DFA Closure Properties

Arjun Chandrasekhar

Design a DFA

- Let $\Sigma = \{a, b\}$

Design a DFA

- Let $\Sigma = \{a, b\}$
- Let
  $\mathrm{L} = \{\mathrm{w} \mid \mathrm{w} \text{ contains exactly two a's}\}$

Design a DFA

- Let $\Sigma = \{a, b\}$
- Let
  L = {w | w contains exactly two a's}
- Then the *complement* of L is
  L^c = {w | w *doesn't* contain exactly two a's}

Design a DFA

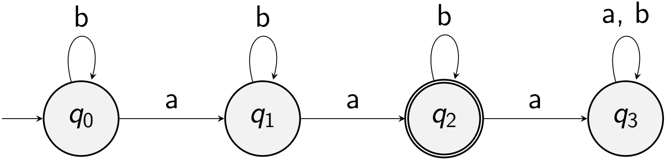
- Let $\Sigma = \{a, b\}$
- Let
  L = {w | w contains exactly two a's}
- Then the *complement* of L is
  L^c = {w | w *doesn't* contain exactly two a's}
- Let's design DFAs to recognize L and L^c

Design a DFA

$$L = \{w \mid w \text{ contains exactly two a's}\}$$

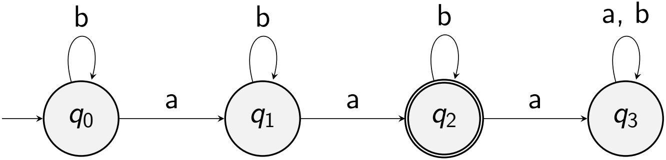
Design a DFA

$$L = \{w \mid w \text{ contains exactly two a's}\}$$



Design a DFA

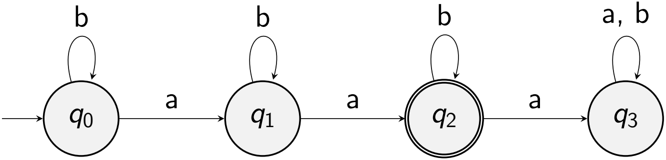
L = {w | w contains exactly two a's}



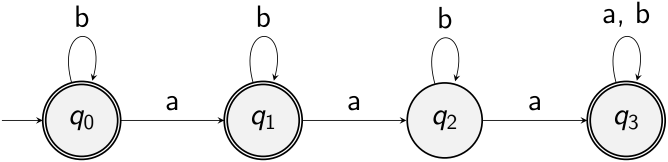
L = {w | w *doesn't* contain exactly two a's}

