COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

February 2, 2023

Lecture 10: Pumping Lemma: Examples

Converse of Pumping Lemma

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^n b^n\}}^{L_1} \cup \overbrace{\{c^k w \mid k \neq 1, w \text{ starts with } a \text{ or } b\ \}}^{L_2}$$

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^nb^n\}}^{L_1} \cup \overbrace{\{c^kw \mid k \neq 1, w \text{ starts with } a \text{ or } b\ \}}^{L_2}$$

- L_2 is regular (why?)

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^n b^n\}}^{L_1} \cup \overbrace{\{c^k w \mid k \neq 1, w \text{ starts with } a \text{ or } b\ \}}^{L_2}$$

- L_2 is regular (why?)
- L_1 is not regular (why??)

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^n b^n\}}^{L_1} \cup \overbrace{\{c^k w \mid k \neq 1, w \text{ starts with } a \text{ or } b\ \}}^{L_2}$$

- L_2 is regular (why?)
- L_1 is not regular (why??)
- For L_1, the long enough word has to be ca^n.

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^n b^n\}}^{L_1} \cup \overbrace{\{c^k w \mid k \neq 1, w \text{ starts with } a \text{ or } b\ \}}^{L_2}$$

- L_2 is regular (why?)
- L_1 is not regular (why??)
- For L_1, the long enough word has to be ca^n.
- But then, consider the partition of $w = xyz$ where $x = \varepsilon, y = c$

Converse of Pumping Lemma

Example

Consider

$$L = \overbrace{\{ca^nb^n\}}^{L_1} \cup \overbrace{\{c^k w \mid k \neq 1, w \text{ starts with } a \text{ or } b\}}^{L_2}$$

- L_2 is regular (why?)
- L_1 is not regular (why??)
- For L_1, the long enough word has to be ca^n.
- But then, consider the partition of $w = xyz$ where $x = \varepsilon, y = c$
- Then if you pump up,i.e., $c^k a^n b^n \in L_2 \subseteq L$
- And if you pump down, i.e., $c^0 a^n b^n = a^n b^n \in L_2 \subseteq L$

A more refined pumping lemma

We have been looking for evidence of bad pumping in prefixes of words.
We can look for such evidence for any subword of length greater than n.

A more refined pumping lemma

We have been looking for evidence of bad pumping in prefixes of words. We can look for such evidence for any subword of length greater than n.

Refined contrapositive to Pumping Lemma

for each n
there exist words x, y, z such that $xyz \in L$ and $|y| \geq n$,
for each breakup of y into three words uvw such that $v \neq \epsilon$, then
there is a $i \geq 0$ such that $xuv^iwz \notin L$.
$\implies L \subseteq \Sigma^*$ is not a regular language

A more refined pumping lemma

We have been looking for evidence of bad pumping in prefixes of words.
We can look for such evidence for any subword of length greater than n.

Refined contrapositive to Pumping Lemma

for each n
there exist words x, y, z such that $xyz \in L$ and $|y| \geq n$,
for each breakup of y into three words uvw such that $v \neq \epsilon$, then
there is a $i \geq 0$ such that $xuv^iwz \notin L$.
$\implies L \subseteq \Sigma^*$ is not a regular language

In our earlier version of pumping lemma, $x, z = \epsilon$
Exercise: Does this help avoid the example in the previous slide?

Pumping Lemma for regular languages

Refined Pumping Lemma

$L \subseteq \Sigma^*$ is a regular language $\implies$
there exists $n \geq 1$ such that
for all strings x, y, z such that $xyz \in L$ and $|y| \geq n$
there exists a breakup of y in to $u, v, w \in \Sigma^*$ with $y = uvw$, $v \neq \epsilon$, such that
for all $i \geq 0$ we have that
$xuv^iwz \in L$.

Pumping Lemma for regular languages

Refined Pumping Lemma

$L \subseteq \Sigma^*$ is a regular language $\implies$
there exists $n \geq 1$ such that
for all strings x, y, z such that $xyz \in L$ and $|y| \geq n$
there exists a breakup of y in to $u, v, w \in \Sigma^*$ with $y = uvw$, $v \neq \epsilon$, such that
for all $i \geq 0$ we have that
$xuv^iwz \in L$.

