More about NFAs

Mark Greenstreet, CpSc 421, Term 1, 2006/07

Lecture Outline

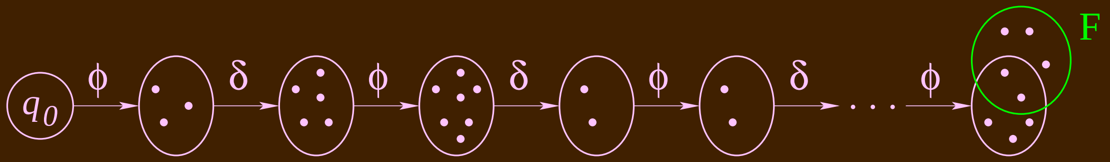
More about NFAs

- Defining NFA Acceptance
- Examples
- Equivalence of NFAs and DFAs

NFA Acceptance

- An NFA is a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$.
- An NFA accepts string w iff there is at least one path from q_0 to some state in F that can be taken by reading w.
- How we will formalize this:
  - Define δ to return a set of states.
    $\delta(q, c)$ is the set of all states that can be reached from state q by reading symbol c.
  - Extend our definition of δ to handle strings.
    This is the same general approach as we took with DFAs.
  - We need to handle ϵ moves.

    We'll define $\phi(q)$ to be the set of all states that are reachable from q by only taking ϵ moves.
- Here's a picture of this process:



Transition Relations for NFAs

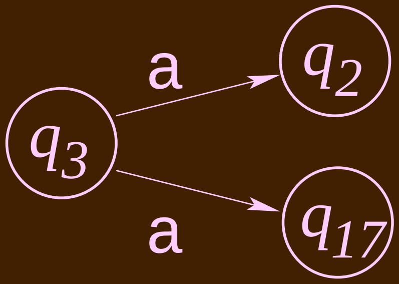
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$.

Transition Relations for NFAs

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$.
  Let's say this in Java:

  ```
  Collection delta(State q, Symbol c) { ...}
  ```

  `delta` is a function that takes a state and a symbol as an argument, and returns a set of states as its result. This set is the set of all states that are legal successors to `q` when symbol `c` is read.



$\delta(q_3, \text{a}) = \{q_2, q_{17}\}$

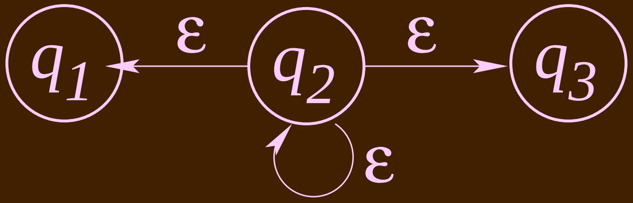
Transition Relations for NFAs

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$.
  Let's say this in Java:

  ```
  Collection delta(State q, Symbol c) { ...}

  Collection delta(State q) { ...}
  ```

  We also allow the NFA to make ϵ-moves without consuming any input. In Java, we could express this by overloading `delta` to have a version that takes a single argument, the state, and no input symbol. In our mathematical formulation, we add ϵ to the set Σ, this is the $(\Sigma \cup \{\epsilon\})$ part of the definition above.

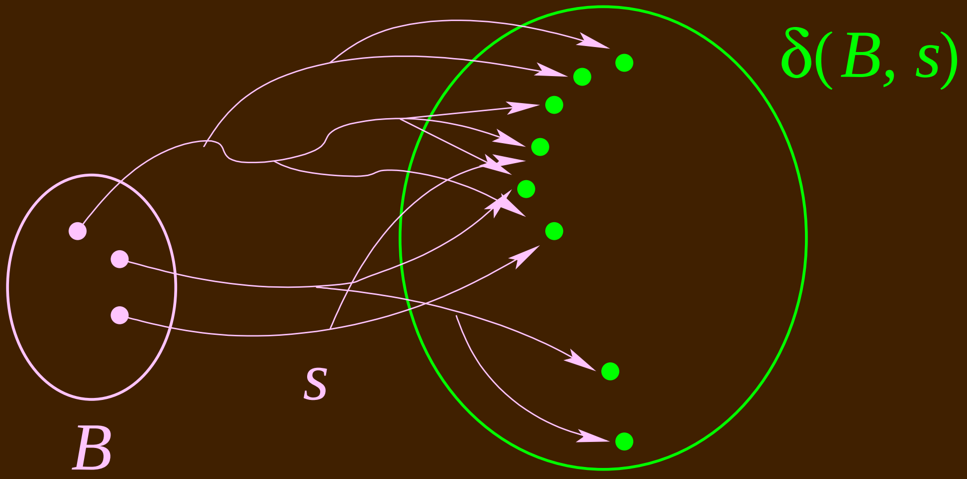


$$\delta(q_2, \epsilon) = \{q_1, q_2, q_3\}$$

Extending δ to Strings (1/3)

- We would like to know all states that an NFA can reach after reading a string.
- For DFAs, we defined $\delta(q, w)$ to be the state reached by reading string s when starting from state q.
- For an NFA, we can so something similar, we'll define $\delta(B, w)$ to be the *set* of states that can be reached by reading string s when starting from any state in set B.
- In other words, if you can pick some state in B and some set of choices for the NFA to make when reading w so that the machine lands in state q, then $q \in \delta(B, w)$.

