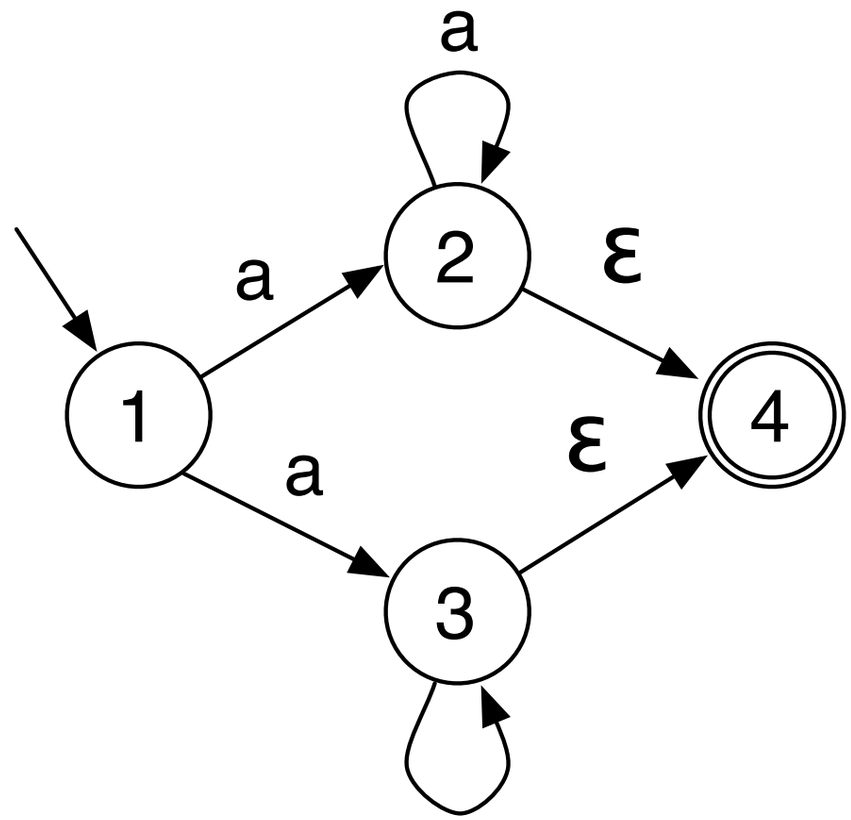
Subset Construction Revisited

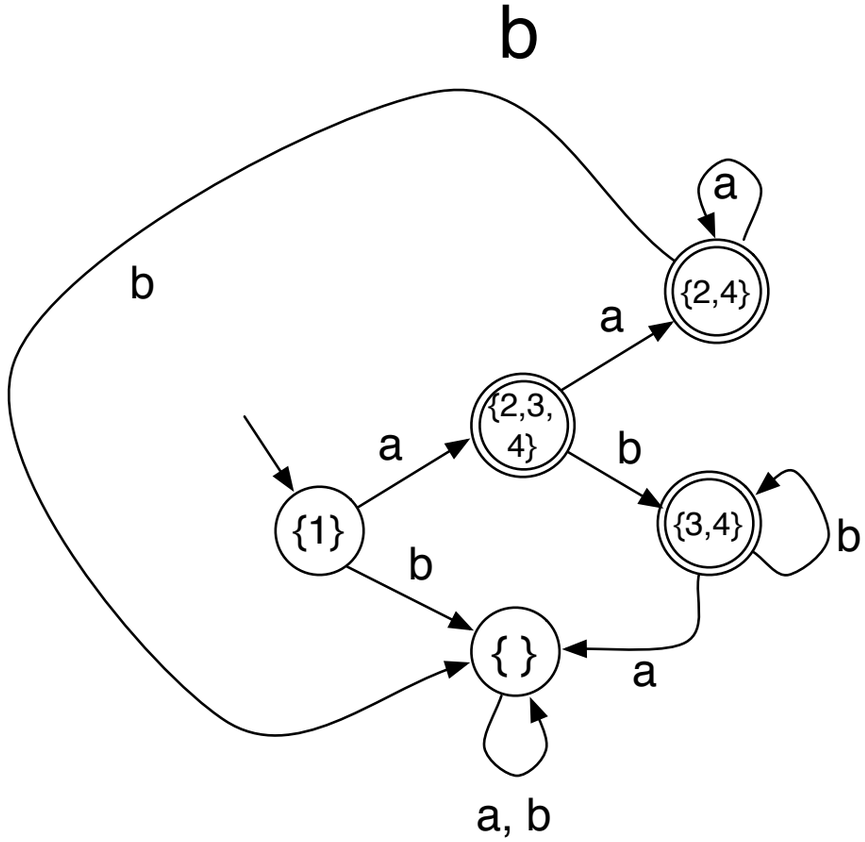
Turning an NFA into a DFA

An Important Result

- Why such emphasis on turning NFAs into DFAs?
  - because we can!
- As a consequence, ***any*** regular language can be recognized by a ***deterministic*** mechanism