Refined contrapositive to Pumping Lemma

for each n
there exist words x, y, z such that $xyz \in L$ and $|y| \geq n$,
for each breakup of y into three words uvw such that $v \neq \epsilon$, then
there is a $i \geq 0$ such that $xuv^iwz \notin L$.
$\implies L \subseteq \Sigma^*$ is not a regular language

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

Games with the Demon

You: Want to show L is non-regular

Demon: Wants to show that L is regular

- **Demon** picks n.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

- Demon picks n

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

- Demon picks n
- You pick $x = a^n, y = b^n, z = \varepsilon$

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

- Demon picks n
- You pick $x = a^n, y = b^n, z = \varepsilon$
- Demon picks $u = b^j, v = b^r, z = b^e$, such that $n = j + r + e$, $r > 0$.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

- Demon picks n
- You pick $x = a^n, y = b^n, z = \varepsilon$
- Demon picks $u = b^j, v = b^r, z = b^e$, such that $n = j + r + e$, $r > 0$.
- No matter what j, r, e, $i = 2$ will make you win.

Games with the Demon

You: Want to show L is non-regular
Demon: Wants to show that L is regular

- **Demon** picks n.
- **You** pick x, y, z such that $xyz \in L$, $|y| \geq n$.
- The demon picks u, v, w such that $y = uvw$ and $v \neq \varepsilon$.
- You pick $i \geq 0$.

Pumping Lemma says that if L is indeed not regular, you always have a winning strategy.

$$L = \{a^p b^q \mid p \geq q\}$$

- Demon picks n
- You pick $x = a^n, y = b^n, z = \varepsilon$
- Demon picks $u = b^j, v = b^r, z = b^e$, such that $n = j + r + e$, $r > 0$.
- No matter what j, r, e, $i = 2$ will make you win.

$$\begin{aligned} xuv^2wz &= a^n b^j b^{2r} b^e \\ &= a^n b^{j+2r+e} \\ &= a^n b^{n+r} \end{aligned}$$

Unary Languages

- Any language of the form $L \subseteq \{a\}^*$ is just a subset of $\mathbb{N}$

Unary Languages

- Any language of the form $L \subseteq \{a\}^*$ is just a subset of $\mathbb{N}$

$$L_{prime} := \{a^p \mid p \ \text{ prime } \}$$

Unary Languages

- Any language of the form $L \subseteq \{a\}^*$ is just a subset of $\mathbb{N}$

$$L_{prime} := \{a^p \mid p \ \text{ prime } \}$$

- $U \subseteq \mathbb{N}$ is said to be **Ultimately Periodic** if there exists $n \geq 0$ and $p > 0$ such that for all $m \geq n$ $m \in U$ if and only if $m + p \in U$. p is the period of U.

Unary Languages

- Any language of the form $L \subseteq \{a\}^*$ is just a subset of $\mathbb{N}$

$$L_{prime} := \{a^p \mid p \text{ prime } \}$$

- $U \subseteq \mathbb{N}$ is said to be **Ultimately Periodic** if there exists $n \geq 0$ and $p > 0$ such that for all $m \geq n$ $m \in U$ if and only if $m + p \in U$. p is the period of U.

$$\{0, 3, 7, 11, 19, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, \dots\}.$$

Unary Languages

- Any language of the form $L \subseteq \{a\}^*$ is just a subset of $\mathbb{N}$

$$L_{prime} := \{a^p \mid p \ \text{ prime } \}$$

- $U \subseteq \mathbb{N}$ is said to be **Ultimately Periodic** if there exists $n \geq 0$ and $p > 0$ such that for all $m \geq n$ $m \in U$ if and only if $m + p \in U$. p is the period of U.

$$\{0, 3, 7, 11, 19, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, \ldots\}.$$

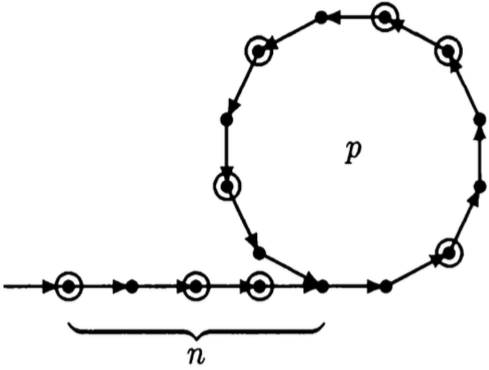
Theorem

$L \subseteq \{a\}^*$ is regular if and only if $\{m \mid a^m \in L\}$ is ultimately periodic.