Extending δ to Strings (2/3)



- Math: $\delta : 2^Q \times \Sigma^* \rightarrow 2^Q$.

$$\begin{array}{rcl} \delta(B, \epsilon) & = & B \\ \delta(B, x \cdot c) & = & \bigcup\limits_{q \in \delta(B,x)} \delta(q, c) \end{array}$$

- Java: `Collection delta(Collection B, SymbolString s) { ...}`

Extending δ to Strings (3/3)

- Math: $\delta : 2^Q \times \Sigma^* \to 2^Q$.

$$\begin{aligned} \delta(B, \epsilon) &= B \\ \delta(B, x \cdot c) &= \bigcup_{q \in \delta(B,x)} \delta(q, c) \end{aligned}$$

- Java:

```
Collection delta(Collection B, SymbolString s) {
  if(s.length() == 0) return(B);
  else {
    SymbolString x = s.substring(0, s.length()-1);
    Symbol c = s.SymbolAt(s.length()-1);
    Collection A = new Collection();
    Iterator it = delta(B, x).iterator();
    while(it.hasNext())
      A.add(delta(it.next(), c));
    return(A);
  }
}
```

Handling ϵ Moves

- As defined above, $\delta(B, w)$ doesn't handle ϵ moves.
- For each state q, we'll calculate the set of states that can be reached from q by taking zero or more ϵ moves. We'll call this set $\phi(q)$.

$$\begin{tikzcd}
 & q & t & \\
p \arrow[ur] \arrow[r] \arrow[dr] & r \arrow[ur] & & \\
 & s \arrow[r] & u \arrow[ul] \arrow[r] & v
\arrow[from=1-3, to=3-3]
\end{tikzcd}$$

$$\begin{aligned}
\phi(p) &= \{p, q, r, s, t, u, v\} \\
\phi(q) &= \{q\} \\
\phi(r) &= \{r, t, u, v\} \\
\phi(s) &= \{r, s, t, u, v\} \\
\phi(t) &= \{r, t, u, v\} \\
\phi(u) &= \{r, t, u, v\} \\
\phi(v) &= \{v\}
\end{aligned}$$

A Formula for ϕ

$p \in \phi(q)$ iff

- $p = q$, or
- $\exists r \in phi(q).\ p \in \delta(r, \epsilon)$.
- In Java,

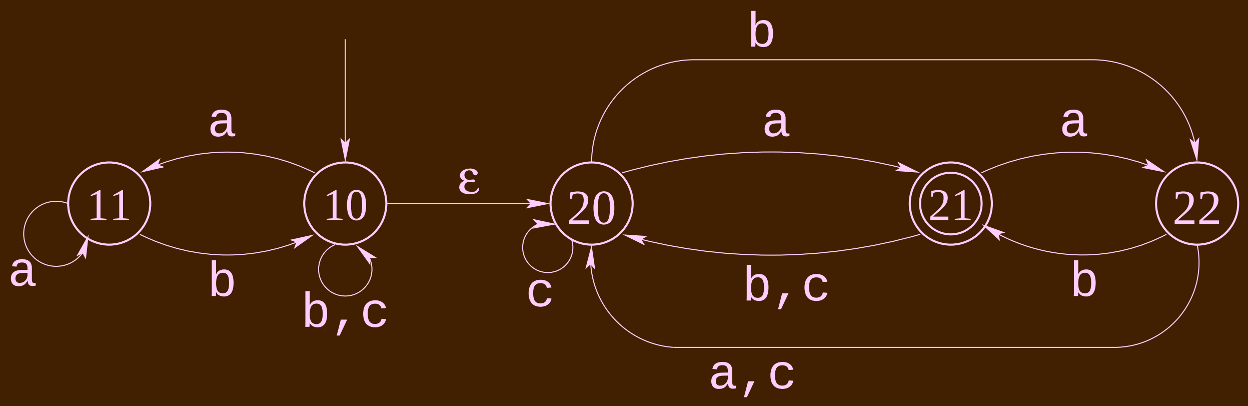
```
Collection phi(State q) {
  // Compute the set of all states reachable
  // from q in the directed graph represented
  // by the transition relation.
  // Use your favorite graph traversal algorithm.
}
```

Combining δ and ϕ

- $$\begin{aligned} \delta(B,\epsilon) &= B \\ \delta(B, x\cdot c) &= \phi\left(\bigcup_{q\in\delta(B,x)} \delta(q,c)\right) \end{aligned}$$
- NFA (Q,Σ,δ,q_0,F) accepts string w iff

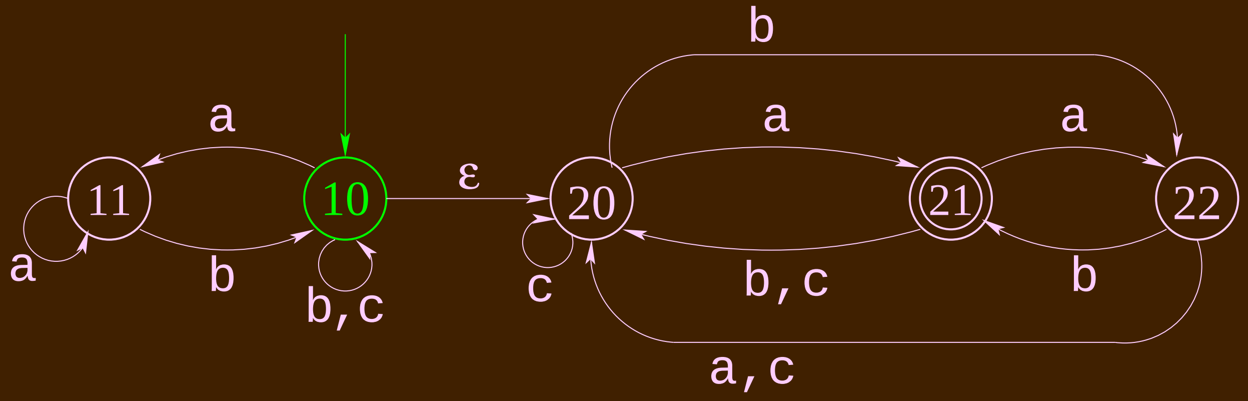
$$\delta(\phi(\{q_0\}),w)\bigcap F \neq \emptyset$$

