# Lecture 5

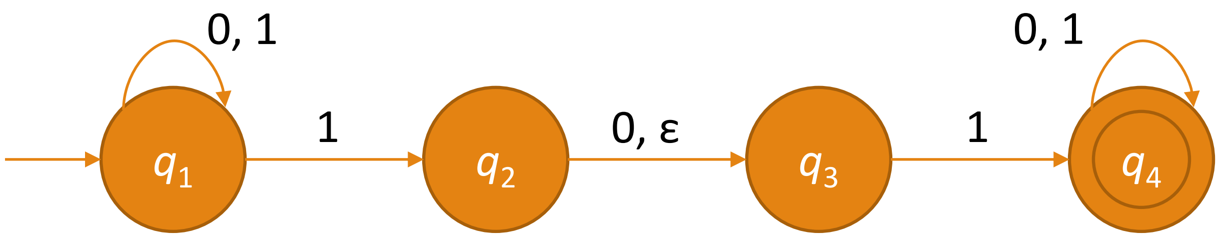
COT4210 DISCRETE STRUCTURES

DR. MATTHEW B. GERBER

6/2/2016

PORTIONS FROM SIPSER, *INTRODUCTION TO THE THEORY OF COMPUTATION*, 3RD ED., 2013

# **Revisited:** NFA-to-DFA Conversion

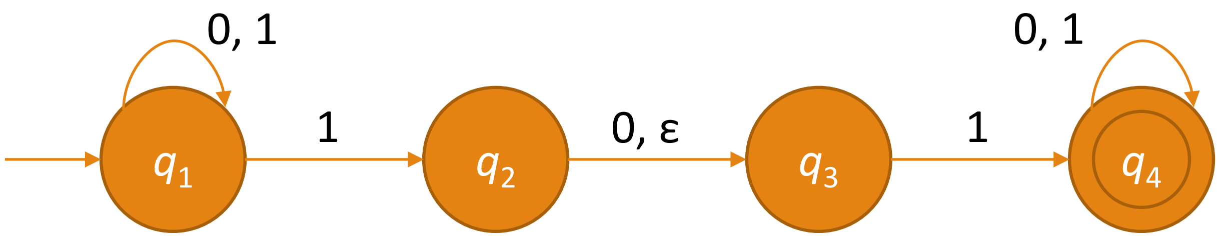



NFAs have three capabilities that DFAs don't:

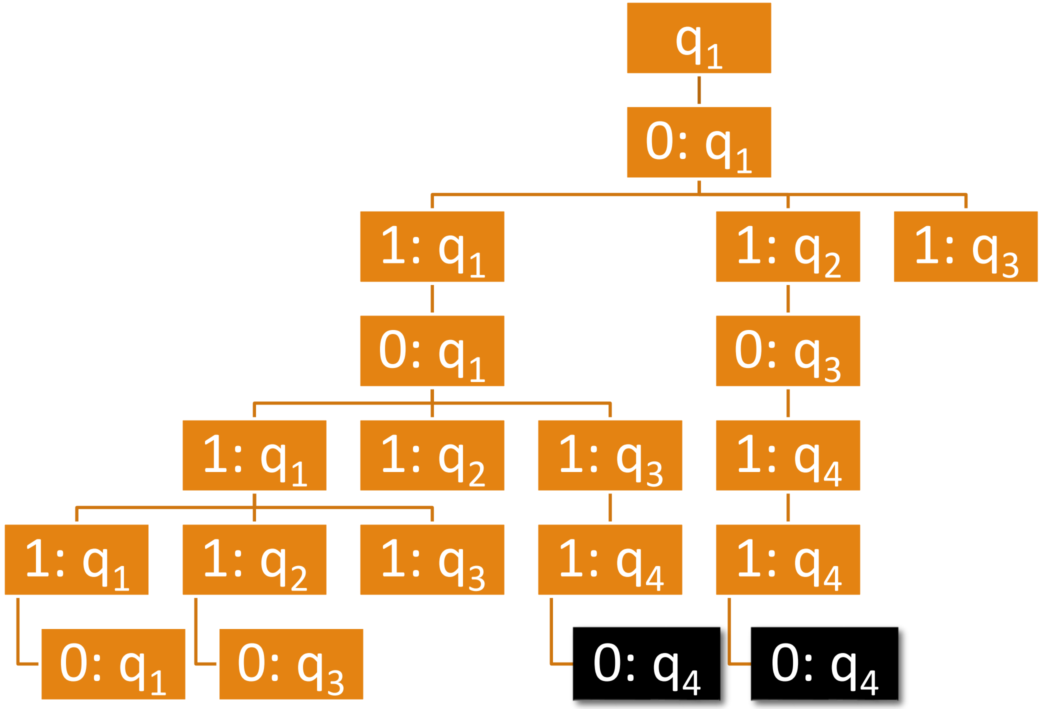
- Multiple transitions on the same symbol
- Empty-string transitions
- No transitions on some symbols

