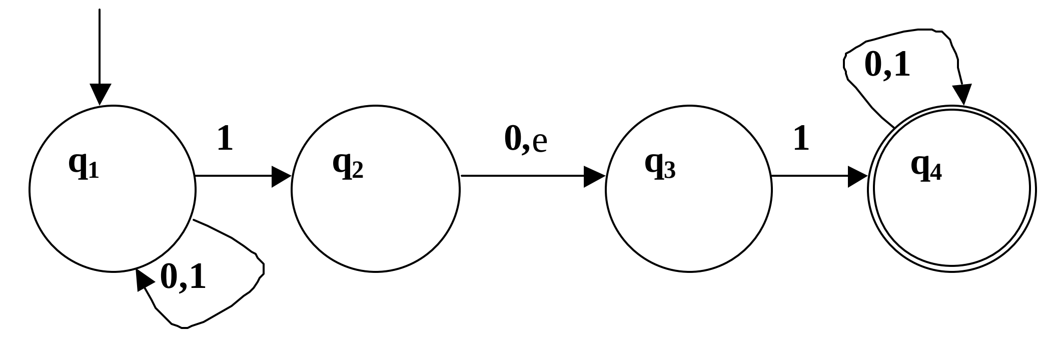
Main topics of the week:

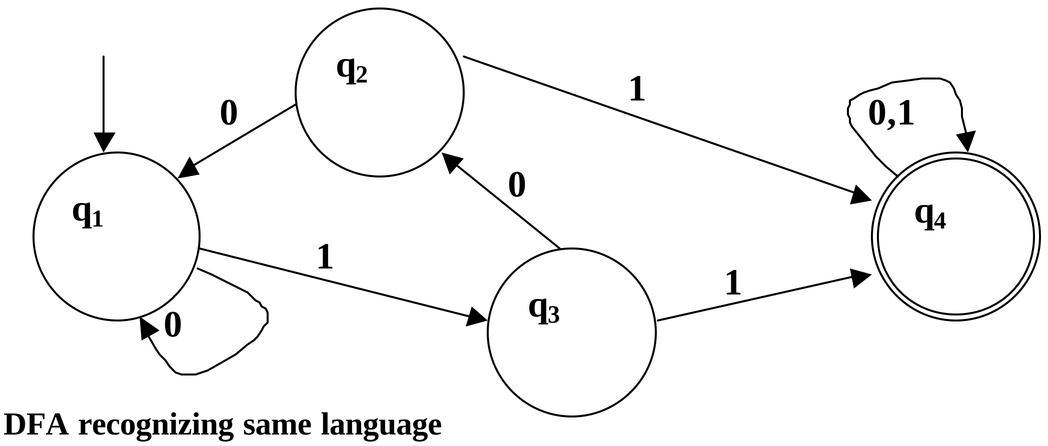
- **Examples and definition of nondeterminism**
- **Equivalence of NFAs and DFAs**
- **Conversion of NFA to DFA**
- **Closure of regular languages under union, concatenation, star**

Example of an NFA.