Unary Languages and Periodicity

Theorem

$L \subseteq \{a\}^*$ is regular if and only if $\{m \mid a^m \in L\}$ is ultimately periodic.



Corollary

Let $L \subseteq \Sigma^*$. Then the set

Unary Languages and Periodicity

Theorem

$L \subseteq \{a\}^*$ is regular if and only if $\{m \mid a^m \in L\}$ is ultimately periodic.

$$\{0, 3, 7, 11, 19, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, \dots\}$$

Corollary

Let $L \subseteq \Sigma^*$. Then the set

$$Lengths(L) := \{|x| \mid x \in L\}$$

is ultimately periodic.

Consider the homomorphism $h : \Sigma^* \to \{a\}$ by $h(b) = a$ for every $b \in \Sigma$

Unary Languages and Periodicity

Theorem

$L \subseteq \{a\}^*$ is regular if and only if $\{m \mid a^m \in L\}$ is ultimately periodic.

$$\{0, 3, 7, 11, 19, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, \dots\}$$

$\square$

Corollary

Let $L \subseteq \Sigma^*$. Then the set

$$Lengths(L) := \{|x| \mid x \in L\}$$

is ultimately periodic.

A Nice Corollary

Theorem

$L \subseteq \{a\}^*$ is regular if and only if $\{m \mid a^m \in L\}$ is ultimately periodic.

Corollary

Let $L \subseteq \Sigma^*$. Then the set

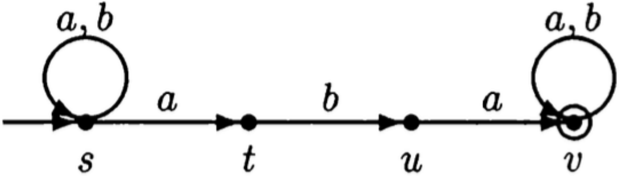
$$Lengths(L) := \{|x| \mid x \in L\}$$

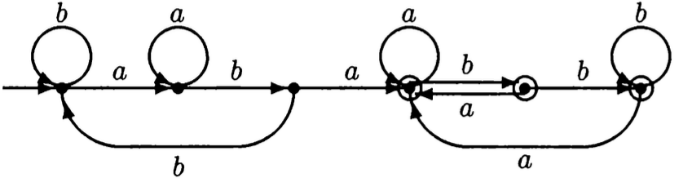
is ultimately periodic.

Consider the homomorphism $h : \Sigma^* \to \{a\}$ by $h(b) = a$ for every $b \in \Sigma$

DFA Minimization

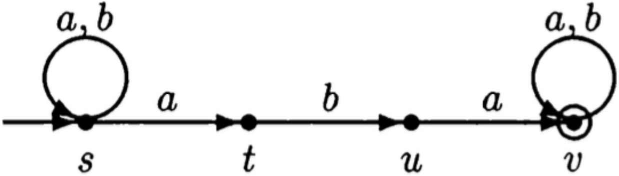
DFA Minimization : Example 1

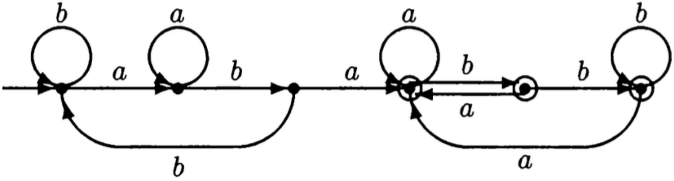


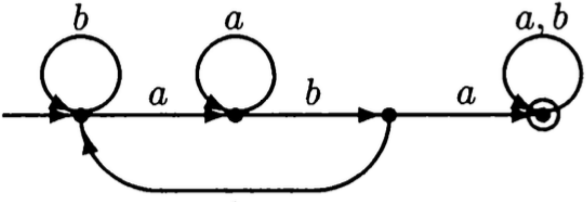


DFA Minimization

DFA Minimization : Example 1







DFA Minimization

- Equivalence of states was obvious in the example, but is it generally the case?