# **Revisited:** NFA-to-DFA Conversion: Multiple Transitions

Recall our original NFA



# **Revisited:** NFA-to-DFA Conversion: Multiple Transitions

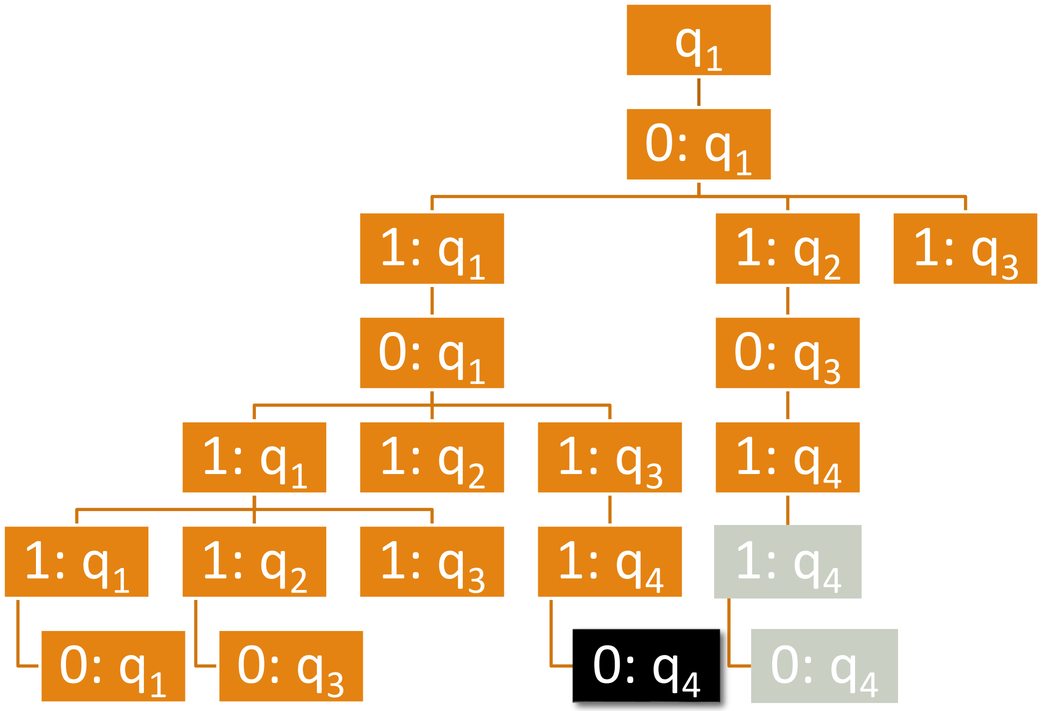



Recall our original NFA

Now recall what computation on it looks like

- Every time we have more than one choice, we spin off as many copies of the NFA as necessary to account for that choice

# **Revisited:** NFA-to-DFA Conversion: Multiple Transitions




Recall our original NFA

Now recall what computation on it looks like

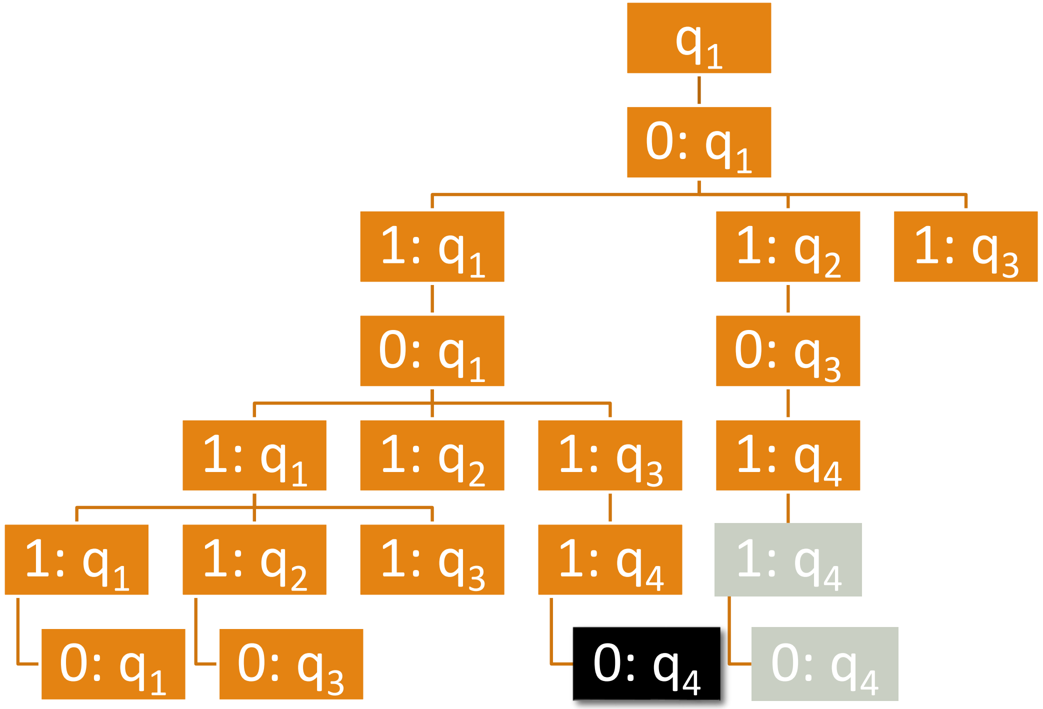
- Every time we have more than one choice, we spin off as many copies of the NFA as necessary to account for that choice

Observe that we only *need* the copies that are actually unique against time and current states

- We only care whether we accept or not, not how many times we accept
- At a given time, it only matters whether we are *in* a given state or not – not how many *times* we are in it
- This means prior computation doesn't matter

**So at a given time, an NFA is in a *set of states***

# **Revisited:** NFA-to-DFA Conversion: Empty Transitions

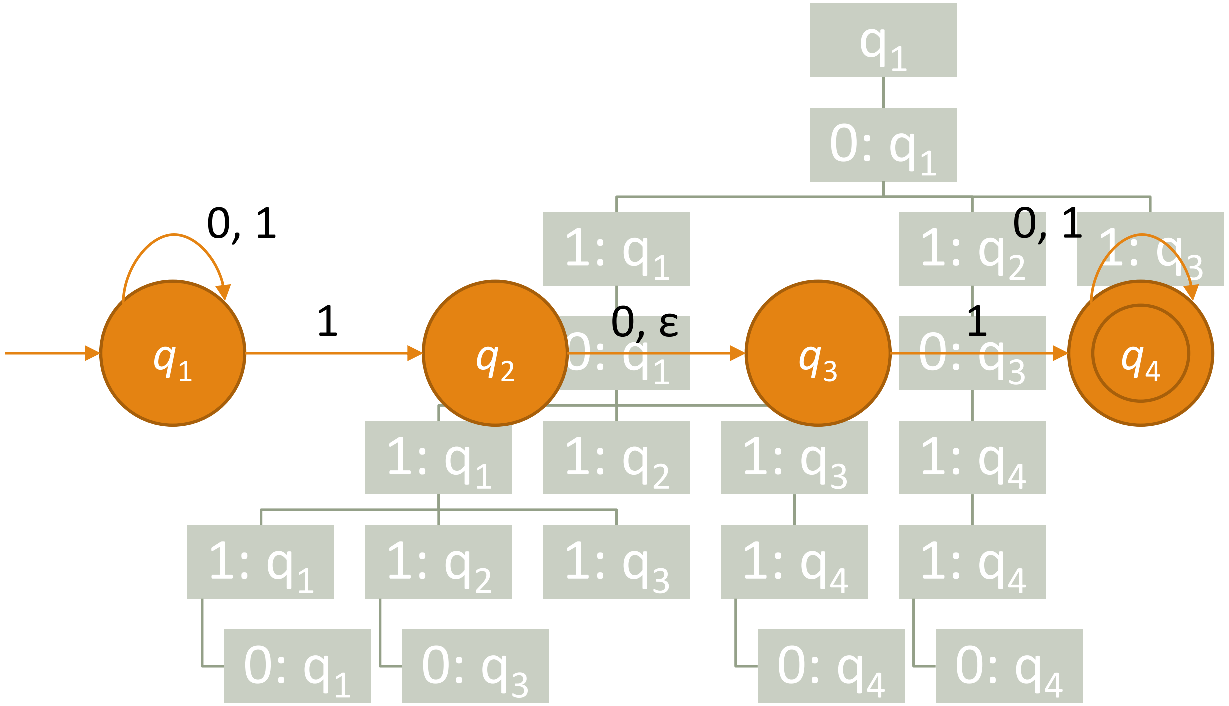



**Multiple transitions imply that at any given time, an NFA is in a *set of states***

What about empty transitions?

