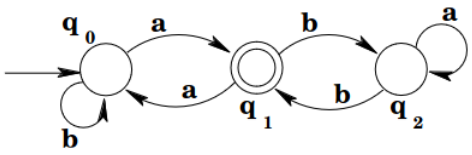
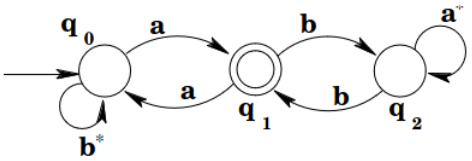
Using the strategy outlined in class, find a regular expression that generates the language accepted by the NFA below.



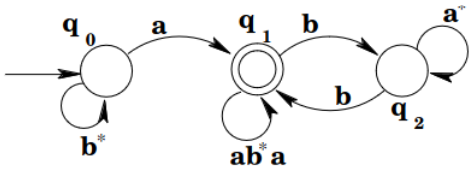

Answer: $b^*a(ba^*b|ab^*a)^*$

Discussion:

We want to replace sequences of transitions with single transitions. Cycles allow a particular sequence of transitions to be repeated any number of times (the $*$ operation in regular expressions). Start with the single-transition cycles i.e. transitions that start and end at the same state:

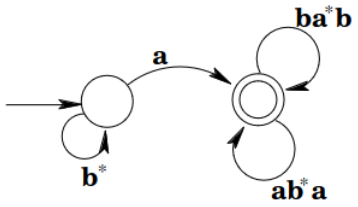



Next, look for two-transition cycles — transitions from state $q_i$ to $q_j$ and then from $q_j$ back to $q_i$. There are two of these, one involving $q_0$ and $q_1$ and one involving $q_1$ and $q_2$. Focus on cases where the middle state has only a single transition in and out (other than self loops) — a property that applies to $q_0$ and $q_2$ — so the cycles considered will be $q_1 \to q_0 \to q_1$ and $q_1 \to q_2 \to q_1$. Start with $q_1 \to q_0 \to q_1$, collapsing the cycle into a single transition labeled with the concatentation of the regular expressions along the cycle:




Note that $q_0$ and the transition $q_0 \to q_1$ are not removed because $q_0$ is the start state. (Similarly, if the middle state was a final state, you would not be able to remove it or its in-transition.) Also note that the $q_1 \to q_1$ transition would more properly be labelled $(ab^*a)^*$ to reflect the fact that loops can be travelled as many times as desired.

Now, $q_1 \to q_2 \to q_1$:



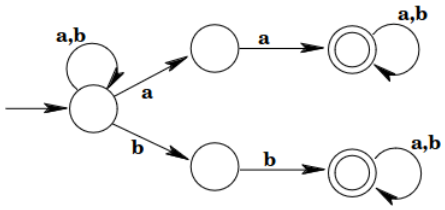

$q_2$ can be removed in this case because it is neither the start state or a final state, and there are no other ways to get to $q_2$ or leave it than from $q_1$. Again, the new $q_1 \rightarrow q_1$ transition would more properly be labelled $(ba^*b)^*$ to reflect the fact that loops can be travelled as many times as desired.

Now only the start and final states remain: the regular expression is

$$b^*a(ba^*b|ab^*a)^*$$

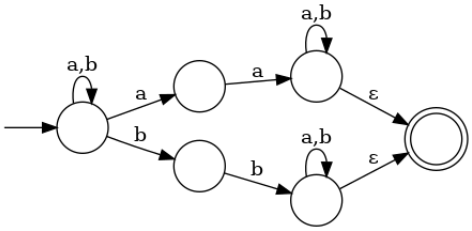
(Remember to account for the ability to go around loops as many times as desired.)

Using the strategy outlined in class, find a regular expression that generates the language accepted by the NFA below.

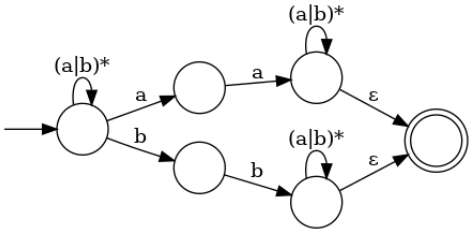



Answer: $(a|b)^*(aa(a|b)^*|bb(a|b)^*)$

Discussion: This NFA has two final states, so start by constructing an equivalent NFA with only one final state:




Now, we want to replace sequences of transitions with single transitions. Cycles allow a particular sequence of transitions to be repeated any number of times (the $*$ operation in regular expressions). Start with the single-transition cycles i.e. transitions that start and end at the same state:



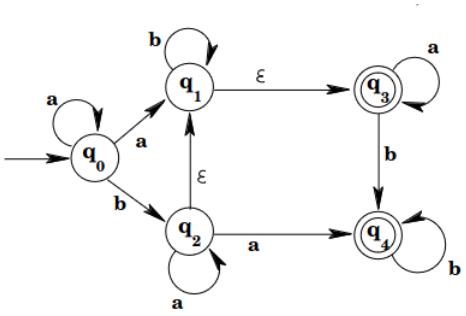

Next, look for two-transition cycles — transitions from state $q_i$ to $q_j$ and then from $q_j$ back to $q_i$. There aren't any of those here, so move on to simple paths. Look first for a sequence of two transitions where the state in the middle is not the start state or a final state and has only a single transition in and a single transition out (other than self-loops).