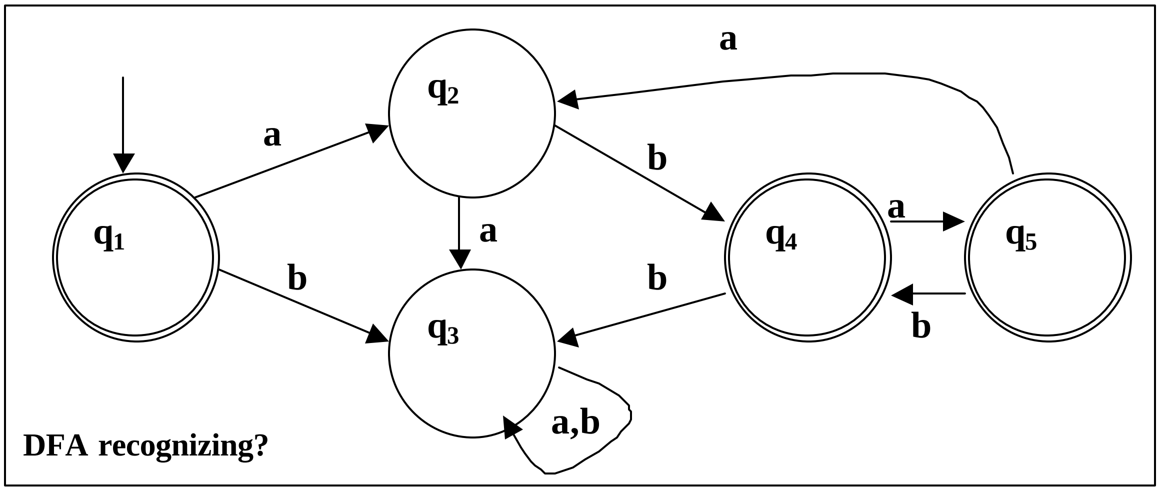
An NFA

The following diagram is a DFA that recognizes the same language. You should convince yourself that these two automata do recognize the same language. Notice that the ε transitions and multiple and missing transitions of the NFA make it much easier to understand.

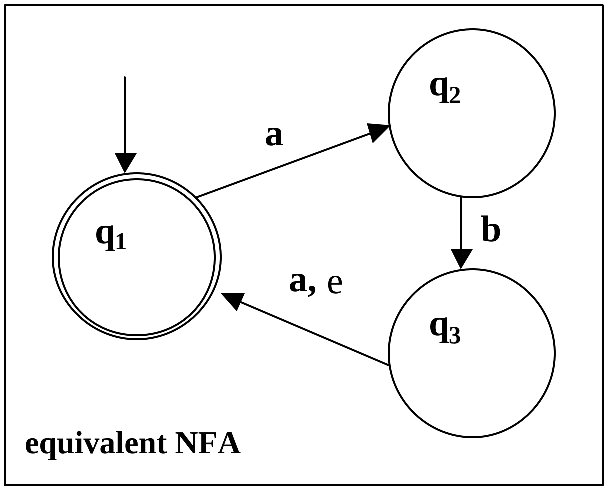


DFA recognizing same language

Another example. The following DFA has 5 states and recognizes the language that we can describe as any sequence of ab and aba. Even with just 5 states, it is not immediately obvious what the DFA does. Between states q_4 and q_5, we can see that any number of ab occurrences keeps the string accepted, but beyond that, it takes a lot of careful analysis to see what is going on. We have a dead state q_3, that is apparently reached by seeing two a's in a row.

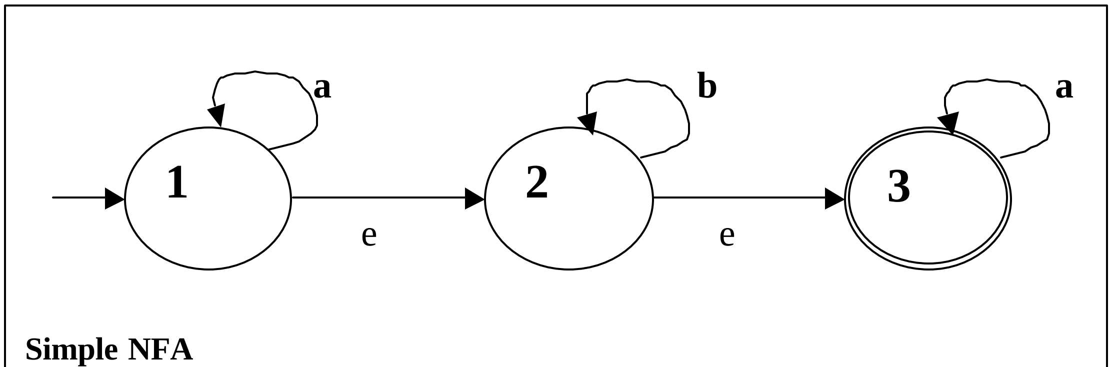


Let's contrast this with recognizing the same language with an NFA.