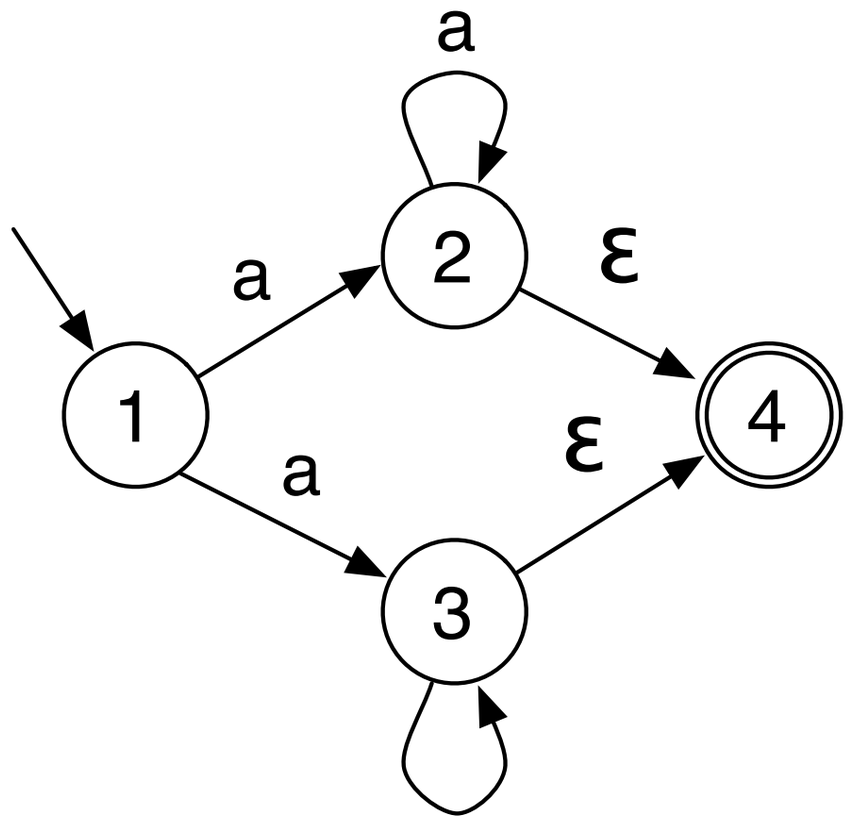
Example

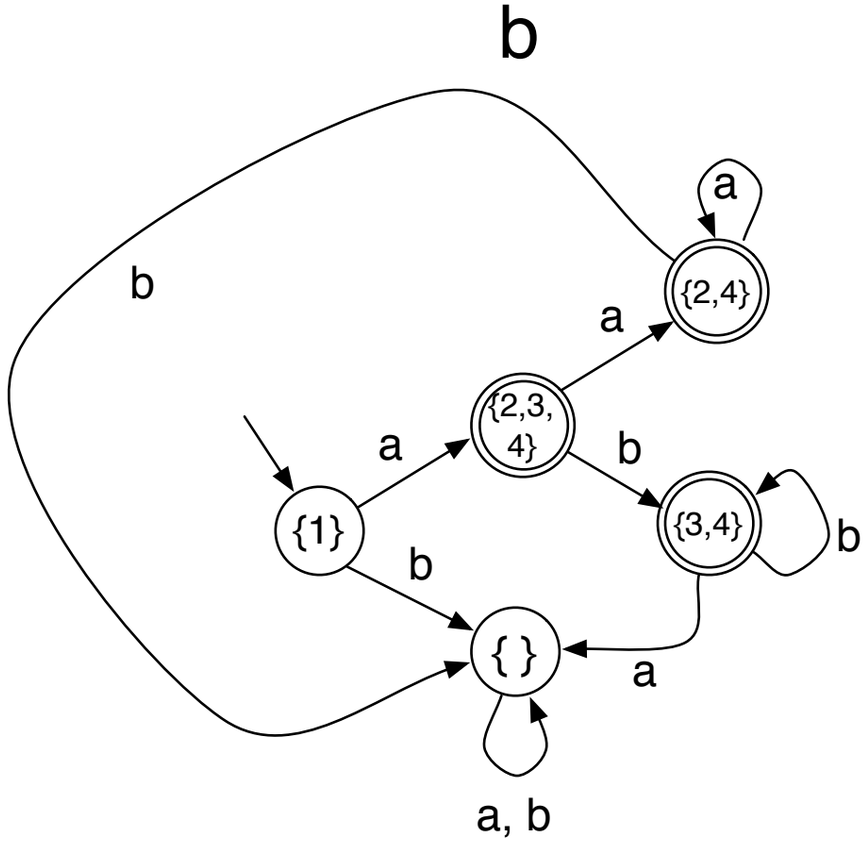




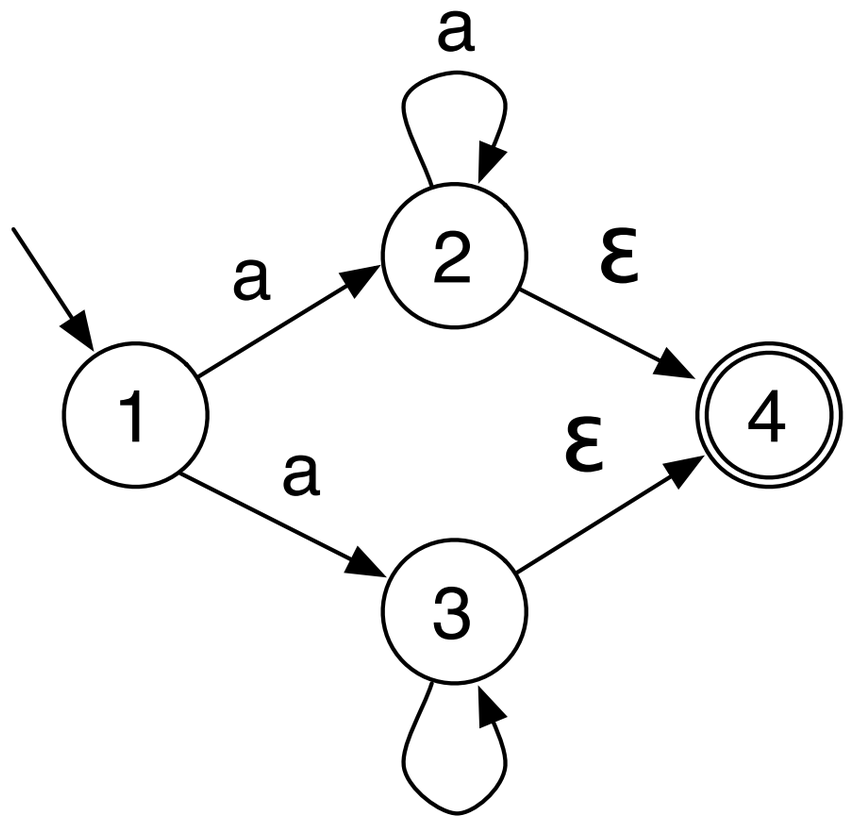
