of its input, M' calculates $h(\sigma)$ and "passes" the value to M by means of an internal "buffer". M then computes as normal, except that instead of reading from the input, it removes symbols from this buffer.

It can be proven that $L(M') = h^{-1}[L(M)]$ by induction on the number of steps in a computation, with a little attention to the two different sorts of transitions avaliable to M', especially as some of them are transitions on e.

2.4.2



In the figure,

$$(a, b, c) = \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Whenever the machine is in an n state, the portion of the addition to the left of the tape head requires no carrying out of the portion to the right. When the machine is in the y state, on the other hand, it expects a carry out of the portion to the right.

You maybe get a little different result if assuming the string is read by the automaton from right hand side.

2.4.5

(a) Assume L is regular, and let k be the constant whose existance the pumping theorem guarantees. Choose string $a^k bba^k$. Clearly this string is of length at least k, so the Pumping Theorem must hold. If $|xy| \leq k$, then $y = a^i$ where $i \geq 0$. But then $xy^n z = a^{k+(n-1)i} bba^k$, which is clearly asymmetric for any $n \neq 1$. The theorem fails, and thus our assumption that L was regular must have been wrong.

(b) Assume L is regular, and let k be ths constant from the pumping theorem. Choose the string $a^k ba^k b$. This string has length $2k + 2$, which is definitely at least k. If $|xy| \leq k$, the $y = a^i$ for some $i \geq 0$. thus $xy^2 z = a^{k+i} ba^k b$, which is clearly asymmetric. Thus the assumption that L is regular must be wrong.

(c) Assume L is regular, and let k be the constant given us by the pumping theorem. Choose the string $a^k b^k$. This string has length $2k$, which is definitely at least k. If $|xy| \leq k$, then $y = a^i$ for some $i \geq 0$. Thus $xy^2 z = a^{k+i} b^k$, which violates the basic condition of strings in L that there be equal numbers of a's and b's. Thus the assumption that L was regular was in error.

2.4.8

(a) false. Every language, including those we know not be regular, is a subset of the regular language Σ^*.

(b) false. The empty set, which is a regular language, has no proper subsets at all, so it certainly cannot have a proper subset which is also a regular language.

(c) true. This language is equivalent to $L\bar{L}$. Since L is regular, so is its complement, and thus their concatenation is the concatenation of two regular languages and is itself regular.

(d) false. This can be shown by trying to pump the string a^kba^k. y will have to consist only of a's, and the resulting xy^2z will be unbalanced. Note, however, that this language is regular over an alphabet of one symbol.

(e) true. This language is $L \cap L^R$. If L is regular, then so is L^R. Since both L and L^R are regular, so is their intersection.

(f) false. Any language can be written as the (possibly infinite) union of the singleton sets containing its individual elements. Since not every language is regular, the claim is false.(It is true when C is required to be finite).

(g) true. This language is Σ^*. By letting $x = e$, y can vary over all the strings of Σ^*.

2.4.12

Suppose $L \subseteq T^*$, the language of correct multiplications, were regular. Define a homomorphism $h : T^* \to \{a, b\}^*$ in which

$$h\begin{pmatrix}0\\0\\0\end{pmatrix} = h\begin{pmatrix}0\\0\\1\end{pmatrix} = a$$

and

$$h\begin{pmatrix}1\\0\\0\end{pmatrix} = h\begin{pmatrix}0\\1\\0\end{pmatrix} = h\begin{pmatrix}1\\1\\0\end{pmatrix} = h\begin{pmatrix}1\\0\\1\end{pmatrix} = h\begin{pmatrix}0\\1\\1\end{pmatrix} = h\begin{pmatrix}1\\1\\1\end{pmatrix} = b.$$

Suppose L is regular, then so is the language represent the correct mutiplication of $(2^n + 1) \times (2^n + 1)$, which we will call L'. By using the result of Problem 2.3.11, we know $h[L']$ is also regular. But $h[L'] = \{a^n w : n \leq |w|\}$, because no mutiplication of two n digit numbers can give a number more than $2n$ digits in length. And this language can easily be shown not to be regular by a straightforward application of the pumping theorem.