DFA Minimization

- Equivalence of states was obvious in the example, but is it generally the case?
- Every regular language has a unique minimal DFA (upto isomorphism)

DFA Minimization

- Equivalence of states was obvious in the example, but is it generally the case?
- Every regular language has a unique minimal DFA (upto isomorphism)
- There is an amazing completely mechanical way to do this!

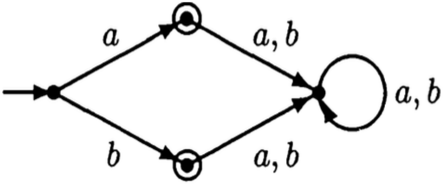
DFA Minimization

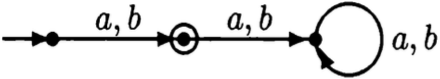
- Equivalence of states was obvious in the example, but is it generally the case?
- Every regular language has a unique minimal DFA (upto isomorphism)
- There is an amazing completely mechanical way to do this!
- Rough idea: Given $M = (Q, \Sigma, q_0, \delta, F)$
  - Get rid of inaccessible states (i.e, there does not exist $x \in \Sigma^*, \hat{\delta}(q, x) = q$.

DFA Minimization

- Equivalence of states was obvious in the example, but is it generally the case?
- Every regular language has a unique minimal DFA (upto isomorphism)
- There is an amazing completely mechanical way to do this!
- Rough idea: Given $M = (Q, \Sigma, q_0, \delta, F)$
  - Get rid of inaccessible states (i.e, there does not exist $x \in \Sigma^*, \hat{\delta}(q, x) = q$.
  - Collapse "equivalent" states.