Example

1. Start: $\varepsilon\{1\} = \{1\}$

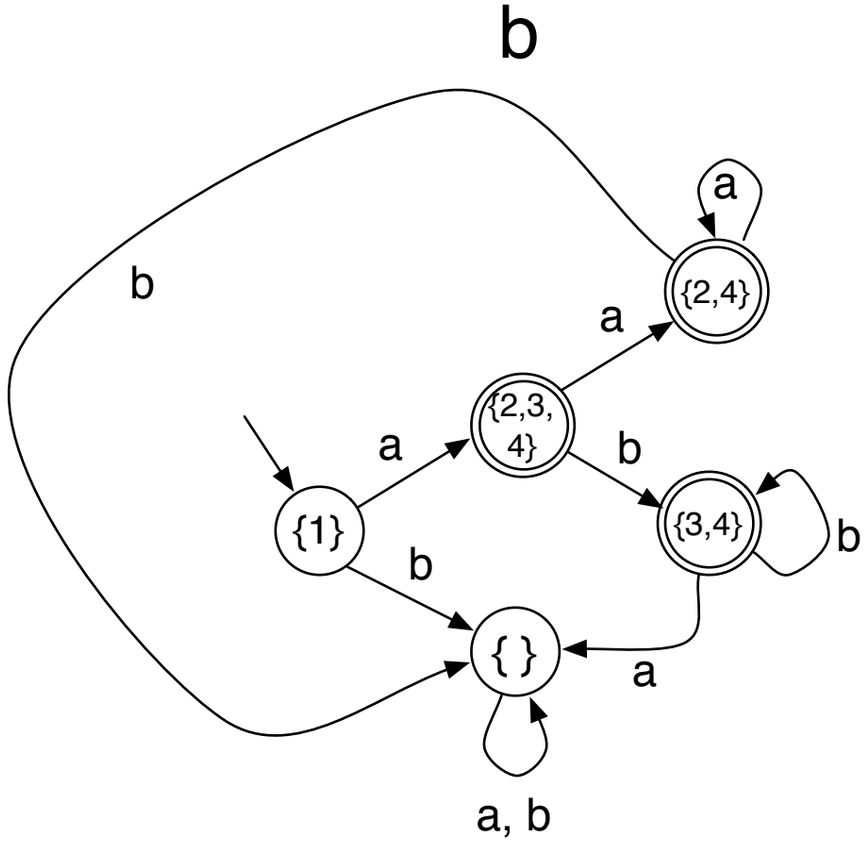




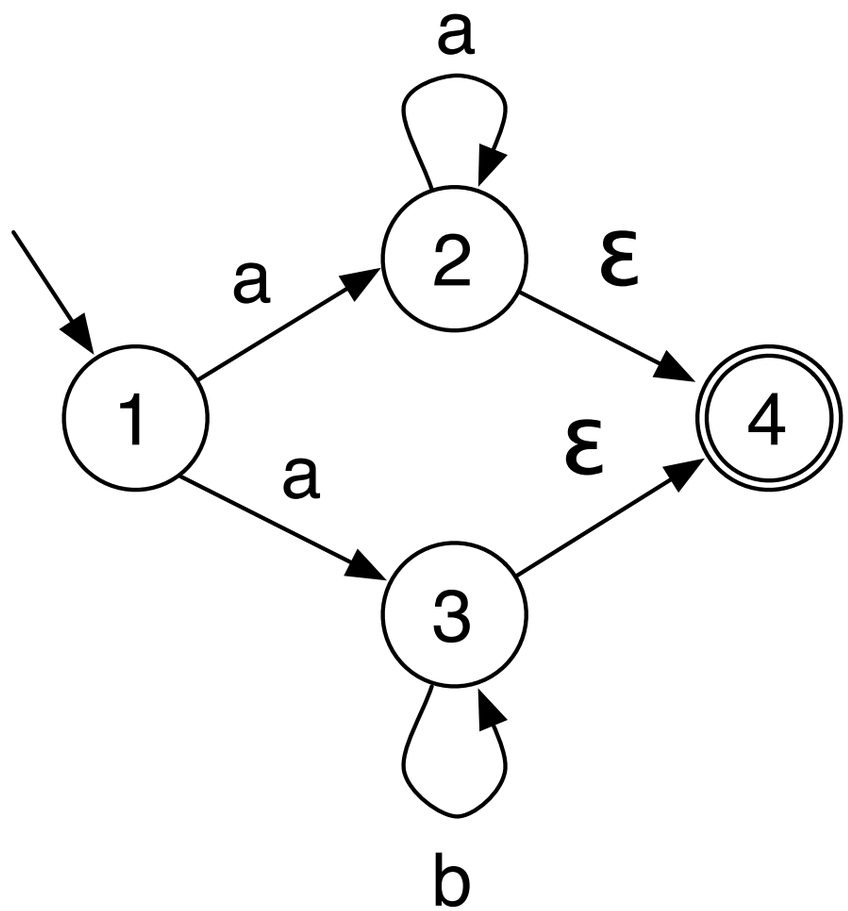