Our Example Again



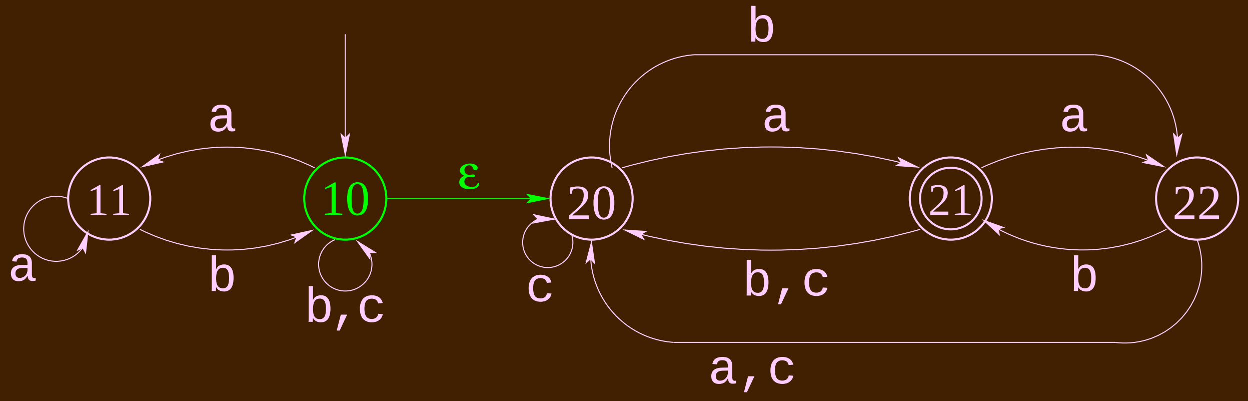
Consider reading the string: abcabbbb.

Our Example Again



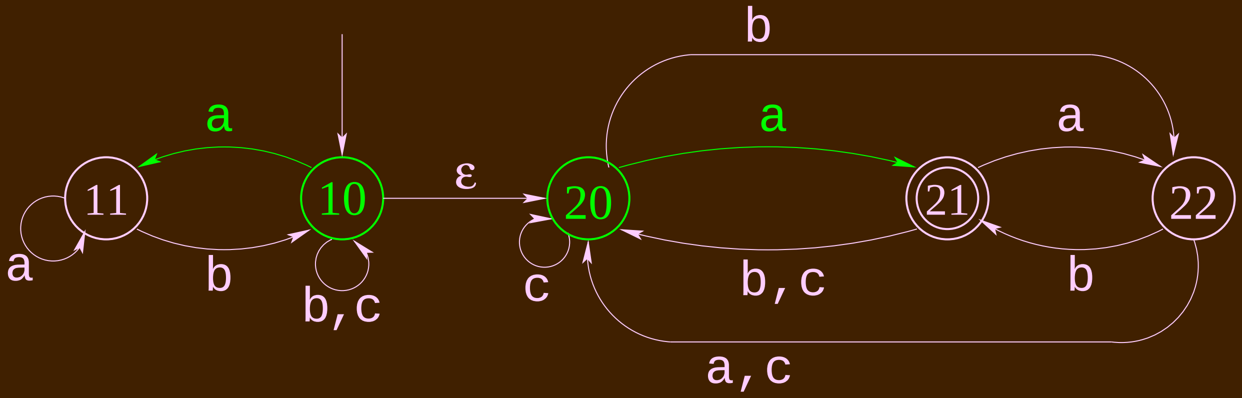
Consider reading the string: abcabbbb.

Our Example Again



Consider reading the string: ϵabcabbbb.

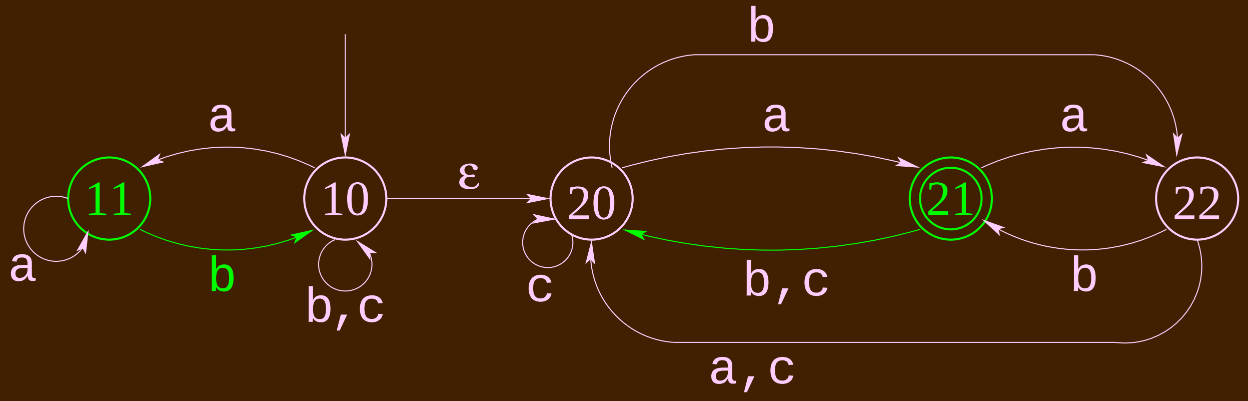
Our Example Again



Consider reading the string: abcabbbb.