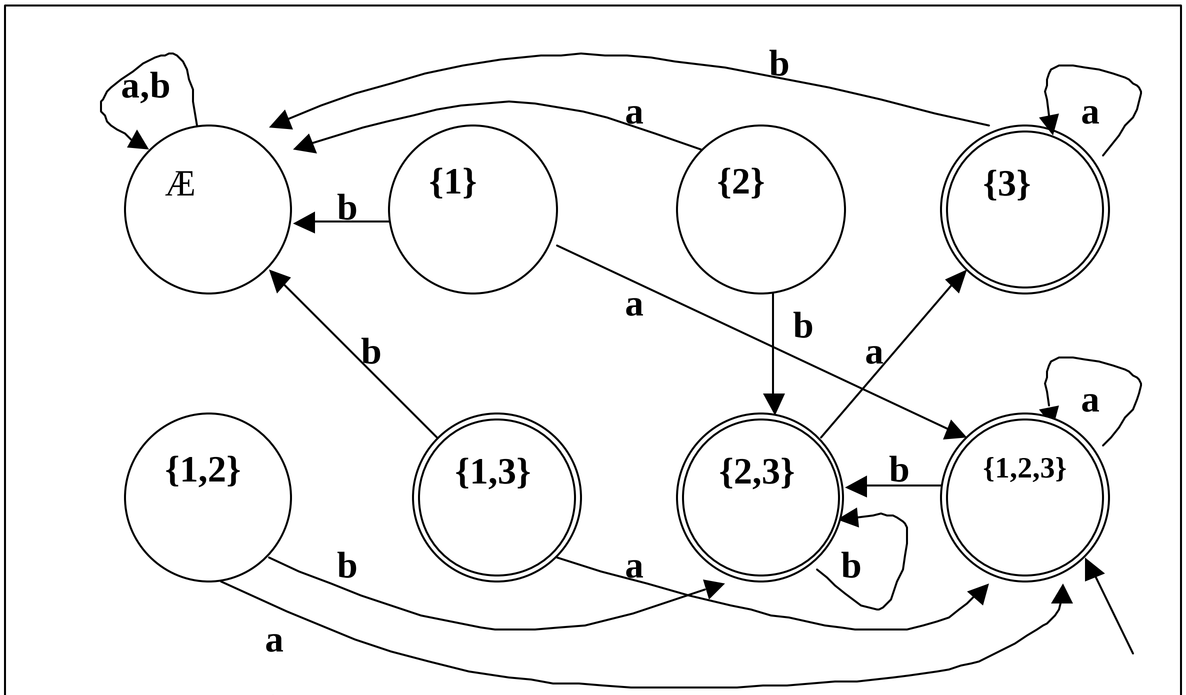
Note that here we have only three states and it is pretty easy to see what is going on, since the state progression is in only one direction, caused by a followed by **b**, and then possibly another **a** or not. This clearly recognizes strings composed of **ab** or **aba**.

Example of the construction from the proof of Theorem 1.19: Consider the simple NFA following, where the states are labeled with just the numbers 1, 2, and 3. The alphabet is {a,b} and the language recognized is all strings with zero or more a's followed by zero or more b's followed by zero or more a's.