Example



1. Start: $\varepsilon\{1\} = \{1\}$
2. read a: $\varepsilon\{2, 3\} = \{2, 3, 4\}$

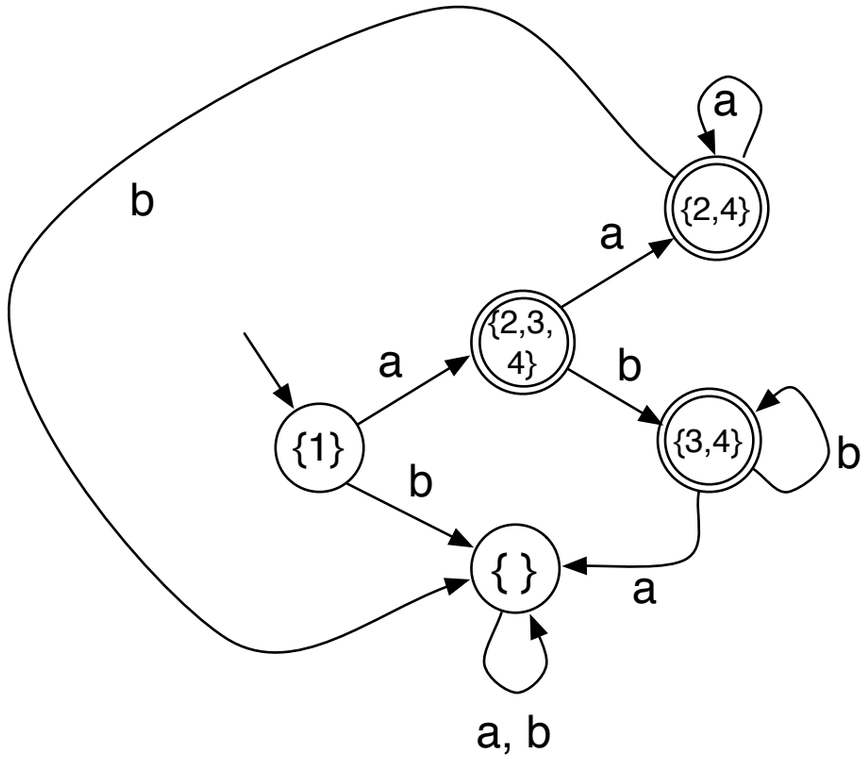


