Solutions

4.2.6 a.

As in the text, example 4.17, we define a homomorphism \( h([paq]) = a \) for all states \( p, q \), and all input symbols \( a \). Thus, when we take the inverse homomorphism of our language, we get strings of triples. This gives us a new regular language.

We can require, in this new language, that the strings of triples represent valid computations of the DFA for our original language \( L \). That is, we require that the first state in the first triple must be the start state. Also, the transitions must follow the delta function of the DFA. Finally, the last state of the last triple must be a final state. Intersecting with valid computations is discussed in further detail in example 4.17. After doing so, we have strings of triples that represent possible valid computations of the DFA.

Next, we must make sure that for each string in the language, no proper prefix is in the language, by the definition of \( \text{min}(L) \). This requires an intersection of the following form:

Let \( E \) be the sum of all triples such that the second state is a final state. Let \( T \) be the sum of all triples

Then we can intersect our language with \( L((T - E)^*E) \), which is any number of non-final transitions followed by a final transition.

This assures us that we only have valid computations such that when we reach a final state, we have not seen a final state before.

As a final step, we apply our homomorphism again. This takes us from triples to our original input language, and we are guaranteed that the resulting strings represent \( \text{min}(L) \). Since we only used homomorphisms, inverse homomorphisms, and intersections, then we know that \( \text{min}(L) \) is a regular language.

4.2.6 a. Other version, abridged

Here also the idea is valid computations. The beginning is the same as previous.

- \( E_1 \) set of \([q_0aq]\) where \( q_0 \) is initial and \( q \) is not final and \( \delta[a, q_0] = q \)
- \( T \) set of \([qap]\) where \( \delta[a, q] = p \) and \( p \) is not final
- \( E_2 \) set of \([qaf]\) where \( \delta[a, q] = p \) and \( f \) is final
- \( E_3 \) set of \([q_0af]\) where \( q_0 \) is initial and \( f \) is final
And consider
\[ E_1T^*E_2 + E_3 \]
Of course you restrict the computations to be valid.

### 4.2.6 Other version

Let us assume our alphabet is \( \mathcal{A} = \{ a, b \} \). Let’s have a different alphabet \( \mathcal{B} = \{ a, \hat{a}, b, \hat{b} \} \). Basically we want two copies of each letter. Then define the following homomorphism \( h \) which removes the hat:
\[
\begin{align*}
    h(a) &= a & h(\hat{a}) &= a \\
    h(b) &= b & h(\hat{b}) &= a
\end{align*}
\]

We now give a look to \( h^{-1}(L) \). This is the set of strings of \( L \), with all possible combinaisons of hat and non hats. So say \( x = abba \) is in \( L \), then \( abba, \hat{a}b\hat{a}, \hat{a}b\hat{a} \ldots \) are in \( h^{-1}(L) \).

Consider now
\[
h^{-1}(L) \cap (a + b)^* (\hat{a} + \hat{b})^*
\]
This is the set of all strings of \( L \) with the first part having no hat, and the second part having a hat (\( ab\hat{a}, ab\hat{a}, \ldots \)). What if we delete the letters with a hat on them. Then we’ll get the set of all prefixes of \( L \).

So let’s define \( g \) to be the following homomorphism:
\[
\begin{align*}
    g(a) &= a & g(\hat{a}) &= \varepsilon \\
    g(b) &= b & g(\hat{b}) &= \varepsilon
\end{align*}
\]

Then
\[
\mathcal{P} = g(h^{-1}(L) \cap (a + b)^* (\hat{a} + \hat{b})^*)
\]
is the set of prefixes of \( L \). We are home. This is exacty \( \text{init}(L) \). Conclusion: \( \mathcal{P} = \text{init}(L) \) is regular.

What about \( \text{max}(L) \). We can do the same. Let us consider
\[
\mathcal{PP} = g(h^{-1}(L) \cap (a + b)^* (\hat{a} + \hat{b})^* (\hat{a} + \hat{b}))
\]
This is the set of proper prefixes of \( L \). By proper prefixes, I mean that the prefix is stricly shorter than the word itself. \( abc \) is a prefix of \( abc \) but not a proper prefix. \( ab \) is a proper prefix, as well as \( a \). I claim that \( \mathcal{PP} \) is the set of proper prefixes because I take off at least one letter with a hat.

Now the magic is:
\[
\text{max}(L) = L - \mathcal{PP}
\]
That is, \( \text{max}(L) \) is the set of all strings of \( L \) which are not a proper prefix of a string in \( L \). How do I convince you of that, I don’t know. It’s not necessarily easy to see. Say your language consists of only 3 strings: \( L = \{11, 111, 222\} \). The proper prefixes are \( \mathcal{PP} = \{1, 11, 22\} \) and
max($L$) is $\{11, 222\}$. Another way to understand $\max(L)$ is: take all words $w$ of $L$. If $w$ is a proper prefix of something else in $L$, then it shouldn’t be part of the max since there is something bigger than it. Recall that max is the set of strings $w$ of $L$ such that there are no $x \neq \varepsilon$ such that $wx \in L$. You can also say it is the set of strings in $L$ minus those which shouldn’t be there (namely the proper prefixes).

As for $\min(L)$, give a look to

$$\mathcal{M} = h^{-1}(L) \cap L(\hat{a} + \hat{b})(\hat{a} + \hat{b})^*$$

This is the set of all strings of $L$ having a proper prefix in $L$ (in plain letters) and postfix in hat. If we take $h(\mathcal{M})$ then we get the set of all strings from $L$ which have a proper prefix in $L$. $\min(L)$ is the set of strings of $L$ which do not have a proper prefix in $L$. So

$$\min(L) = L - h(\mathcal{M})$$