Reject: ϵ

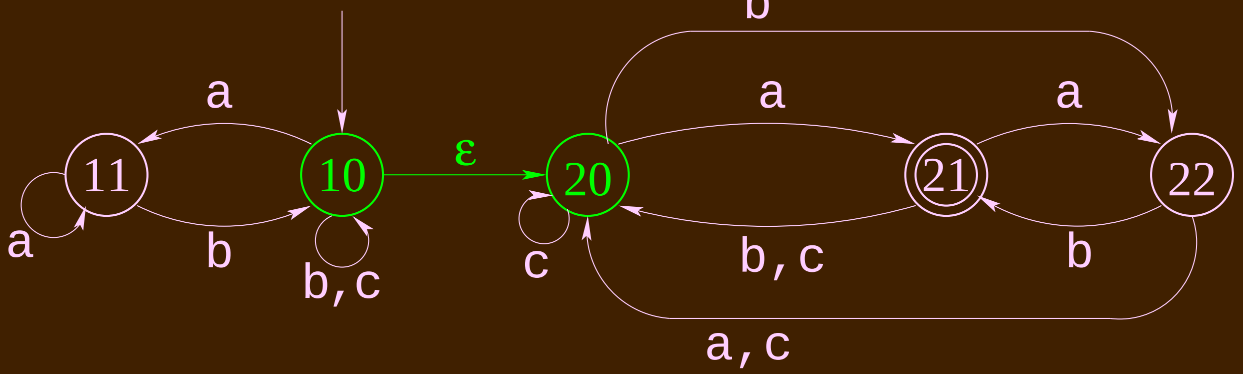
Our Example Again



Consider reading the string: abcabbbb.

Accept: a

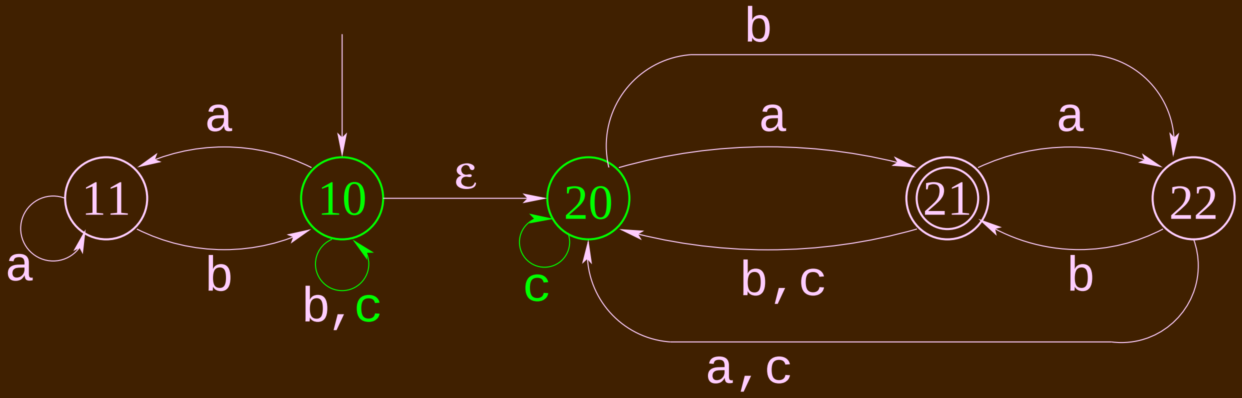
Our Example Again



Consider reading the string: abϵcabbbb.

Reject: a

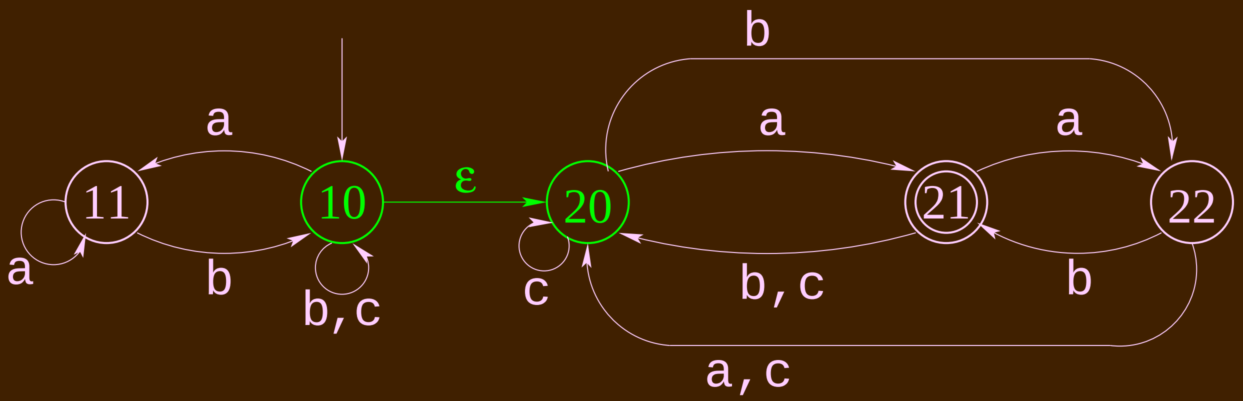
Our Example Again



Consider reading the string: abcabbbb.

Reject: ab

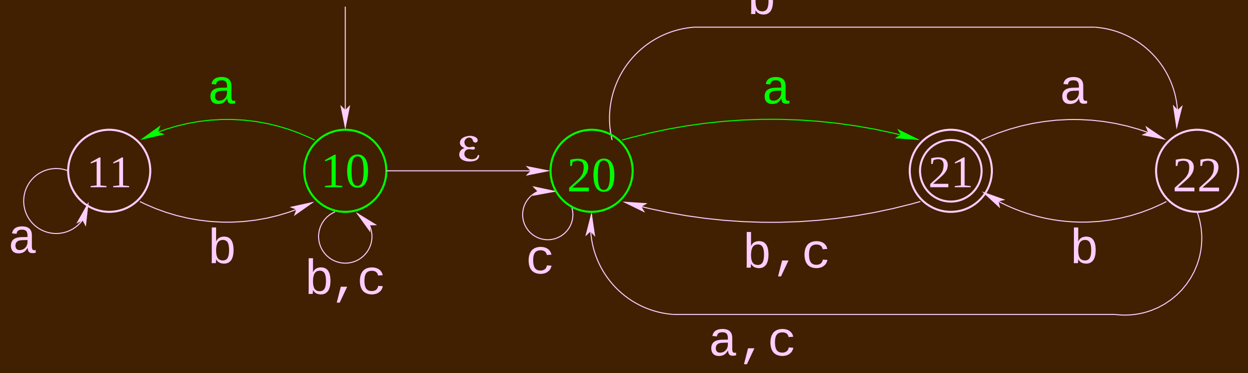
Our Example Again



Consider reading the string: abcϵabbbb.