Example



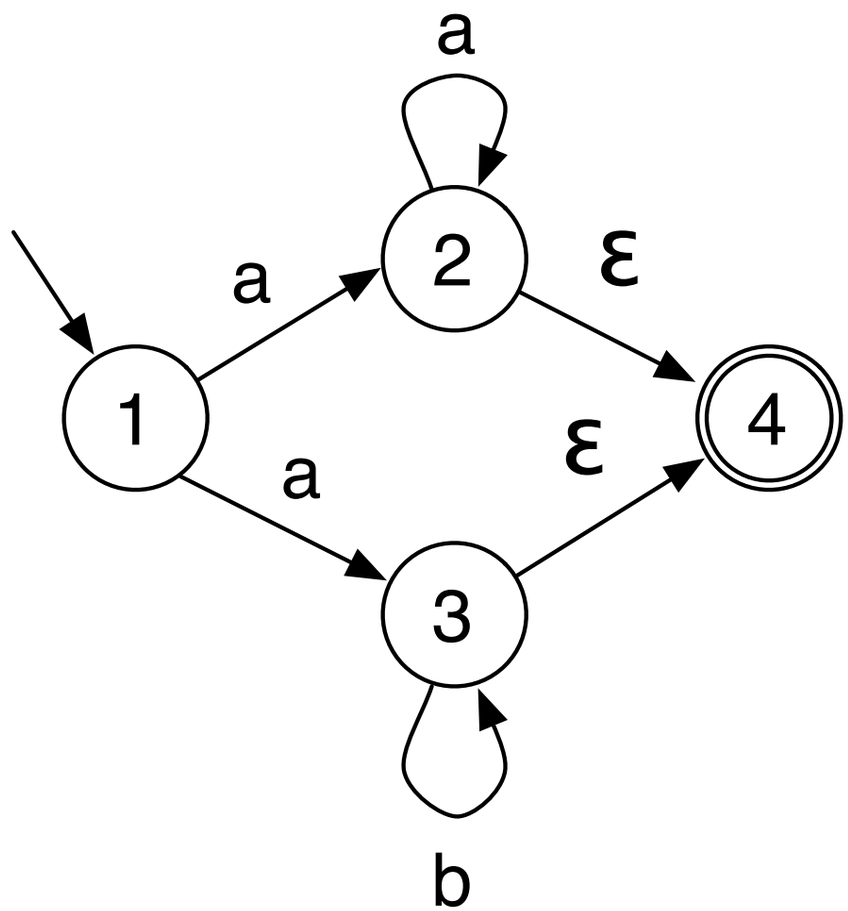
1. Start: $\varepsilon\{1\} = \{1\}$
2. read a: $\varepsilon\{2, 3\} = \{2, 3, 4\}$