- Note that the empty string never appears here

# **Revisited:** NFA-to-DFA Conversion: Empty Transitions

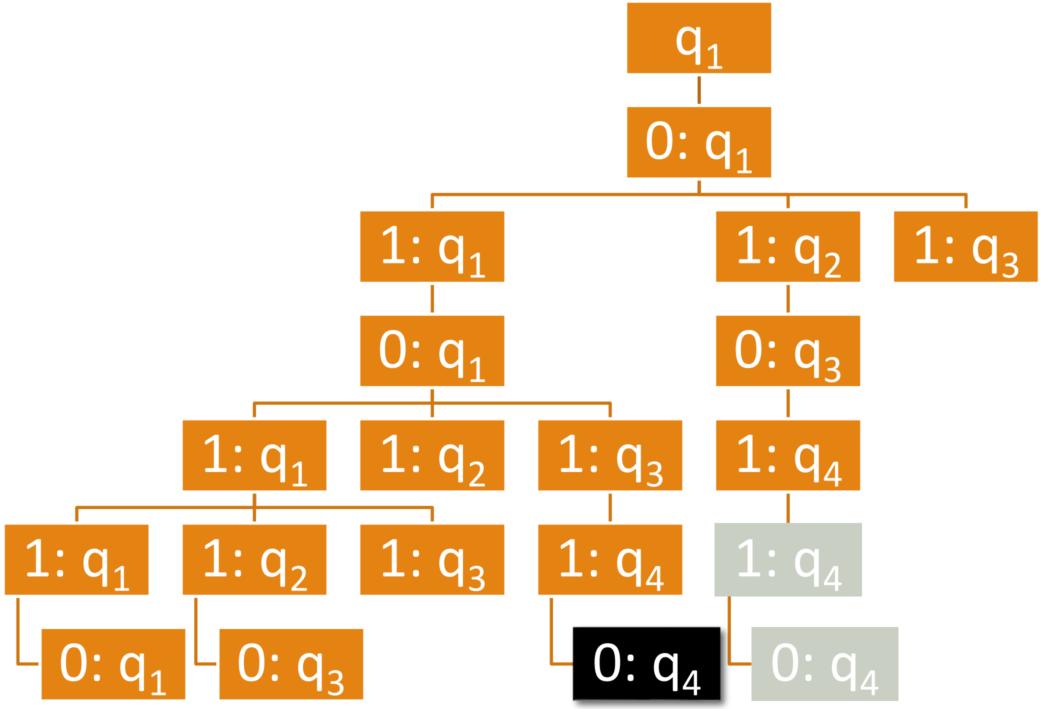



**Multiple transitions imply that at any given time, an NFA is in a *set of states***

What about empty transitions?

- Note that the empty string never appears here
- …but everywhere $q_2$ does, $q_3$ does too

# **Revisited:** NFA-to-DFA Conversion: Empty Transitions



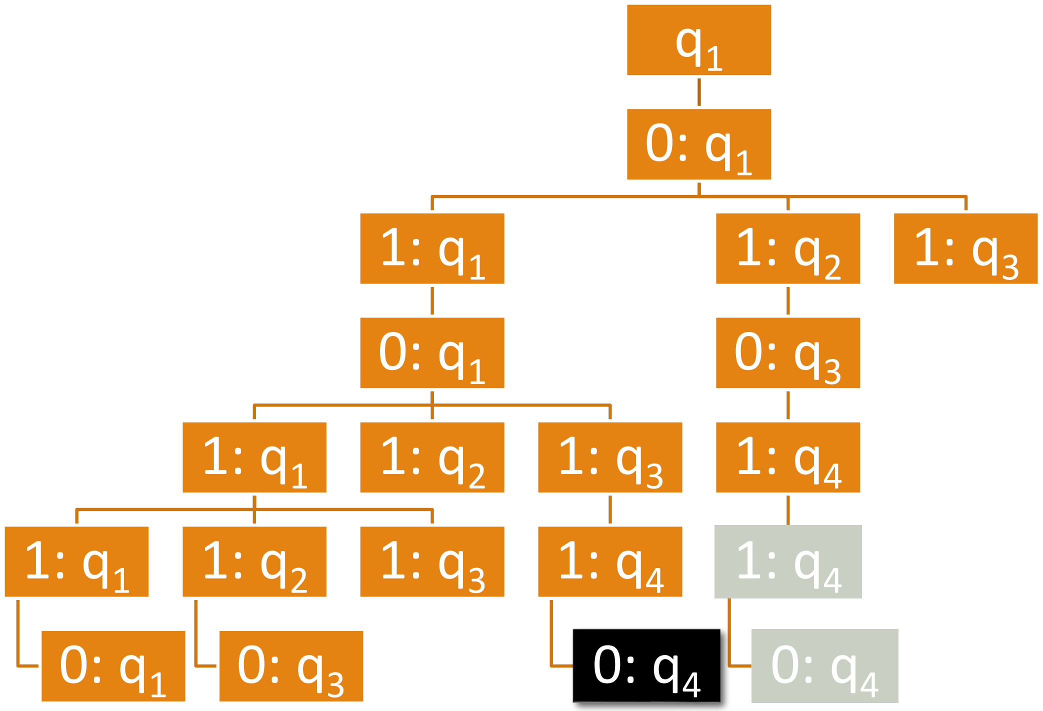

**Multiple transitions imply that at any given time, an NFA is in a *set of states***

What about empty transitions?

- Note that the empty string never appears here
- …but everywhere $q_2$ does, $q_3$ does too
- We can handle empty-string transitions by just "looking forward" at them and including them in the set of possible states at a given moment

**Empty transitions are handled in computation by *including their possibilities* in the set of states**

# **Revisited:** NFA-to-DFA Conversion: Empty Transitions




**Multiple transitions imply that at any given time, an NFA is in a *set of states***

**Empty transitions are handled in computation by *including their possibilities* in the set of states**

What about missing transitions?