Reject: ab

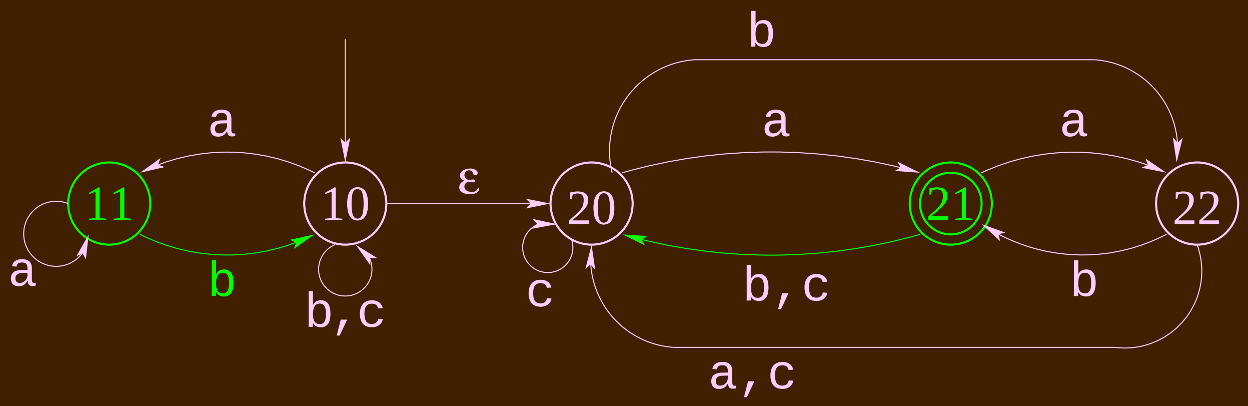
Our Example Again



Consider reading the string: abcabbbb.

Reject: abc

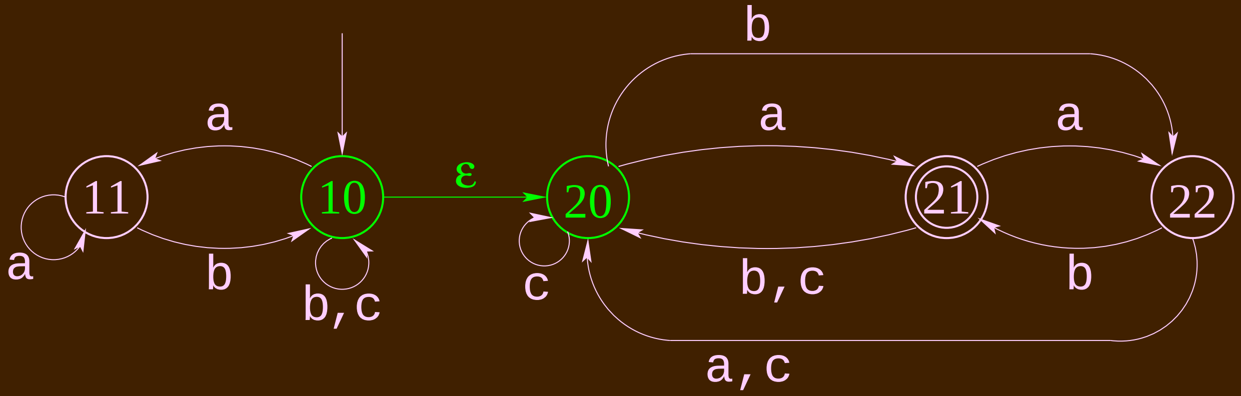
Our Example Again



Consider reading the string: abcabbbb.

Accept: abca

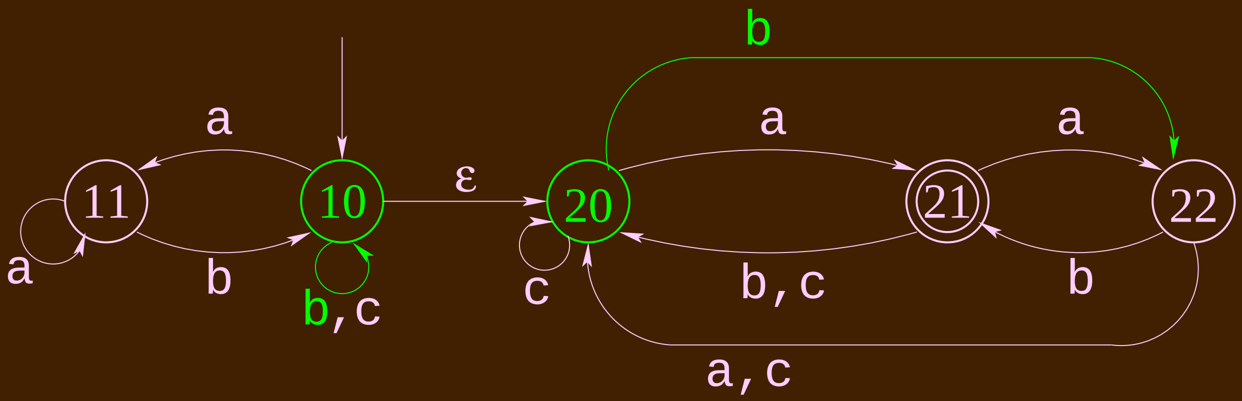
Our Example Again



Consider reading the string: abcabϵbbb.