$$\delta(a, 2) = \{2\}$$
$$\delta(a, 3) = \{\}$$
$$\delta(a, 4) = \{\}$$



Portland State UNIVERSITY

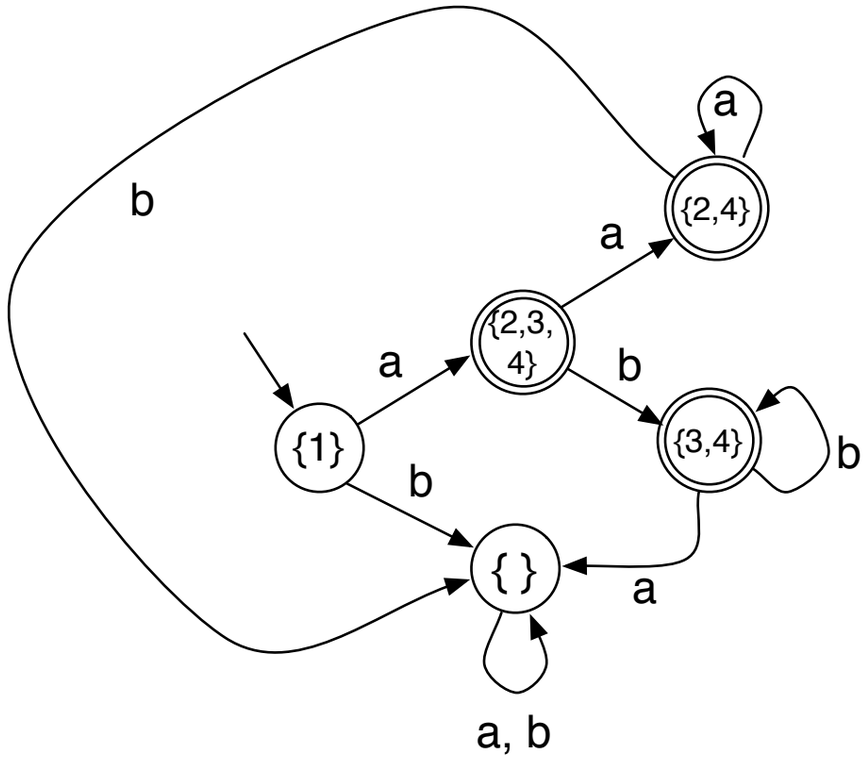
Example



1. Start: $\varepsilon\{1\} = \{1\}$
2. read a: $\varepsilon\{2, 3\} = \{2, 3, 4\}$

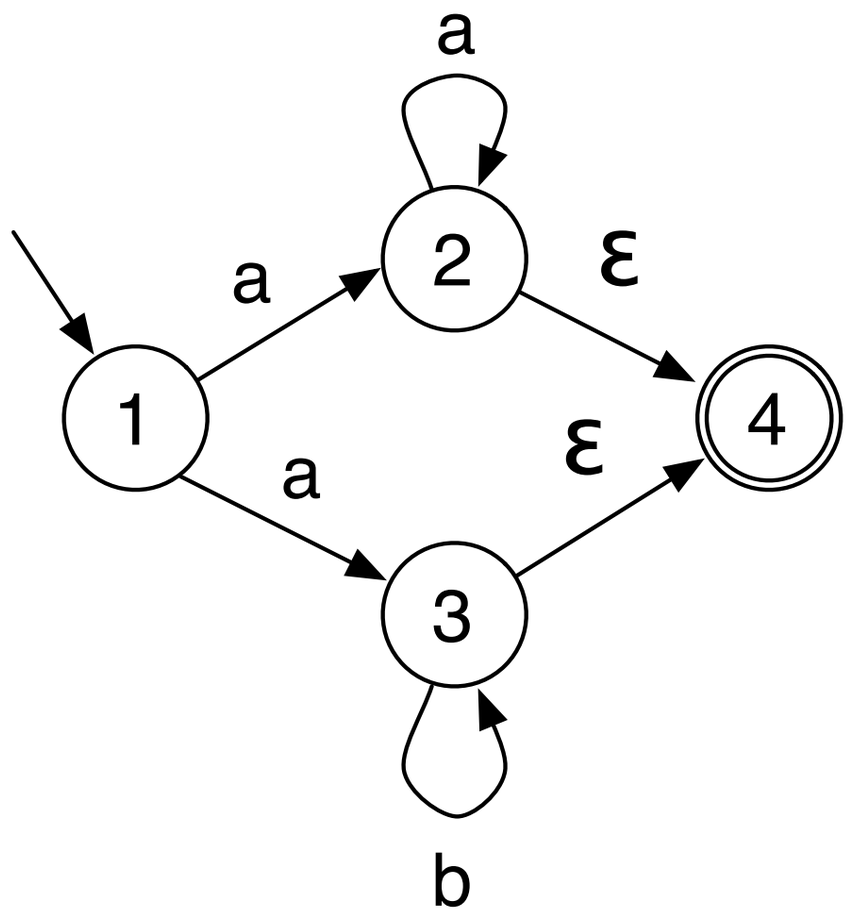
$$\delta(a, 2) = \{2\}$$
$$\delta(a, 3) = \{\}$$
$$\delta(a, 4) = \{\}$$

