Complete the proof of Theorem 3.3 by showing how to modify a machine that accepts $L(r)$ into a machine that accepts $L(r^*)$.

Answer: Let M be the machine that accepts $L(r)$. To accept $L(r^*)$, M' should have a new start state q_0' and an ϵ-transition from q_0' to M's start state q_0. In addition, add ϵ-transitions from each of M's final states back to q_0. Finally, designate q_0' a final state. (M's final states should remain final states too.)

Discussion: Observe that $L(r^*) = L(\epsilon|r|rr|rrr|\ldots)$.

Let M be an NFA accepting $L(r)$ and let M_* be an NFA accepting $L(r^*)$.

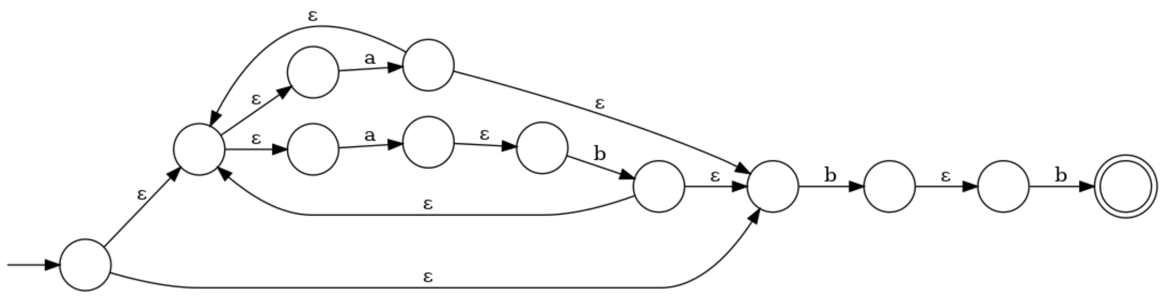
For a fixed number of copies of r, such as $L(rrr)$, we can use the construction for concatenation: connect that many copies of M in sequence, with the final state(s) of each copy connected to the start start of the next with ϵ-transitions, the start state of the whole machine being the start state of the first copy, and the final state(s) for the whole machine being the final state(s) of the last copy.

Using this idea, we can construct a machine M' accepting $L(r|rr|rrr|\ldots)$ with just a single copy of M, connecting the final state(s) of M to its start state with an ϵ-transition and leaving the start state and final state(s) as they are.

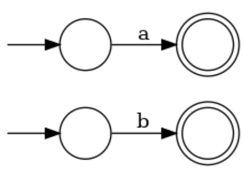
The only thing missing from $L(r^*)$ is the empty string. An NFA M_ϵ accepting $\{\epsilon\}$ consists of a single state which is both the start state and a final state. We then use the construction for $|$ to combine M_ϵ and M': create a new start state which is connected to the start states of M_ϵ and M' with ϵ-transitions. The NFA described in the answer above is a slightly simplified version of this — since nothing connects back to this new start state, it is only reachable when ϵ has been consumed, so making it final and getting rid of M_ϵ doesn't break anything.

Using the construction described in Theorem 3.3, build an NFA that accepts $L((ab|a)^*(bb))$.

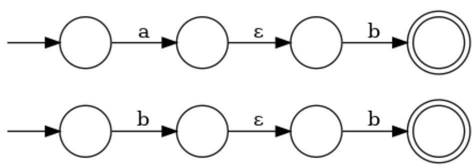
Answer:



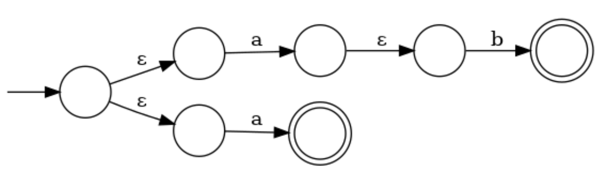
Discussion: Break $(ab|a)^*(bb)$ down into its smallest pieces (individual symbols), build the NFAs for those elements, then combine those NFAs using the construction appropriate to each operator. So, to start, NFAs accepting $L(a)$ and $L(b)$:



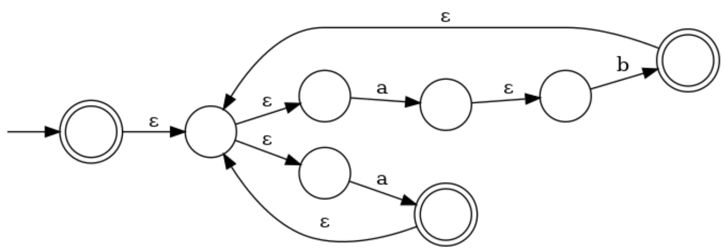
Next, NFAs for $L(ab)$ and $L(bb)$, using the construction for concatenation:



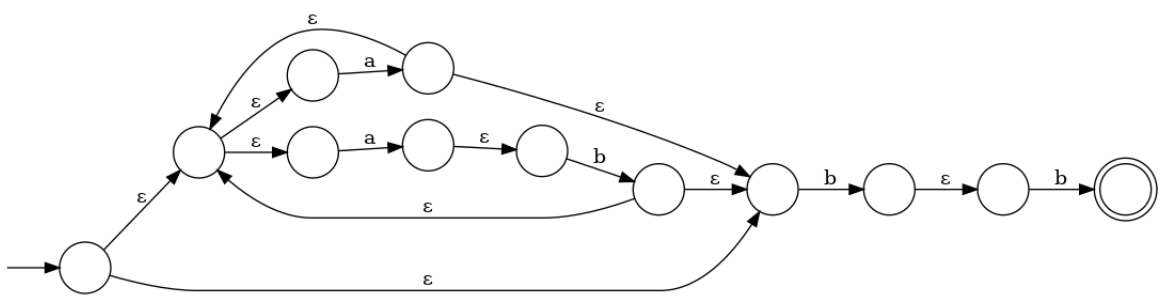
Now, for $L(ab|a)$, using the construction for $|$:



Now, for $L((ab|a)^*)$, using the construction for $*$:



And finally, the NFA for $L((ab|a)^*(bb))$, using the construction for concatenation:



This is certainly not the simplest possible NFA for $L((ab|a)^*(bb))$, but the task here was to build *an* NFA and to utilize the construction from Theorem 3.3.

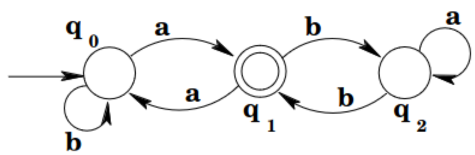
Show that for any DFA or NFA, there is an NFA with exactly one final state that accepts the same language.

Answer: Let M be the original DFA or NFA. Construct M' with the same states and transitions as M with the following modifications:

- Add a new final state q_f and ϵ-transitions from M's final states to q_f.
- The only final state in M' is q_f. (M's final states are not final in M'.)

Any string that reached a final state in M will be able to reach q_f along the ϵ-transition so M' accepts all of those strings, and q_f is only reachable from one of M's final states so no other strings will be accepted by M'.

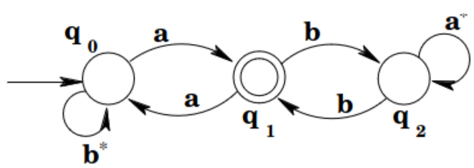
Using the strategy outlined in class, find a regular expression that generates the language accepted by the NFA below.



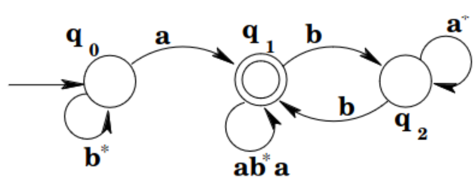
Answer: $b^*a(ba^*b|ab^*a)^*$

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:

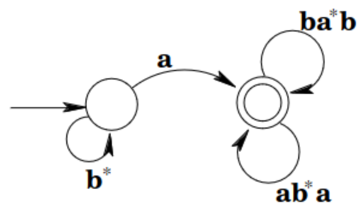


Next, look for two-transition cycles — transitions from state q_i to q_j and then from q_j back to q_i. There are two of these, one involving q_0 and q_1 and one involving q_1 and q_2. Focus on cases where the middle state has only a single transition in and out (other than self loops) — a property that 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, $q_1 \to q_2 \to q_1$:



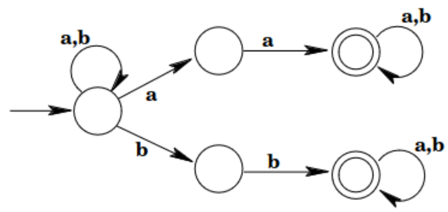
q_2 can be removed in this case because it is neither the start state or a final state, and there are no other ways to get to q_2 or leave it than from q_1. Again, the new $q_1 \rightarrow q_1$ transition would more properly be labelled $(ba^*b)^*$ to reflect the fact that loops can be travelled as many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

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

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

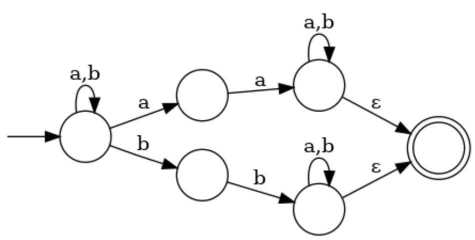
(Remember to account for the ability to go around loops as many times as desired.)

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

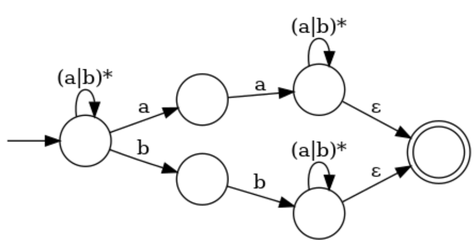


Answer: $(a|b)^*(aa(a|b)^*|bb(a|b)^*)$

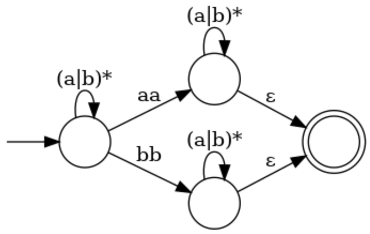
Discussion: This NFA has two final states, so start by constructing an equivalent NFA with only one final state:

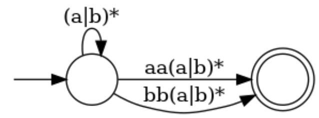


Now, we want to replace sequences of transitions with single transitions. Cycles allow a particular sequence of transitions to be repeated any number of times (the $*$ operation in regular expressions). Start with the single-transition cycles i.e. transitions that start and end at the same state:



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).