Accept: abca

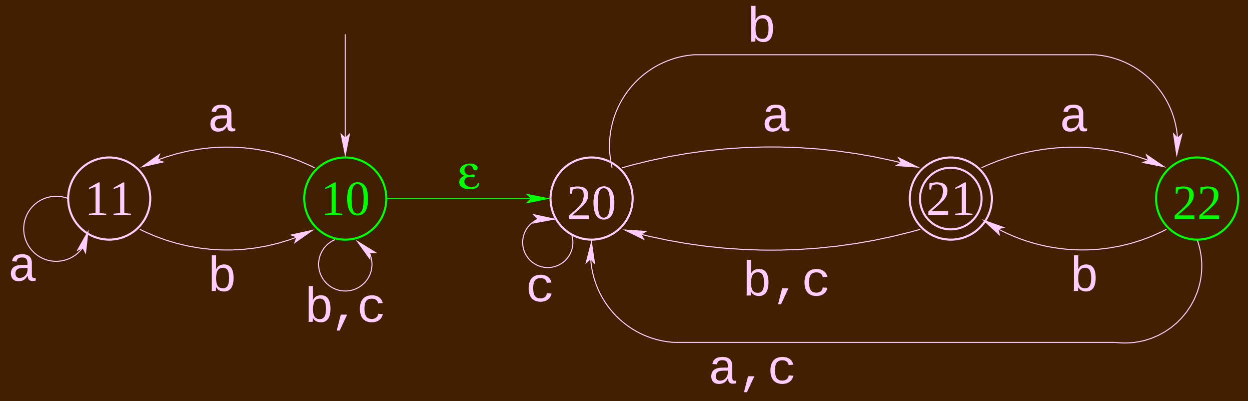
Our Example Again



Consider reading the string: abcabbbb.

Reject: abcab

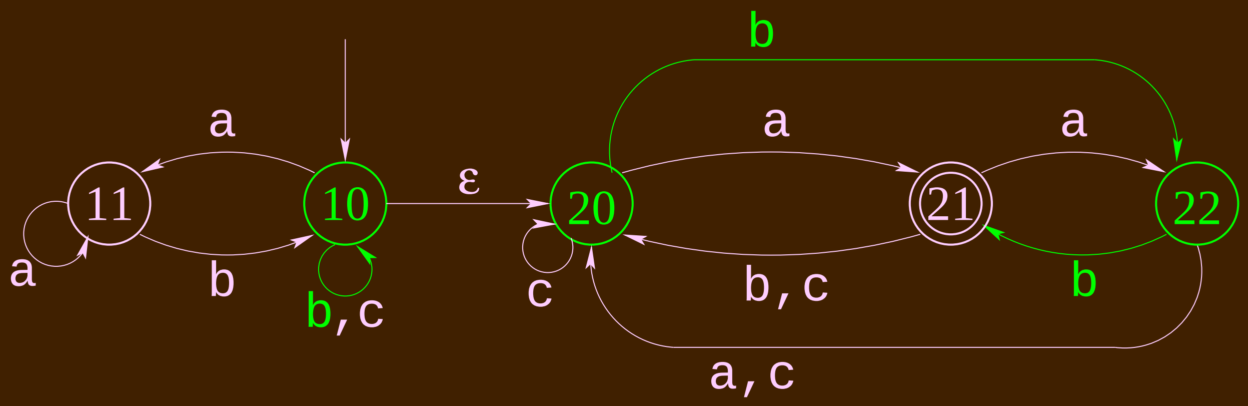
Our Example Again



Consider reading the string: abcabbϵbb.

Reject: abcab

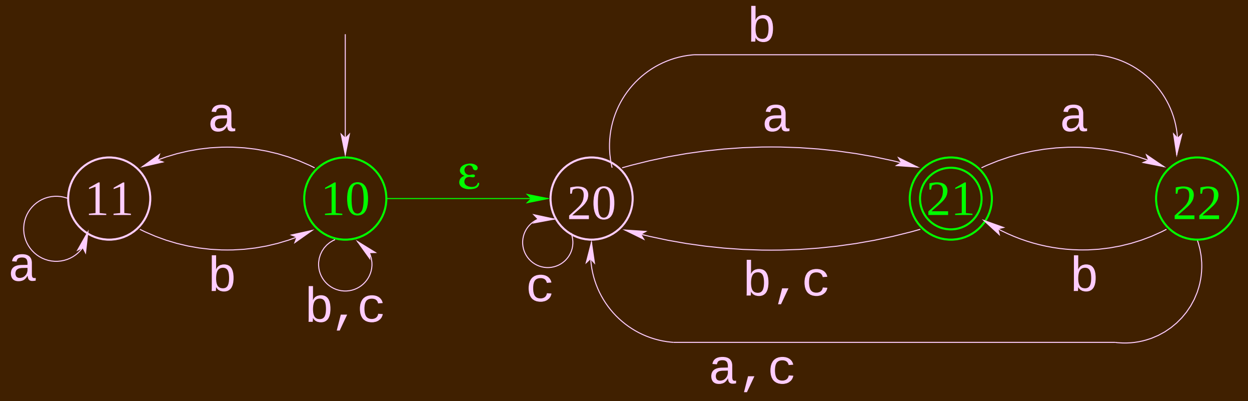
Our Example Again



Consider reading the string: abcabbbb.