3. read aa: $\varepsilon\{2\} = \{2, 4\}$



Portland State UNIVERSITY

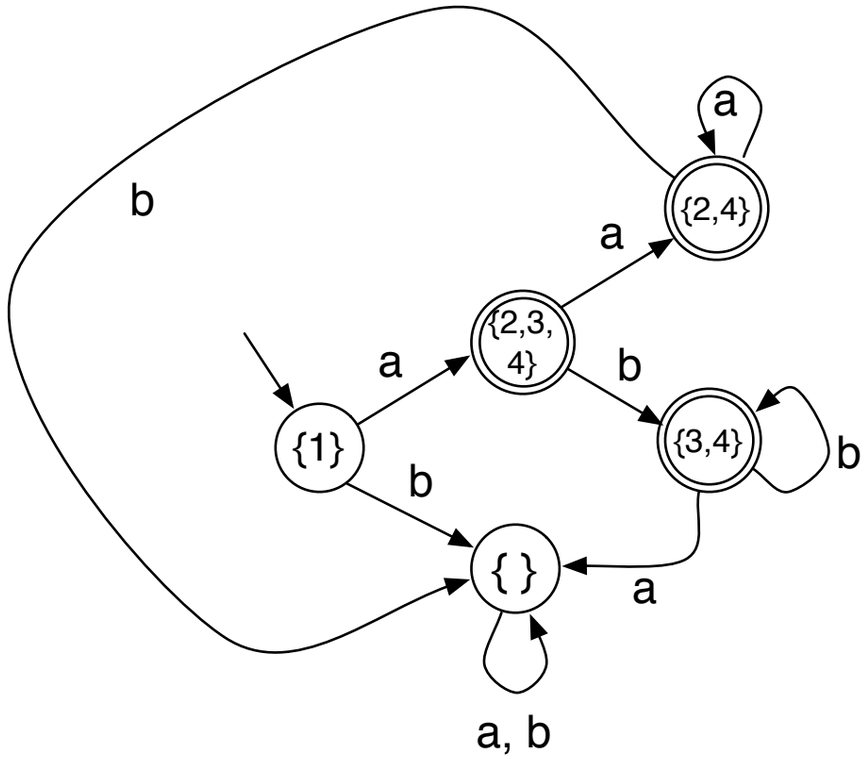
Example



1. Start: $\varepsilon\{1\} = \{1\}$
2. read a: $\varepsilon\{2, 3\} = \{2, 3, 4\}$

$$\delta(a, 2) = \{2\}$$
$$\delta(a, 3) = \{\}$$
$$\delta(a, 4) = \{\}$$

3. read aa: $\varepsilon\{2\} = \{2, 4\}$
4. read aaa: $\varepsilon\{2\} = \{2, 4\}$