- We're already in a *set of states*
- If a state in that set is missing a transition from our next input symbol, it just doesn't add anything to the next set of states

**Missing transitions simply *don't add anything* to the next set of states**

# **Revisited:** NFA-to-DFA Conversion: Simulating the NFA

Keep in mind our three observations about the computation process of an NFA:

1. **Multiple transitions imply that at any given time, an NFA is in a *set of states***
2. **Empty transitions are handled in computation by *including their possibilities* in the set of states**
3. **Missing transitions simply *don't add anything* to the next set of states**

Now take this a bit further:

- The power set P($Q$) of an NFA's states is the set of all possible subsets of its states
- So at any given time, the set of states an NFA is in is an element in P($Q$)
- P($Q$) is *itself* a set

So **on a given transition, an NFA is simply transitioning from one element of P($Q$) to another**

# **Revisited:** NFA-to-DFA Conversion: Simulating the NFA

Keep in mind our three observations about the computation process of an NFA:

1. **Multiple transitions imply that at any given time, an NFA is in a *set of states***
2. **Empty transitions are handled in computation by *including their possibilities* in the set of states**
3. **Missing transitions simply *don't add anything* to the next set of states**

We have also observed that:

4. **On a given transition, an NFA is simply transitioning from one element of P(*Q*) to another**

Now consider empty and missing transitions:

- The possibilities of empty transitions are included in the set of states by look-ahead
- Missing transitions are handled by simply not adding anything to the set of states
- So given the next input symbol, we account for them completely in the next set of states

This means that **an NFA transitions from one element of P(*Q*) to another element of P(*Q*) based only on the next input symbol**

# **Revisited:** NFA-to-DFA Conversion: Simulating the NFA

Keep in mind our three observations about the computation process of an NFA:

1. **Multiple transitions imply that at any given time, an NFA is in a *set of states***
2. **Empty transitions are handled in computation by *including their possibilities* in the set of states**
3. **Missing transitions simply *don't add anything* to the next set of states**

We have also observed that:

4. **On a given transition, an NFA is simply transitioning from one element of P($Q$) to another**
5. **An NFA transitions from one element of P($Q$) to another element of P($Q$) based only on the next input symbol**

So while an NFA is computing, we have:

- A finite set of states it can be in, and
- A way to know which state it will be in next, given only its current state and the input symbol

# **Revisited:** NFA-to-DFA Conversion: Simulating the NFA

Keep in mind our three observations about the computation process of an NFA:

1. **Multiple transitions imply that at any given time, an NFA is in a *set of states***
2. **Empty transitions are handled in computation by *including their possibilities* in the set of states**
3. **Missin**

We have a

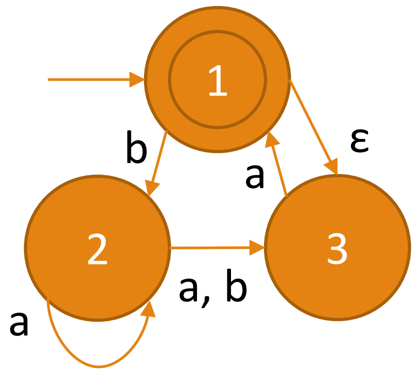
4. **On a gi ... other**
5. **An NFA ... on the next input s**

# That's a DFA.

So while an NFA is computing, we have:

- A finite set of states it can be in, and
- A way to know which state it will be in next, given only its current state and the input symbol

# Building the DFA

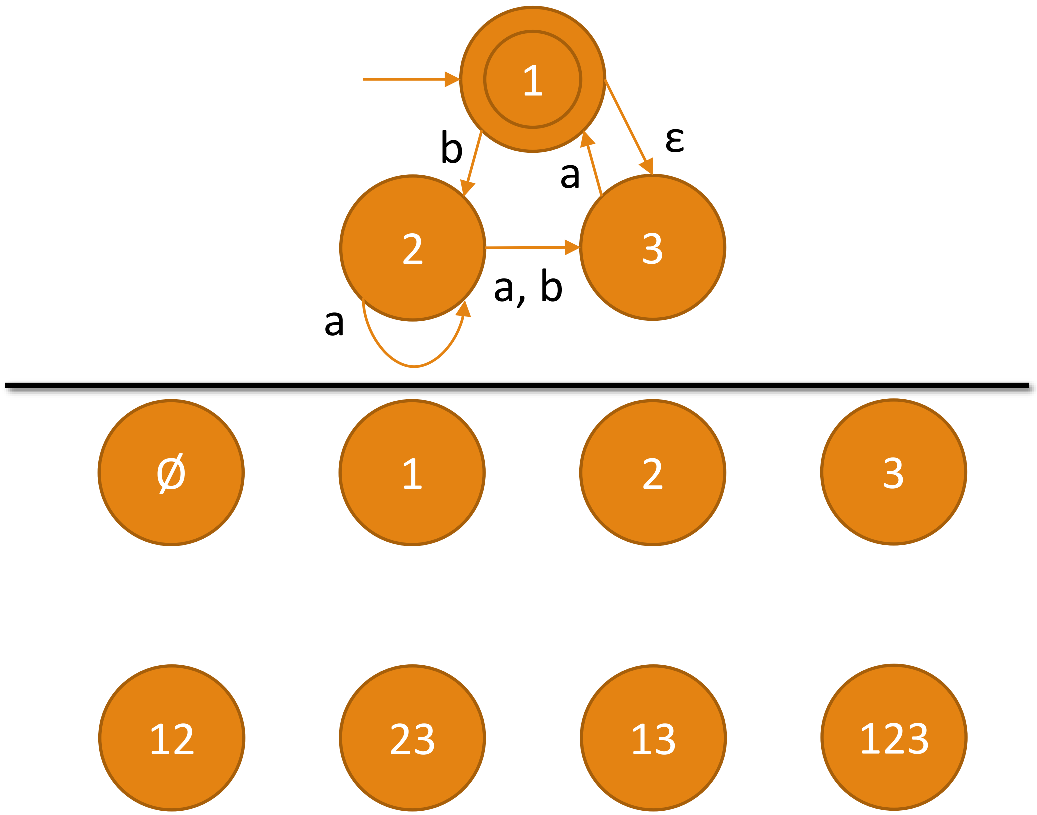



We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

We need to figure out:

- The state set
- The transition function
- The start state
- The final states

# Building the DFA: States

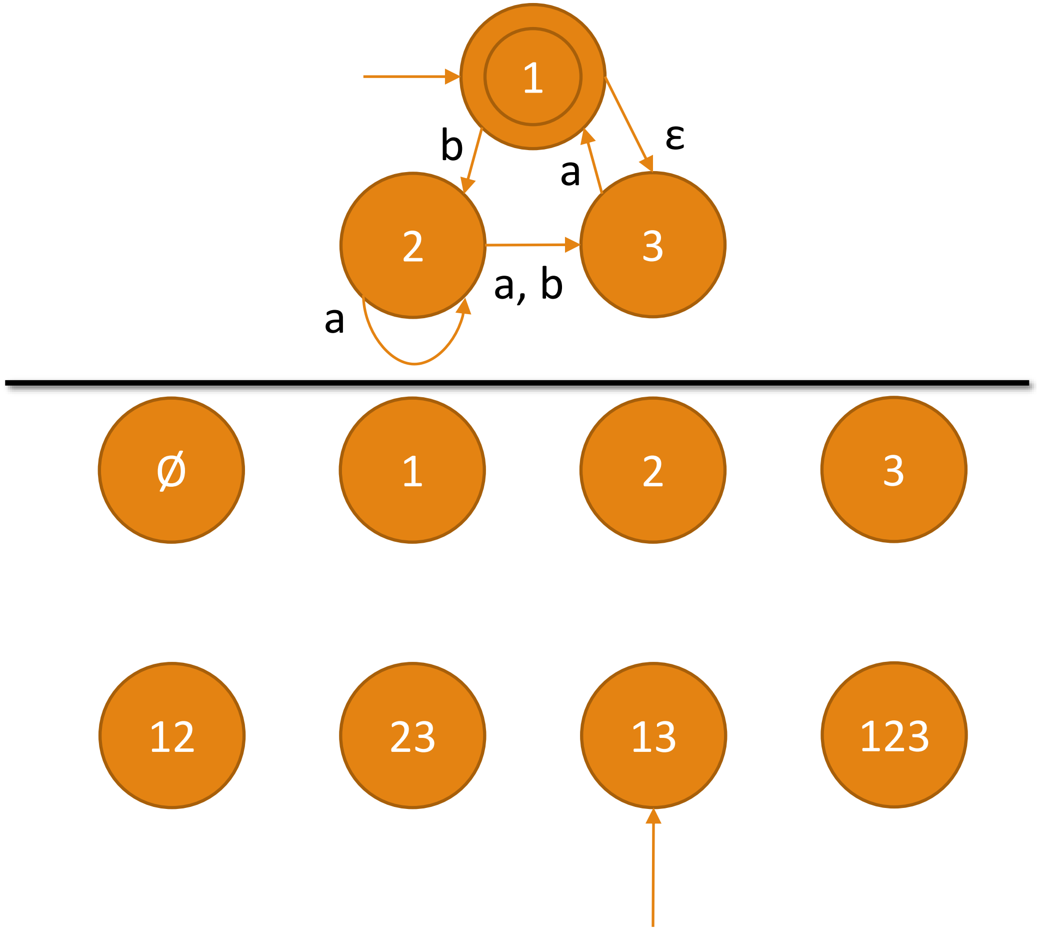



We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

The state set is the easiest part: just remember that we need to simulate being in some subset of the states of *N*, and say:

- **$Q_D$ = P($Q_N$), the power set of $Q_N$**

# Building the DFA: Starting




We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- **$Q_D$ = P($Q_N$), the power set of $Q_N$**

Next let's look at the start state

- Easy answer: the state corresponding to being in, and only in, the start state of the NFA
- …with one wrinkle: empty-string transitions

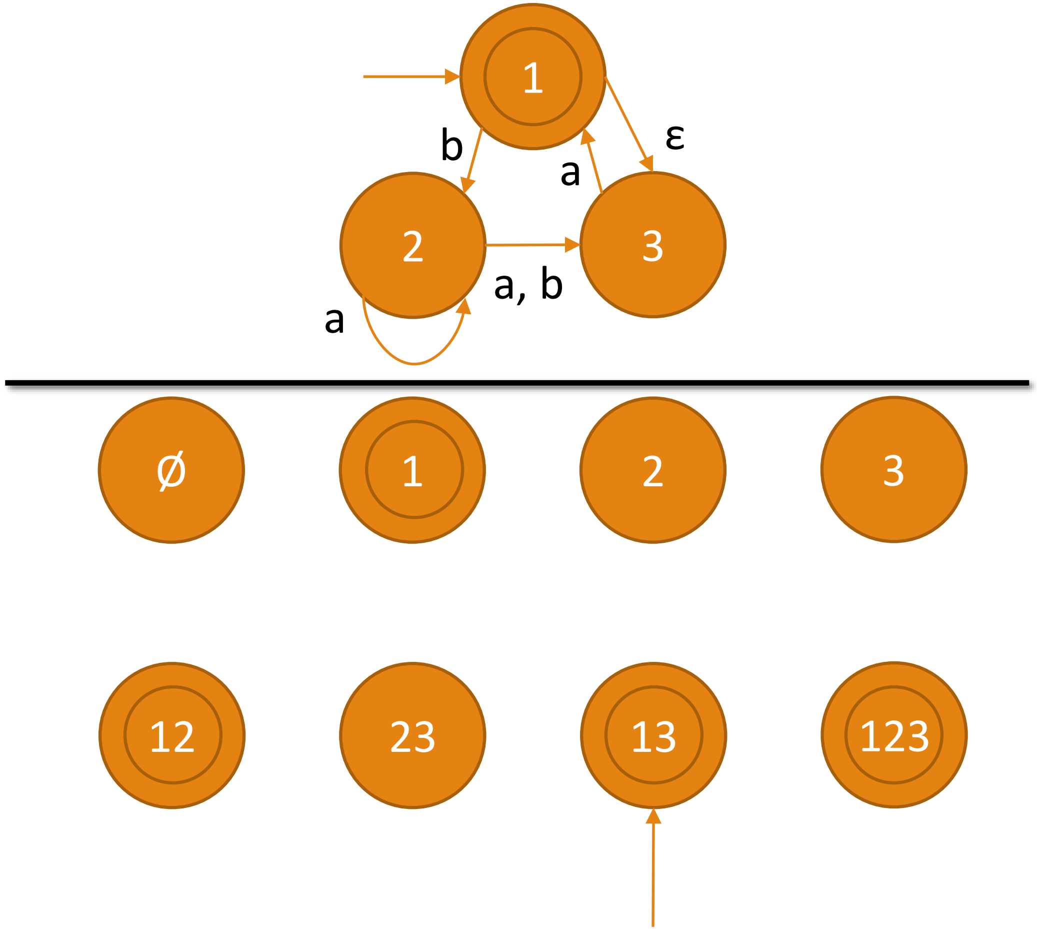
So we need the set-state containing:

- $q_{0N}$
- The states you can reach from $q_{0N}$ with only empty-string transitions

Let's call that set-state E({$q_{0N}$}), and say:

- **$q_{0D}$ = E({$q_{0N}$})**

# Building the DFA: Acceptance




We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- **$Q_D = P(Q_N)$, the power set of $Q_N$**
- **$q_{0D} = E(\{q_{0N}\})$**