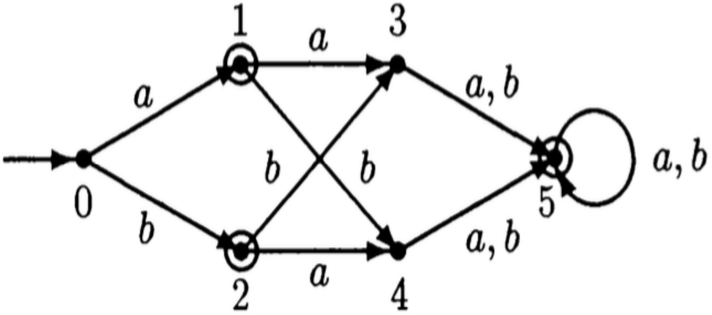
Minimization : Example 2





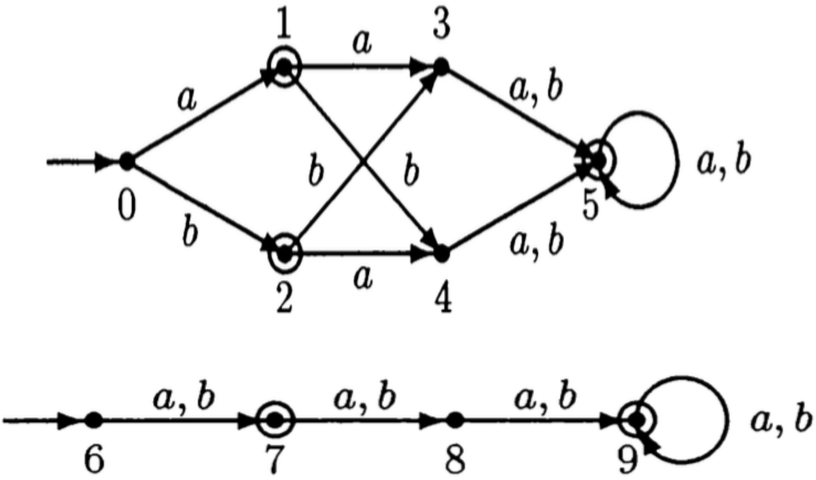
$$L = \{a, b\}$$

Minimization Example 3

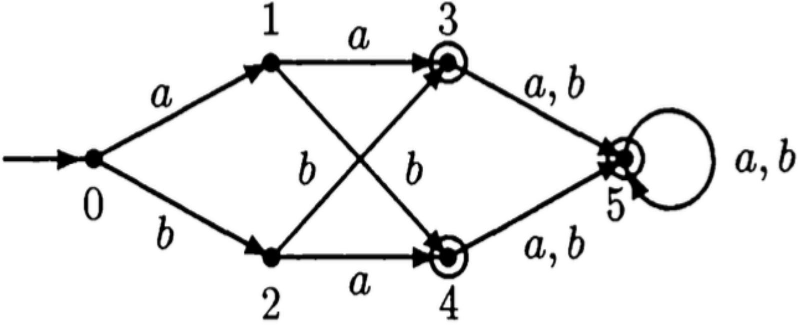


$L = \{a, b\} \cup \{\text{Strings of length } \geq 3\}$

Minimization Example 3

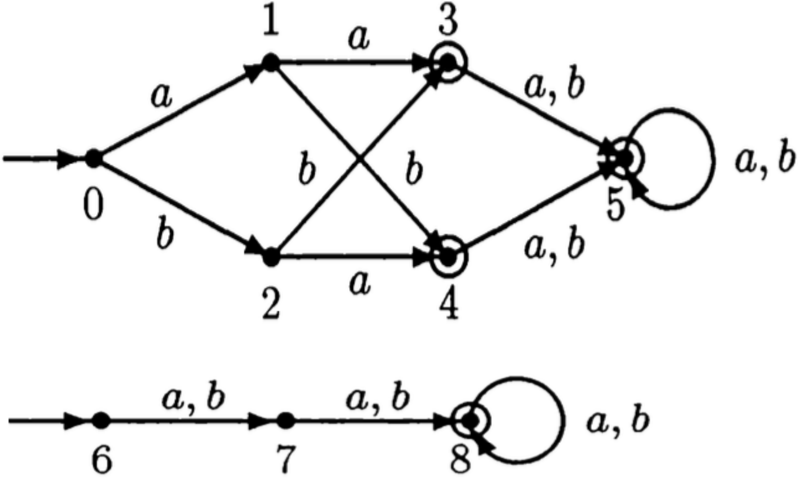


Minimization Example 4

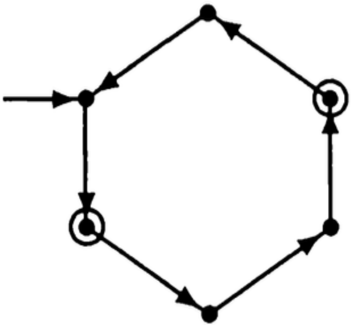


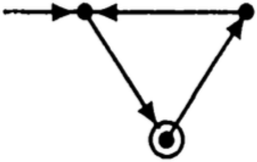
$L = \{\text{Strings of length } \geq 2\}$

Minimization Example 4



Minimization Example 5





$$L = \{a^m \mid m \equiv 1 \pmod 3\}$$

A few pressing questions

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!

   If $\hat{\delta}(s, x) = p \in F$ and $\hat{\delta}(s, y) = q \notin F$, so cannot collapse p and q!

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!
   If $\hat{\delta}(s,x) = p \in F$ and $\hat{\delta}(s,y) = q \notin F$, so cannot collapse p and q!
2. If we are collapsing p and q, better also collapse $\delta(p,a)$ and $\delta(q,a)$ for all $a \in \Sigma$.

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!
   If $\hat{\delta}(s,x) = p \in F$ and $\hat{\delta}(s,y) = q \notin F$, so cannot collapse p and q!
2. If we are collapsing p and q, better also collapse $\delta(p,a)$ and $\delta(q,a)$ for all $a \in \Sigma$.

Inductively, these two imply that we cannot collapse p and q if $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ for some string x

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!
   If $\hat{\delta}(s, x) = p \in F$ and $\hat{\delta}(s, y) = q \notin F$, so cannot collapse p and q!
2. If we are collapsing p and q, better also collapse $\delta(p, a)$ and $\delta(q, a)$ for all $a \in \Sigma$.

Inductively, these two imply that we cannot collapse p and q if $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ for some string x Turns out this is necessary and sufficient to decide if a pair of states can be collapsed or not!