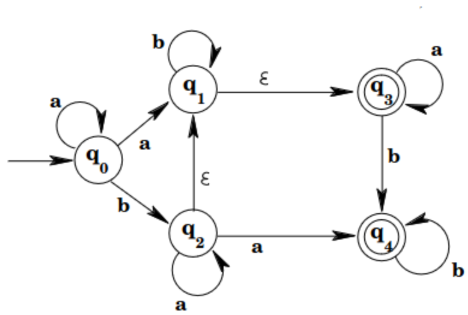


Now only the start and final states remain: the regular expression is

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

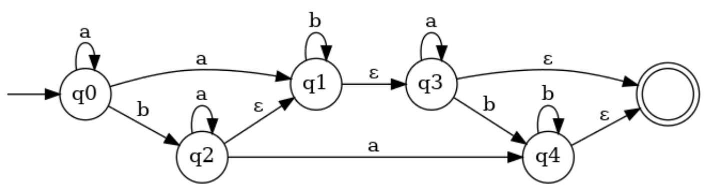
This could be simplified to $(a|b)^*(aa|bb)(a|b)^*)$ but the goal here is to use the strategy identified rather than attempt more ad hoc methods.

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

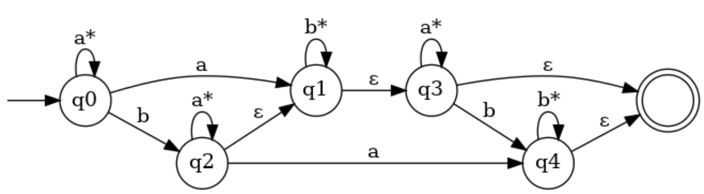


Answer: $a^*(((a|ba^*)b^*a^*b|ba^*a)b^*|(a|ba^*)b^*a^*)$

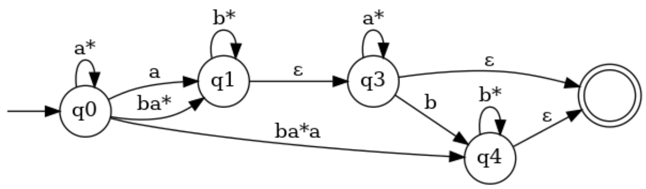
Discussion: This NFA has two final states, so start by constructing an equivalent NFA with only one final state:



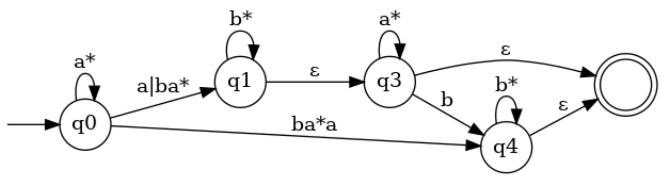
Now, we want to replace sequences of transitions with single transitions. Cycles allow a particular sequence of transitions to be repeated any number of times (the $*$ operation in regular expressions). Start with the single-transition cycles i.e. transitions that start and end at the same state:



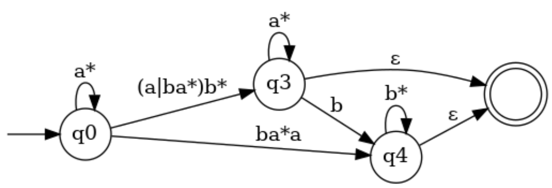
Next, look for two-transition cycles — transitions from state q_i to q_j and then from q_j back to q_i. There aren't any of those here, so move on to simple paths. Look first for a sequence of two transitions where the state in the middle is not the start state or a final state and has only a single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



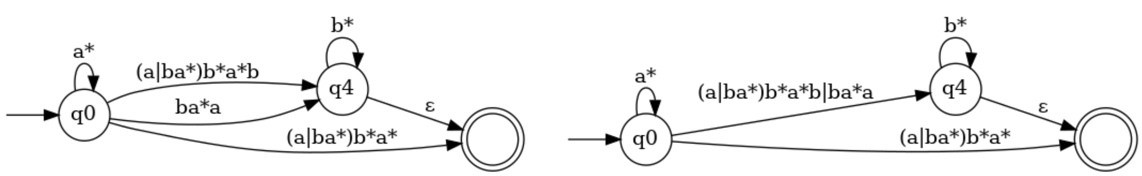
Two parallel transitions can be combined with |:



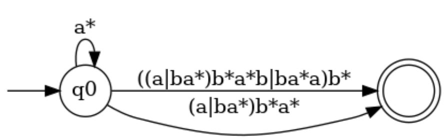
$q_0 \to q_1 \to q_3$ is now a simple path with only one transition in and out of q_1, so replace that next.



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.