Portland State UNIVERSITY

What's the DFA?

- A DFS is a 5-tuple:

1. alphabet = alphabet nfa
2. allStates = subset of $\wp$ (allStates nfa)
3. finalStates = $\{s \in$ allStates $\mid s = \{n_0, n_1, \ldots, n_k\}$ and $n_i \in$ finalStates nfa$\}$
4. startState =
5. transitions =

How to build the transition function

NFA to DFA Algorithm (11.8)

Input: An NFA over alphabet A with transition function T_N.

Output: A DFA over A with transition function T_D that accepts the same language as the NFA. The states of the DFA are represented as certain subsets of NFA states.

1. The DFA start state is $\varepsilon(s)$, where s is the NFA start state. Now perform Step 2 for this DFA start state.

2. If $\{s_1, ..., s_n\}$ is a DFA state and $a \in A$, then construct the following DFA state as a DFA table entry in either of two ways:

$$T_D(\{s_1, ..., s_n\}, a) = \varepsilon(T_N(s_1, a) \cup \cdots \cup T_N(s_n, a)) \quad \text{(closure of union)}$$

$$= \varepsilon(T_N(s_1, a)) \cup \cdots \cup \varepsilon(T_N(s_n, a)) \quad \text{(union of closure).}$$

   Repeat Step 2 for each new DFA state constructed in this way.

3. A DFA state is final if one of its elements is an NFA final state.

States are not Sets of States

- In the book, a single state in the DFA is a circle containing a label like $\{n_0, n_1, \ldots, n_k\}$
- The text goes freely from the single DFAstate $\{n_0, n_1, \ldots, n_k\}$ to the individual NFAstates $n_0, n_1, \ldots, n_k$
- When implementing the algorithm, we have to be explicit about this:
  - For example, use functions
    - DFAState: NFAstate list → DFAstate
    - NFAState: DFAstate → NFAstate list

In ML

```
> val DFAState = fn : State/1 list -> State/1
  val NFAState = fn : State/1 -> State/1 list
val n0 = newState();                    > val n0 = State "q157" : State/1
val n1 = newState();                    > val n1 = State "q158" : State/1
val n2 = newState();
val d0 = DFAState [];
val d1 = DFAState [n0, n1];
val d2 = DFAState [n0, n2];
val d3 = DFAState [n1, n2];
val d4 = DFAState [n1, n0, n2];         > val d4 = State "q164" : State/1
```

```
val testDFAState1 = assert (DFAState[] = d0);
val testDFAState2 = assert (DFAState[n0, n1] = d1);
val testDFAState3 = assert (DFAState[n1, n0] = d1);
val testDFAState4 = assert (DFAState[n2, n1] = d3);
val testDFAState5 = assert (DFAState[n2, n0] = d2);
val testDFAState6 = assert (DFAState[n0, n1, n2] = d4);
val testDFAState7 = assert (DFAState[n0, n1, n1, n2] = d4);

val testNFAState1 = assert (NFAState d0 = []);
val testNFAState2 = assert (NFAState d1 = [n0, n1]);
val testNFAState3 = assert (NFAState d3 = [n1, n2]);
val testNFAState4 = assert (NFAState d2 = [n0, n2]);
val testNFAState5 = assert (NFAState d4 = [n0, n1, n2]);
```

In conventional notation

- DFAState $\{n_0, n_1, \ldots, n_k\}$ = the single state $\{n_0, n_1, \ldots, n_k\}_{\mathrm{d}}$
- NFAState $\{n_0, n_1, \ldots, n_k\}_{\mathrm{d}}$ = the set $\{n_0, n_1, \ldots, n_k\}$