Reject: abcabb

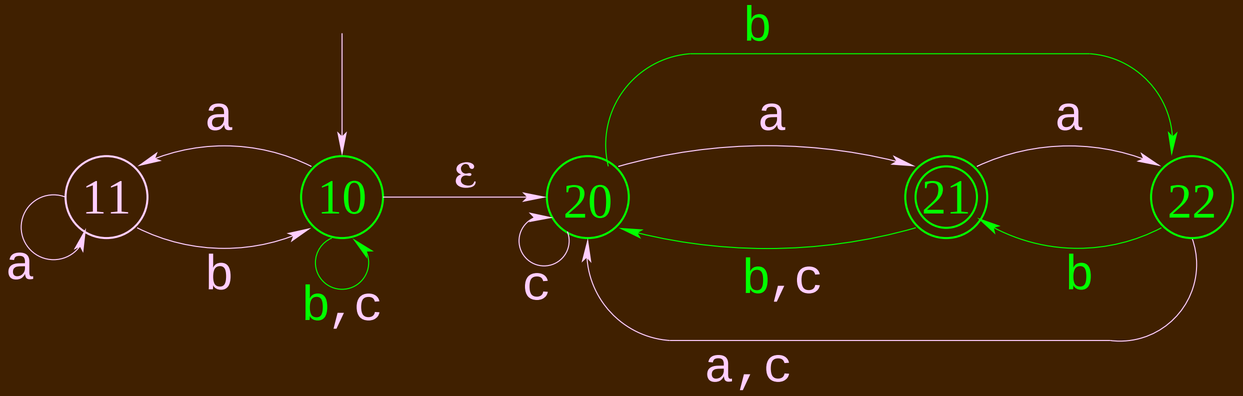
Our Example Again



Consider reading the string: abcabbbϵb.

Reject: abcabb

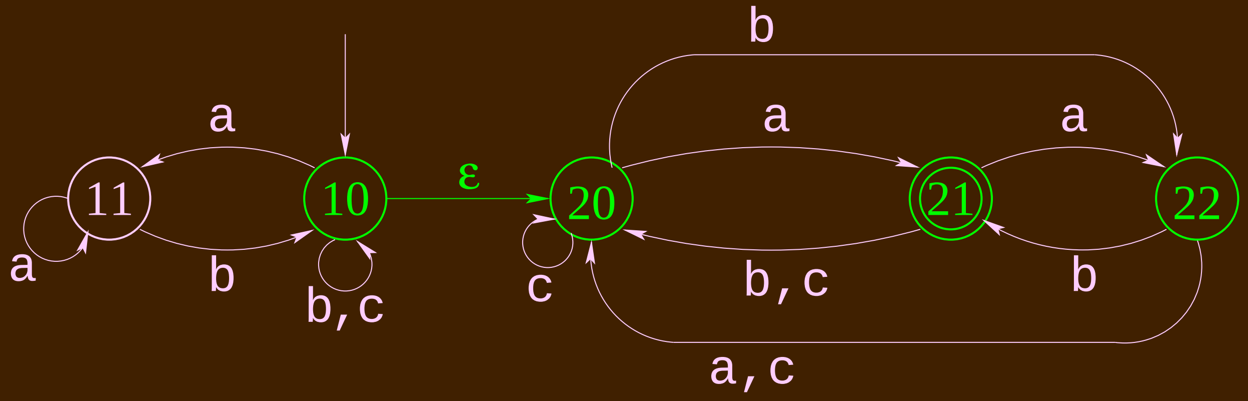
Our Example Again



Consider reading the string: abcabbbbb.

Accept: abcabbb

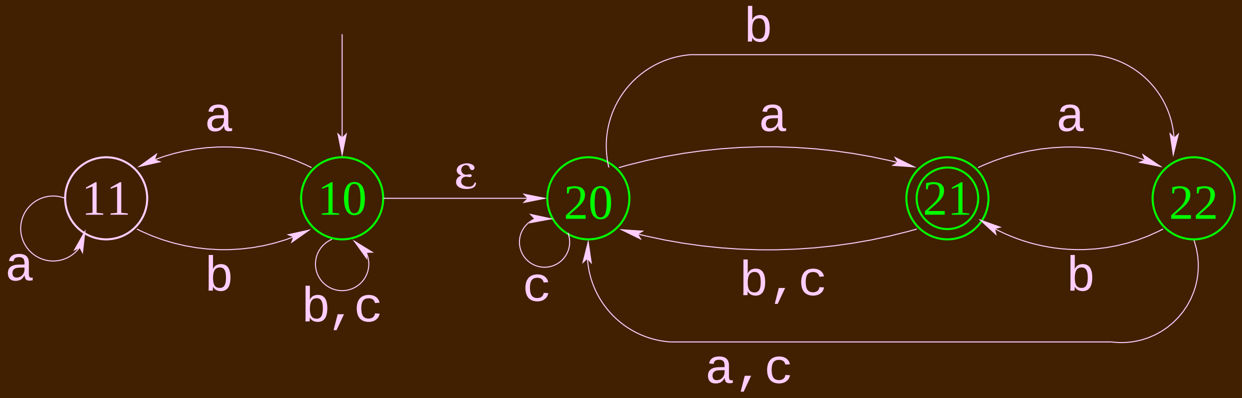
Our Example Again



Consider reading the string: abcabbbbbϵ.

Accept: abcabbb

Our Example Again



Consider reading the string: abcabbbbb.

Accept: abcabbbb

Equivalence of NFAs and DFAs

- We want to show that the sets of languages recognized by NFAs and the set recognized by DFAs are the same.
- Showing that every language recognized by a DFA is also recognized by an NFA is easy: every DFA is an NFA.
- Showing that every language recognized by an NFA is also recognized by a DFA is more work. That's what we'll take on in the next few slides.