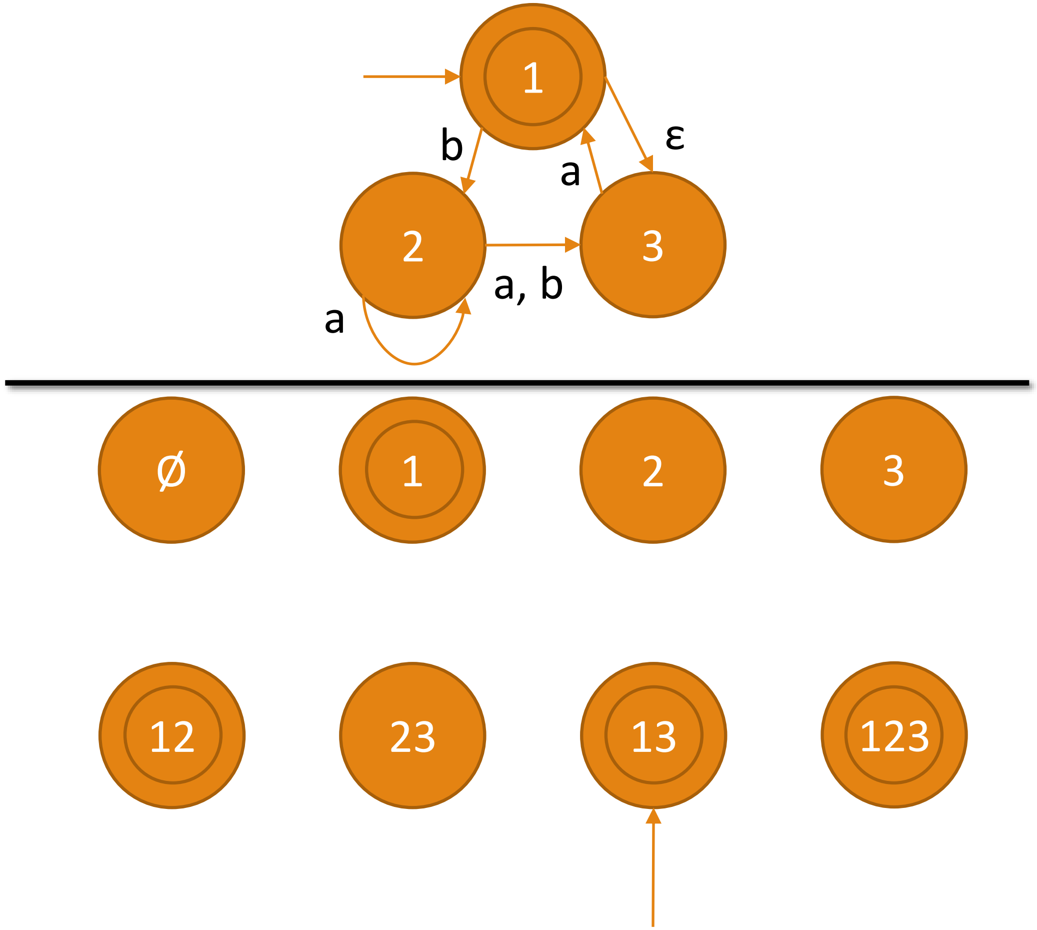
Next the accept states – which actually *are* easy

- Recall that the NFA accepts if it has *any* computation path to an accept state
- This means that in our computation, if there is any state we could be in that is an NFA accept state, we accept

So a state-set accepts if it *contains any accept state* from the NFA

- $F_D = \{R \in Q_D \mid R$ and $F_N$ have a common member$\}$, or
- **$F_D = \{R \in Q_D \mid R \cap F_N \neq \emptyset\}$**

# Building the DFA: Transition




We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- **$Q_D = P(Q_N)$, the power set of $Q_N$**
- **$q_{0D} = E(\{q_{0N}\})$**
- **$F_D = \{R \in Q_D \mid R \cap F_N \neq \emptyset\}$**

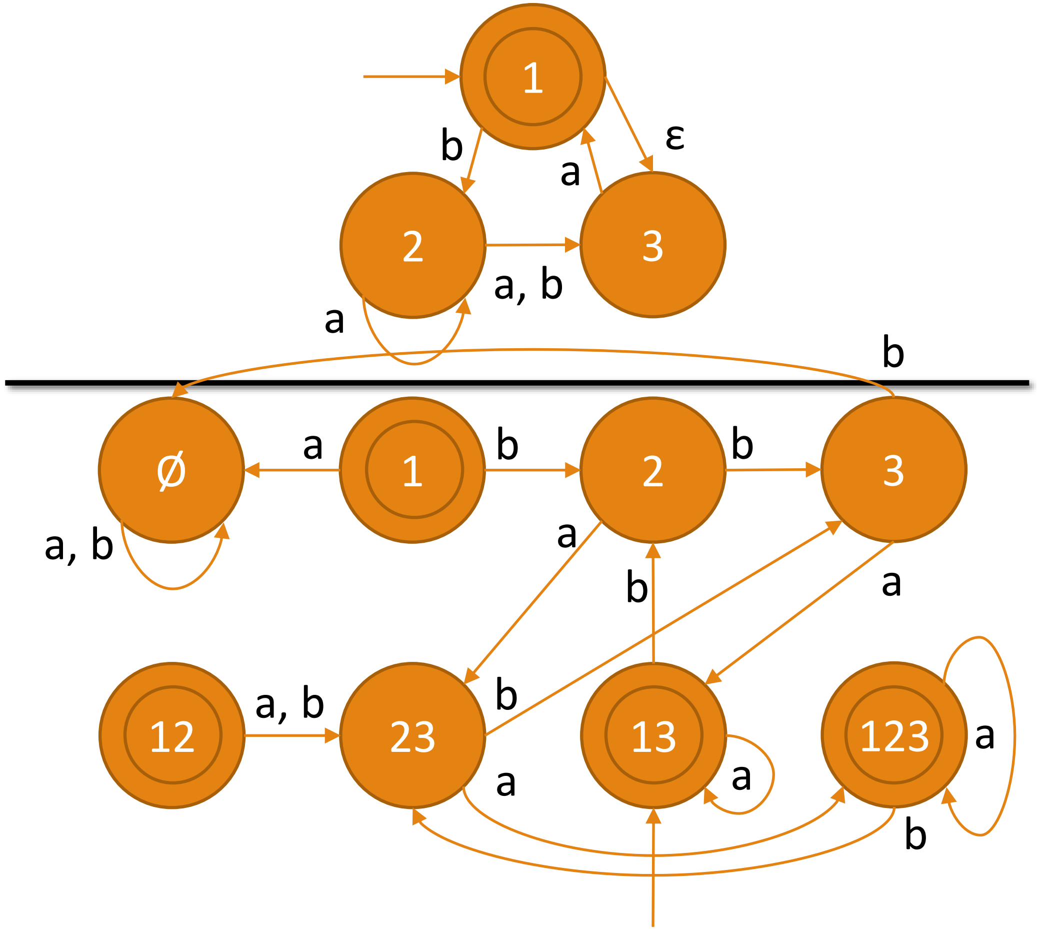
Now the transition function. Remember:

- The NFA transitions between *sets of states*
- We simulate that by having a state for each possible set

So to transition on a given symbol *a*, we:

- Look at *all* the NFA states we are currently simulating
- Look at all the states *they* can possibly transition to on *a*
- Transition to the set of all of those states

# Building the DFA: Transition




We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- **$Q_D$ = P($Q_N$), the power set of $Q_N$**
- **$q_{0D}$ = E({$q_{0N}$})**
- **$F_D = \{R \in Q_D \mid R \cap F_N \neq \emptyset\}$**

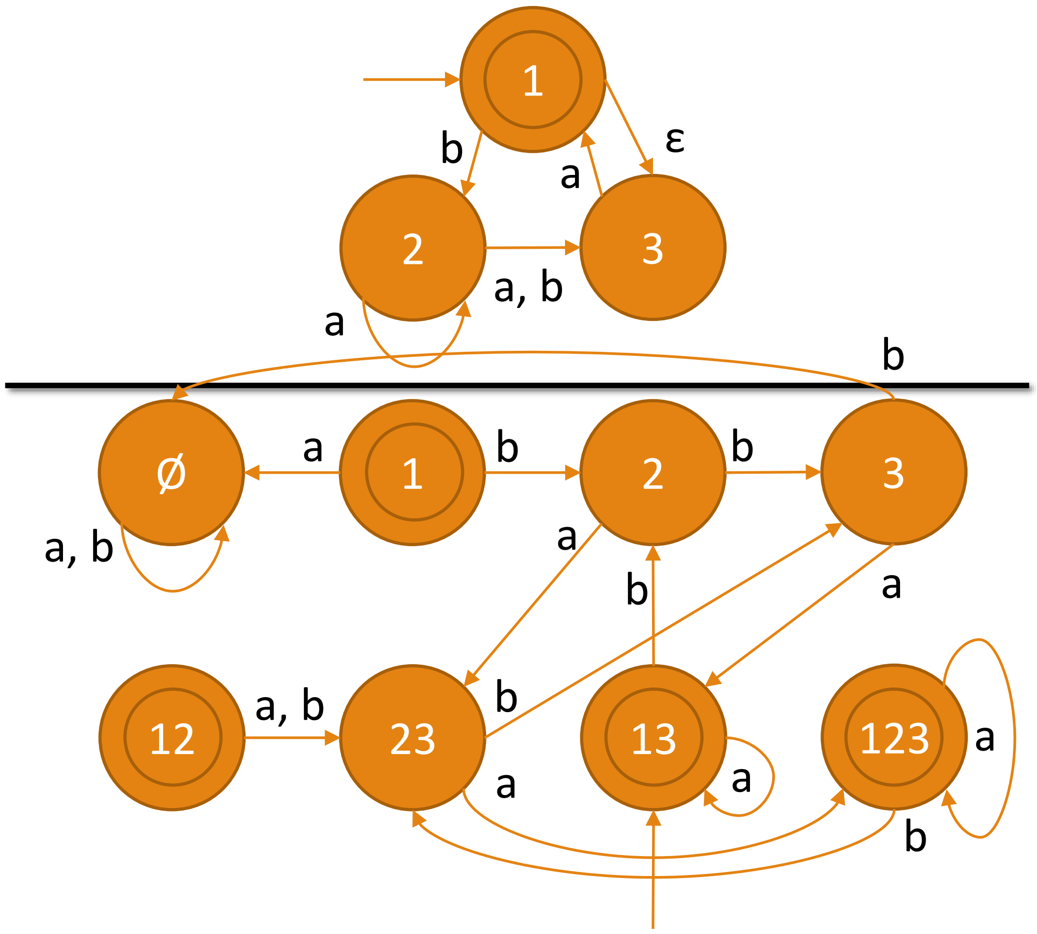
To transition on a given symbol *a*, we:

- Look at *all* the NFA states we are currently simulating
- Look at all the states *they* can possibly transition to on *a*
- Transition to the set of all of those states

So we define $\delta_D\colon\ Q_D \times \Sigma \rightarrow Q_D$ as:

- $\delta_D(R, a) = \{q \in Q_N \mid q \in \delta_N(r, a) \text{ for some } r \in R\}$
- …almost

# Building the DFA: Transition




We want to build a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- **$Q_D = P(Q_N)$, the power set of $Q_N$**
- **$q_{0D} = E(\{q_{0N}\})$**
- **$F_D = \{R \in Q_D \mid R \cap F_N \neq \emptyset\}$**

We just defined $\delta_D\colon\ Q_D \times \Sigma \rightarrow Q_D$ as:

- $\delta_D(R, a) = \{q \in Q_N \mid q \in \delta_N(r, a) \text{ for some } r \in R\}$

