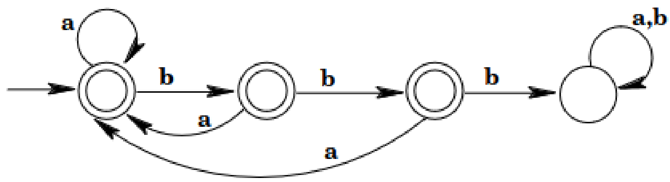
What language does the following DFA accept?



Answer: $L(M) = L((a^*|ba|bba)^*(\epsilon|b|bb)) = \{x \in \{a,b\}^* \mid x \text{ doesn't contain } bbb\}$

Discussion: For the sake of discussion, let's label the states from left to right as q_0, q_1, q_2, and q_3.

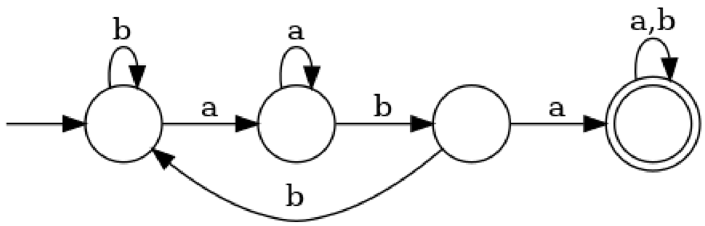
Consider the situation if just the start state q_0 is an accepting state. Observe that for any $w \in L(a^*|ba|bba)$, $\delta^*(q_0, w) = q_0$ — starting from q_0, the strings ba, bba, and any string with only as will end up back at q_0. Thus this modified DFA accepts $L((a^*|ba|bba)^*)$.

Now consider the DFA as written. From q_0, ϵ, b, and bb will get to an accepting state — this is how accepted strings can end. Putting this all together, $L(M) = L((a^*|ba|bba)^*(\epsilon|b|bb))$ — or strings which don't contain bbb.

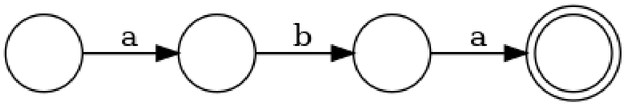
Another way of arriving at an English description of the language is to observe that there is a trap state and that bbb gets from q_0 to the trap state. Since everything else is an accepting state, strings containing bbb are the only things *not* accepted by this DFA.

Give a DFA that accepts the language $\{x \mid x \text{ contains the substring } aba\}$.

Answer:

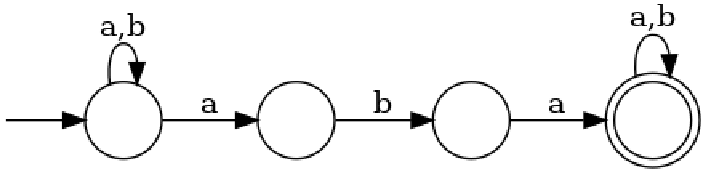


Discussion: An observation here is that there is a particular substring — aba — that the machine needs to accept. This lets us start with

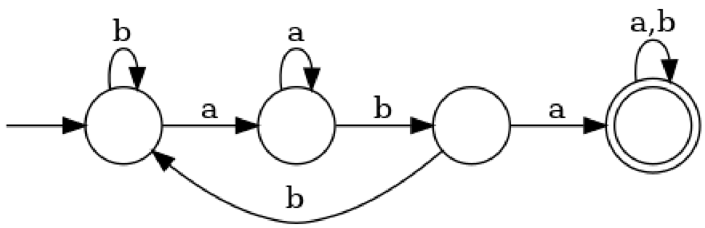


(For the sake of discussion, let's label the states from left to right as q_0, q_1, q_2, and q_3.) q_0 reflects none of aba being matched so far, q_1 means we have a, q_2 means we have ab, and q_3 means we have aba.

Next, consider what can come before and after the aba — any number of as and bs (including 0), in any order.

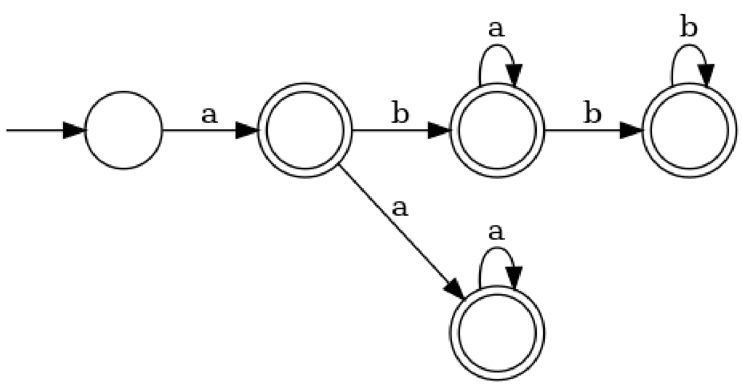


...but this isn't a valid DFA — q_0 has two transitions for a, and several transitions are missing. (The latter is OK if those transitions would lead to a trap state.) To deal with the two a transitions from q_0, consider what the a, b self-loop accomplishes: it matches any combination of as and bs occurring before the aba. Thus if we follow the a transition from q_0 to q_1 and then get another a, we should stay in q_1 — the new a is now the beginning of aba. If we get a b in q_2, however, we have to start over — the symbols immediately before this b were ab, so another b no longer matches any part of aba.