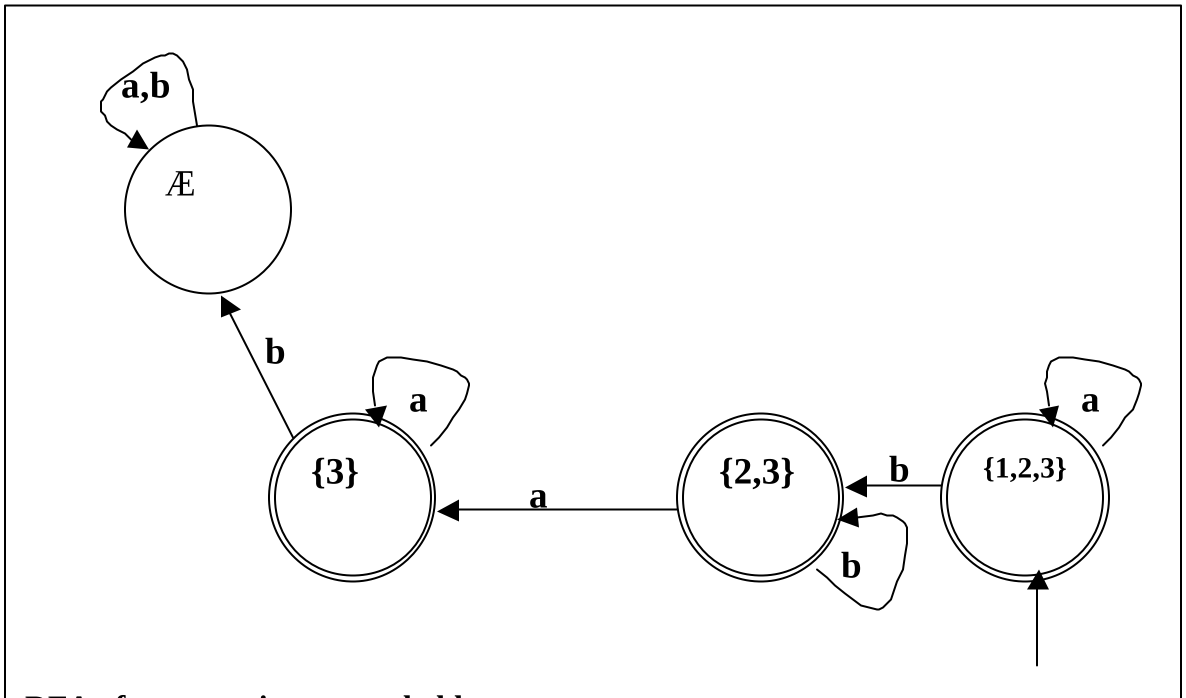
Simple NFA

Following the construction, we build our DFA using eight states, corresponding to the eight subsets of {1,2,3}.



DFA constructed from it

Notice that the start state is {1,2,3} (the ε-closure of 1), and there are four final states, the ones that contain 3. Also notice that the state ∅ has transitions back to itself since the mapping for δ' we defined in the proof results in an empty set of states. As we look at this diagram, we see that 4 states have no arrows going into them and are not the start state: {1}, {2}, {1,2}, {1,3}. So we can eliminate these and their arrows from the diagram without affecting the operation of the machine. Doing so leaves us with the diagram:



DFA after removing unreachable states

We have moved the state {3} down to make it a little less confusing. Notice that we have three final states, and roughly, {1,2,3} corresponds to leading a's, {2,3} corresponds to middle b's, and {3} corresponds to trailing a's. The ∅ state traps the occurrence of b appearing after a's, b's, and a's.

We were lucky in this example that we could eliminate half of the states. Generally, for a pattern match – looking for a substring – the constructed DFA can be simplified to the same number of states as the NFA. However, there are examples where we get the exponential increase in the number of states. The NFA recognizing sequences of 0s and 1s where the nth symbol from the end is a 1 has n+1 states (the start state transitions back to itself on a 0 or 1, but may also transition on 1 to the next state. All other transitions are for both 0 and 1. When you think about the DFA that can recognize this language, it must "remember" the last n characters it has seen to be able to tell if there is a 1 n places from the end. Since there are 2^n possible sequences of such characters, we see that there must be that many states, hence the exponential explosion in the number of states.