Now only the start and final states remain: the regular expression is

$$(a|b)^*(aa(a|b)^*|bb(a|b)^*)$$

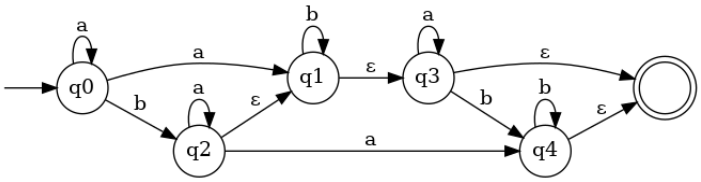
This could be simplified to $(a|b)^*(aa|bb)(a|b)^*)$ but the goal here is to use the strategy identified rather than attempt more ad hoc methods.

Using the strategy outlined in class, find a regular expression that generates the language accepted by the NFA below.

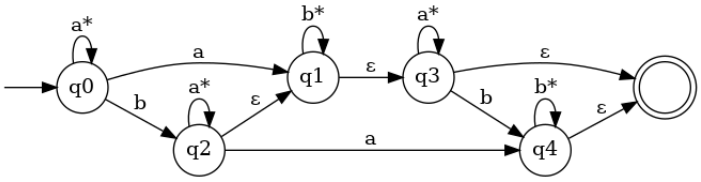



Answer: $a^*(((a|ba^*)b^*a^*b|ba^*a)b^*|(a|ba^*)b^*a^*)$

Discussion: This NFA has two final states, so start by constructing an equivalent NFA with only one final state:

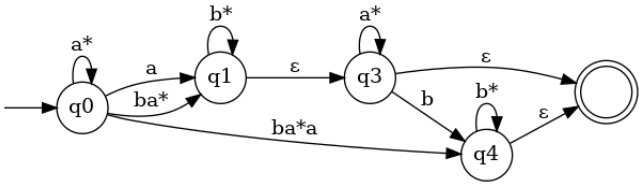



Now, we want to replace sequences of transitions with single transitions. Cycles allow a particular sequence of transitions to be repeated any number of times (the $*$ operation in regular expressions). Start with the single-transition cycles i.e. transitions that start and end at the same state:

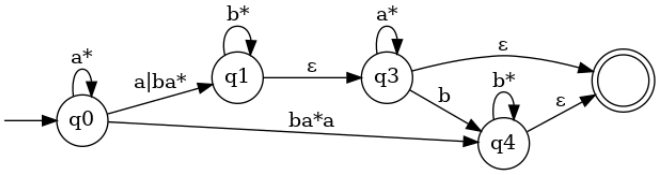



Next, look for two-transition cycles — transitions from state $q_i$ to $q_j$ and then from $q_j$ back to $q_i$. There aren't any of those here, so move on to simple paths. Look first for a sequence of two transitions where the state in the middle is not the start state or a final state and has only a single transition in and a single transition out (other than self-loops). There aren't any of those either, so look for cases where the state in the middle has only a single transition in or a single transition out (other than self-loops). This is actually the case for every state other than the start and final states, so we'll start with $q_2$ as it has a single in-transition. Since there are two ways to leave $q_2$ we'll

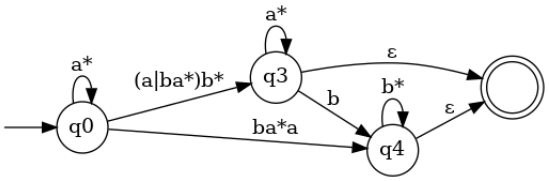
handle both $q_0 \to q_2 \to q_1$ and $q_0 \to q_2 \to q_4$ at the same time, because then all the routes through $q_2$ are covered and $q_2$ can be removed.



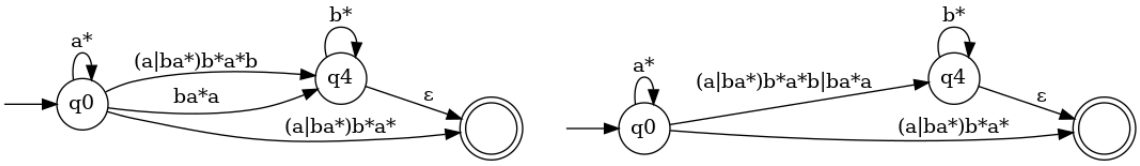

Two parallel transitions can be combined with |:



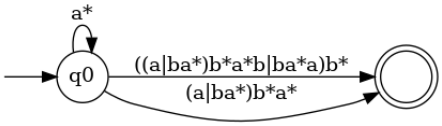

$q_0 \to q_1 \to q_3$ is now a simple path with only one transition in and out of $q_1$, so replace that next.




$q_3$ has a single in-transition, so handle that next. Also combine the two parallel transitions between $q_0$ and $q_4$.




Finally, $q_4$ has a single transition in and out.




Now only the start and final states remain: the regular expression is

$$a^*(((a|ba^*)b^*a^*b|ba^*a)b^*|(a|ba^*)b^*a^*)$$

This again is not necessarily the simplest regular expression possible, but the goal is to use the strategy identified rather than attempt more ad hoc methods.