1 Regular Expression Solutions

1. Naively we can simply enumerate the finite language: \(aa|ab|ba|bb\). However, we can also take advantage of concatenation to get \((a|b)(a|b)\).

2. The regular expression for a term is \(a|b\), and the regular expression for an operator is \(+|−|·|/\). We want to allow one term followed by arbitrarily many operator/term pairs, yielding: \((a|b)((+|−|·|/)(a|b))^*)\).

3. Notice that an even number of 0s can occur in two ways: either there can be an even number on each side of the first 1, or an odd number on each side. First, if we allow exactly one 1 and no 2s, the regular expressions look as follows:
   - \((00)^*1(00)^*\)
   - \((00)^*010(00)^*\)

   Next, we want to allow 2s: simply insert \(2^*\) between any two symbols in the above expressions.
   - \(2^*(02^*02^*)12^*(02^*02^*)^*\)
   - \(2^*(02^*02^*)02^*12^*02^*(02^*02^*)^*\)

   Next, we want to allow more than a single 1: simply replace every \(2^*\) after the first 1 with a \((1|2)^*\) to allow arbitrary combinations of 1s and 2s.
   - \(2^*(02^*02^*)1(1|2)^*(0(1|2)^*0(1|2)^*)^*\)
   - \(2^*(02^*02^*)02^*1(1|2)^*0(1|2)^*(0(1|2)^*0(1|2)^*)^*\)

   Finally, we need to combine the two expressions. We could simply OR them together, but notice that they only differ by the 1 vs \(02^*1(1|2)^*0\) in the middle, allowing us to arrive at a slightly shorter final regular expression:

\[2^*(02^*02^*)^*(1|02^*1(1|2)^*0)(1|2)^*(0(1|2)^*0(1|2)^*)^*\]
2 NFA Solutions

• First, we add \( \{0\} \) as a starting state (step 1, 2, 3):

Next, we notice that 0 can transition on \( a \) to either
• 0 or 1, and can transition on \( b \) to 2, so we add those
states and transitions (step 4).

Next repeat for state \( \{0, 1\} \). Notice that 0 can tran-
sition on \( a \) to either 0 or 1 and 1 can transition on \( a \)
to 2, so we add a transition from \( \{0, 1\} \) to \( \{0, 1, 2\} \).
Similarly, \( \{0, 1\} \) transitions on a \( b \) to \( \{1, 2, 3\} \).

Next repeat for state \( \{2\} \). On an \( a \) we go to state \( \{1\} \),
and on a \( b \) we go to state \( \{3\} \).

Next repeat for state \( \{3\} \): the only transition is on a
\( b \) to state \( \{1\} \).

Next repeat for state \( \{1\} \): there is a transition on \( a \) to
\( \{2\} \), and a transition on \( b \) to \( \{1, 3\} \).
• Next repeat for state \{1,3\}: there is a transition on \(a\) to \{2\}, and a transition on \(b\) back to \{1,3\}.

• Next repeat for state \{0,1,2\}: there is a transition on \(a\) back to \{0,1,2\}, and a transition on \(b\) to \{1,2,3\}.

• Next repeat for state \{1,2,3\}: there is a transition on \(a\) to \{1,2\}, and a transition on \(b\) to \{1,3\}.

• Next repeat for state \{1,2\}: there is a transition on \(a\) back to \{1,2\}, and a transition on \(b\) to \{1,3\}.
• We’ve finally added transitions to every state. To finish off with step 6, label every state containing an element of $A$ (in this case, every state containing 3) as accepting.
2.1 $\epsilon$-NFA Solutions

First, we add the start state and every state which has
• a non-$\epsilon$ transition leading into it. This ends up being all states other than state 1.

Next, notice that 0 can reach itself on an $a$ and 2 on $a, b$ (taking an $\epsilon$-transition to state 1 first).

Next, notice that 2 can reach both itself and 3 on an $a$ and 0 on a $b$.

Next, notice that 3 can reach itself on an $a$ and everything on a $b$.

Finally, both 2 and 3 should be accepting states since they can both reach 2 without any input required.