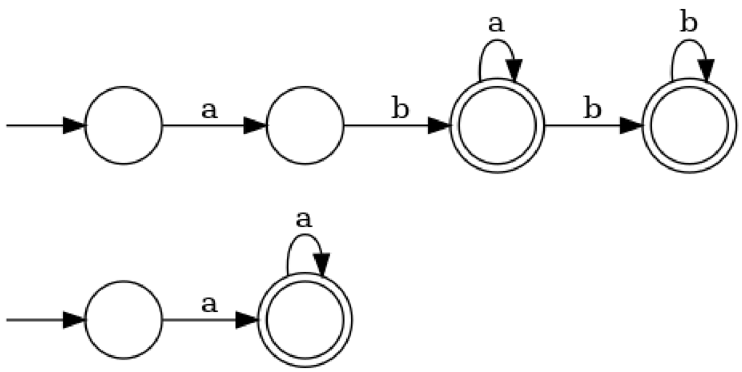


Give a DFA that accepts the language $L(aa^*|aba^*b^*)$.

Answer:

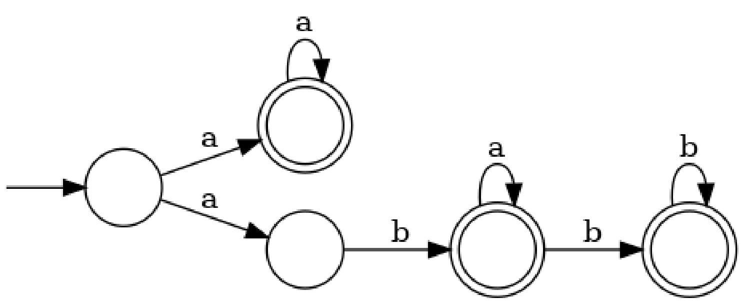


Discussion: there are two separate patterns here — aa^* and aba^*b^*. So try drawing a separate DFA for each pattern.

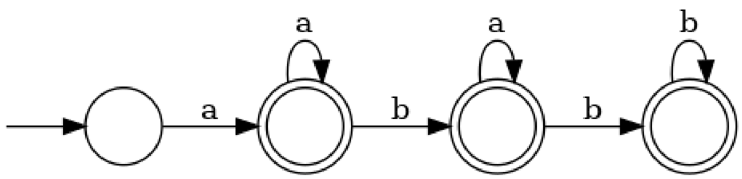


Again label the states q_0, q_1, q_2, etc from left to right. On the top, q_0 is "seen nothing", q_1 is a, q_2 is aba^*, and q_3 is aba^*bb^*. On the bottom, q_0 is "seen nothing" and q_1 is "at least one a".

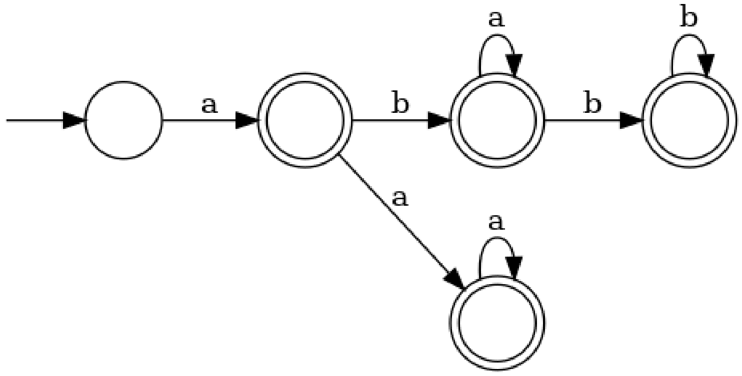
Now, merge the DFAs. Start by merging the q_0 states:



Since there are now two transitions for a from q_0, also merge the q_1 states:



This is now a valid DFA (with the omission of transitions that would lead to a trap state), but does it accept the right language? Working forwards from the start state, it accepts strings matching a, aa^*, aa^*b, ... However, aa^*b is not valid — if a string starts with aa instead of ab, it can only have as after that. So getting an a in q_1 requires a new state. (This can also be seen because the original q_1s had different meanings — a vs "at least one a".)


a
b
b
a
b
a
a