From an NFA to an Equivalent DFA

- Basic strategy: we noted that the definitions of NFAs and DFAs are quite similar – the main difference is the definition of δ.
- Given an NFA, $N = (Q_N, \Sigma, \delta_N, q_{0,N}, F_N)$, we'll construct a DFA, $D = (Q_D, \Sigma, \delta_D, q_{0,D}, F_D)$ such that $L(D) = L(N)$.
- Our strategy is based on thinking about how we defined the acceptance condition for an NFA – we wrote a function that keeps track of the set of possible states that the NFA can be in after reading each symbol of the input.
- If Q_N is finite, then the set 2^{Q_N} is finite as well (even though it may be very big). We'll let $Q_D = 2^{Q_N}$: now each state of the DFA describes the *set* of states that the NFA could be in at that point.
- Now, we need to define δ_D, $q_{0,d}$, and F_D.
  We'll start with δ_D: once we have that, $q_{0,d}$ and F_D are pretty straightforward.

Defining δ_D (1/3)

- By the definition of D, $\delta_D : (Q_D \times \Sigma) \to Q_D$.
  Having chosen $Q_D = 2^{Q_N}$ we have $\delta_D : (2^{Q_D} \times \Sigma) \to 2^{Q_D}$.
- We also want

$$\delta_D(q_{0,d}, w) \in F_D \quad \Leftrightarrow \quad (\delta_N(\phi_N(\{q_{0,N}\}), w) \bigcap F_N) \neq \emptyset$$

  The left side of the $\Leftrightarrow$ says that D accepts w, and the right side says that N accepts w.
- We'll look at the definition of δ_N to find a way to define δ_D.

Defining δ_D (2/3)

- Definition of δ_D (from Sept. 13 lecture):

$$\begin{aligned}\delta_D(q,\epsilon) &= q\\ \delta_D(q,x\cdot c) &= \delta_D(\delta_D(q,x),c)\end{aligned}$$

- Definition of δ_N (from slide 10):

$$\begin{aligned}\delta_N(B,\epsilon) &= B\\ \delta_N(B,x\cdot c) &= \phi\left(\bigcup_{q\in\delta_N(B,x)} \delta_N(q,c)\right)\end{aligned}$$

- If we can combine the ϕ, $\bigcup_{q\in\ldots}$ and $\delta_N(q,c)$ into a single function, we can make the definition of δ_N for strings look like the definition of δ_D. Let:

$$\begin{aligned}\delta_\dagger(B,c) &= \\ \delta_\dagger(B,c) &= \phi\left(\bigcup_{q\in\delta_N(B,x)} \delta_N(q,c)\right)\end{aligned}$$

Defining δ_D (3/3)

- Definition of δ_N (from previous slide):

$$\begin{aligned} \delta_N(B,\epsilon) &= B \\ \delta_N(B, x\cdot c) &= \phi\left(\bigcup_{q\in\delta_N(B,x)} \delta_N(q,c)\right) \end{aligned}$$

- Definition of $\delta_\dagger$ (from previous slide):

$$\delta_\dagger(B,c) = \phi\left(\bigcup_{q\in\delta_N(B,x)} \delta_N(q,c)\right)$$

- Thus,

$$\begin{aligned} \delta_N(B,\epsilon) &= B \\ \delta_N(B, x\cdot c) &= \delta_\dagger(\delta_N(B,x),c) \end{aligned}$$

- Now, we note that B is a set of states of N, i.e. a subset of Q_N and c is a symbol in Σ. Thus, $\delta_\dagger : (2^{Q_N} \times \Sigma) \rightarrow 2^{Q_n}$. In fact, we will show that $\delta_\dagger$ is the function δ_D that we are looking for.

Constructing an Equivalent DFA

- Let $N = (Q_N, \Sigma, \delta_N, q_{0,N}, F_N)$ be an NFA.
- Let
$$\begin{aligned} Q_D &= 2^Q \\ \delta_D &= \delta_\dagger, && \text{see slide 16} \\ q_{0,D} &= \phi(\{q_{0,N}\}), && \phi \text{ defined on slide 9} \\ F_D &= \{B \subseteq Q_N \mid (B \cap F_N) \neq \emptyset\} \end{aligned}$$
- Let $M = (Q_D, \Sigma, \delta_D, q_{0,D}, F_D)$ be a DFA.
- Claim: $L(M) = L(N)$.

Proof that $L(M) = L(N)$

- Let w be a string in $\Sigma*$.
- I'll show that $w \in L(M)$ iff $w \in L(N)$.

	$w \in L(N)$	
$\Leftrightarrow$	$(\delta_N(\phi(\{q_{0,N}\}), w) \cap F_N) \neq \emptyset,$	def. NFA accept, slide 10
$\Leftrightarrow$	$\delta_N(q_{0,D}, w) \in F_D,$	def. $q_{0,D}$ and F_D, slide 17
$\Leftrightarrow$	$\delta_D(q_{0,D}, w) \in F_D,$	def. δ_D slides 17 and 16
$\Leftrightarrow$	$w \in L(M)$	def. DFA accept, Sept. 13 lecture notes

Comments about the proof:

- A proof starts by identifying the key idea(s) that show why the claim is true. In this example, the critical observation was that the power set of Q_N could be used as Q_D.
- Once the observation was made, we had to work out how to use it. This was the effort for defining $\delta_\dagger$, δ_D, $q_{0,D}$, and F_D. While I used quite a few slides explaining how I came up with these, these explanations aren't part of the final proof. The formal proof can just state the definitions (as I did on slide 17). In fact, adding lots of intuitive explanation clutters a formal proof and should be avoided. On the other hand, textbooks and instructors should provide the intuition before writing the formal proof, or the formal proof seems magical, hard to understand, and hard to see how you could come up with something similar.
- Once the definitions were made, the actual proof was very short and simple.