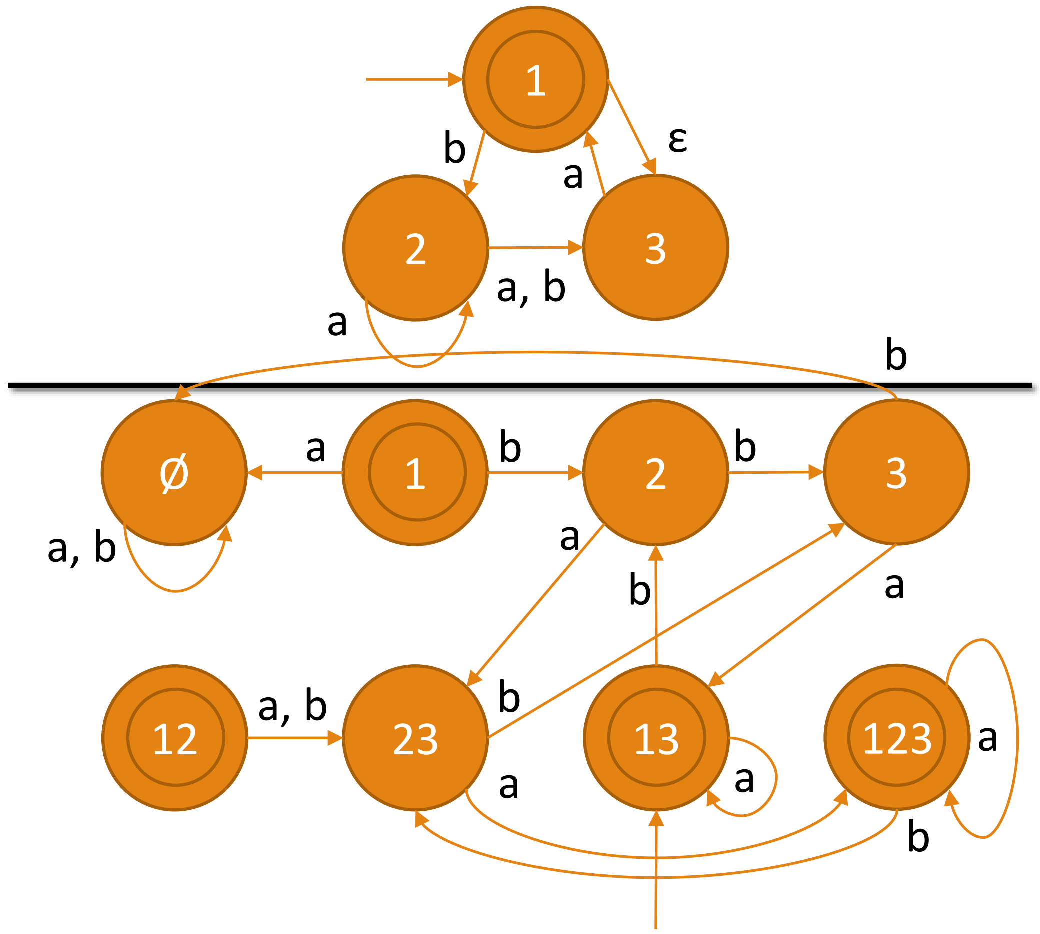
But we need to consider empty string transitions

- We can just do this the same way we did with the start state

So we finally say:

- $\boldsymbol{\delta_D(R, a) = \{q \in Q_N \mid q \in E(\delta_N(r, a)) \textbf{ for some } r \in R\}}$

# Building the DFA: Transition




We **have built** a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

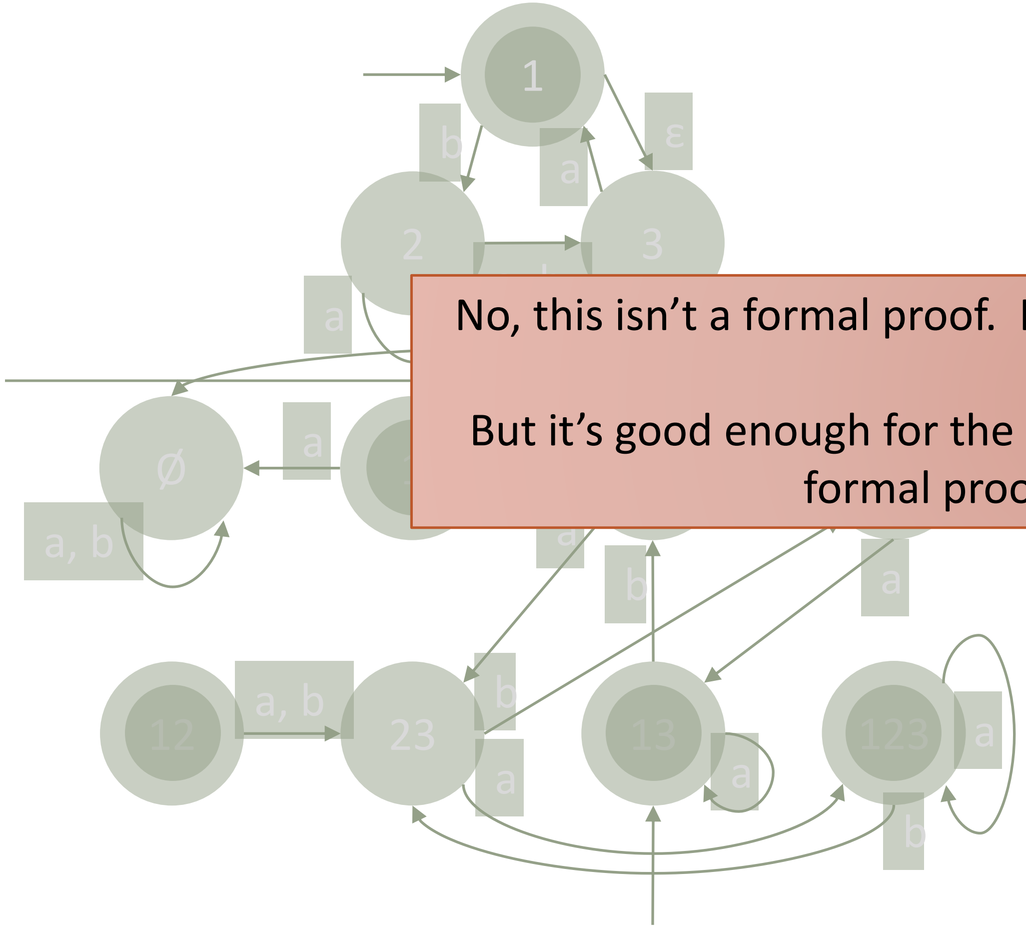
- **$Q_D$ = P($Q_N$), the power set of $Q_N$**
- **$q_{0D} = E(\{q_{0N}\})$**
- **$F_D = \{R \in Q_D \mid R \cap F_N \neq \emptyset\}$**
- $\boldsymbol{\delta_D(R, a) = \{q \in Q_N \mid q \in E(\delta_N(r, a)) \text{ for some } r \in R\}}$

From our observations during construction, *D* is always in a state corresponding to the subset of states *N* could be in on the same input

We have observed that for any NFA *N*, a corresponding DFA *D* exists that recognizes the same language as *N*

Hence, by definition of regular languages, any language recognized by an NFA is regular.

# Building the DFA: Transition




We **have built** a DFA $D = \{ Q_D, \Sigma, \delta_D, q_{0D}, F_D \}$ that simulates NFA $N = \{ Q_N, \Sigma, \delta_N, q_{0N}, F_N \}$

- $Q_D = P(Q_N)$, **the power set of** $Q_N$
- $q_{0D} = E(\{q_{0N}\})$

$\delta_N(r, a))$ **for some** $r \in R\}$

construction, *D* is always ... subset of states *N* could be in on the same input

No, this isn't a formal proof. I'm not going to draw the little square.

But it's good enough for the book, let alone this class, and doing a formal proof would take weeks.

We have observed that for any NFA *N*, a corresponding DFA *D* exists that recognizes the same language as *N*

Hence, by definition of regular languages, any language recognized by an NFA is regular.

# Next Time:
# Non-Regular Languages