Design a DFA

L = {w | w contains exactly two a's}

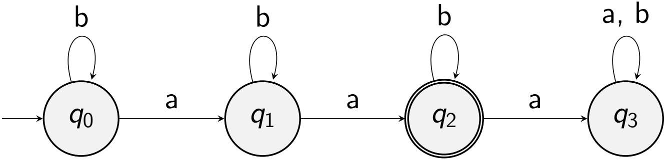


L = {w | w *doesn't* contain exactly two a's}

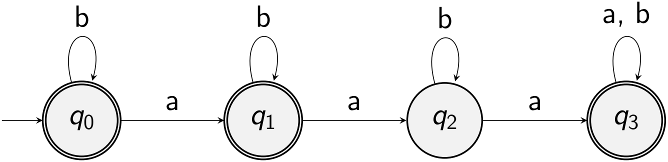


Design a DFA

$$L = \{w \mid w \text{ contains exactly two a's}\}$$

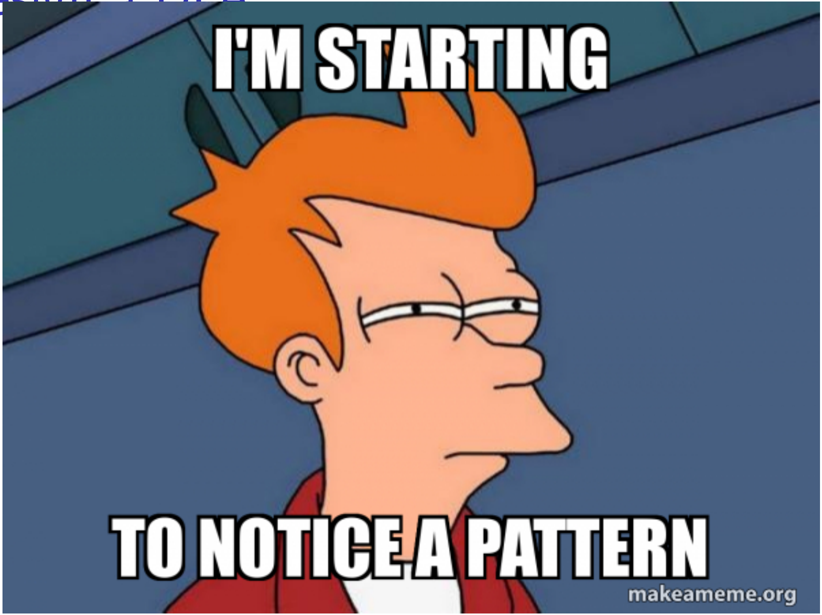


$$L = \{w \mid w \textit{ doesn't} \text{ contain exactly two a's}\}$$



Notice a pattern?

Design a DFA



Closure of regular languages under complement

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?
  - L is regular...

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?
  - L is regular...
  - and thus it is recognized by a DFA

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?
  - L is regular...
  - and thus it is recognized by a DFA
- What do we want to show for L^c?

Closure of regular languages under complement

Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?
  - L is regular...
  - and thus it is recognized by a DFA
- What do we want to show for L^c?
  - That L^c is also regular...

Closure of regular languages under complement

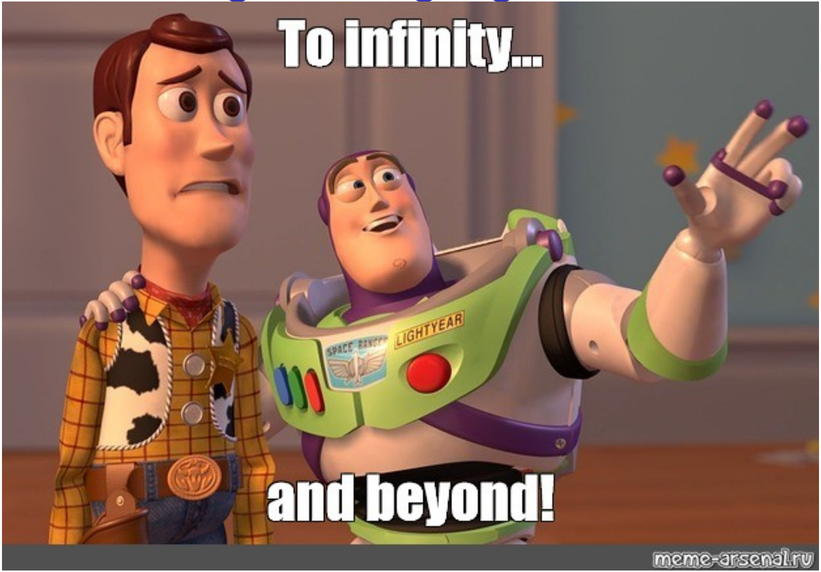
Proposition: Regular languages are closed under complement

- In other words, if L is a regular language, then we claim that L^c must also be regular
- What do we know about L?
  - L is regular...
  - and thus it is recognized by a DFA
- What do we want to show for L^c?
  - That L^c is also regular...
  - i.e., there is also a DFA to recognize L^c

Closure of regular languages under complement

Technique: Go through every single regular language one by one, and show that its complement is also regular

Closure of regular languages under co



Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

- **Proof idea:** Since L is regular, there must be a DFA D that recognizes L.

Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

- **Proof idea:** Since L is regular, there must be a DFA D that recognizes L.
- We will use this to construct a DFA D^c that recognizes L^c.

Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

- **Proof idea:** Since L is regular, there must be a DFA D that recognizes L.
- We will use this to construct a DFA D^c that recognizes L^c.
- This is actually quite simple!

Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

- **Proof idea:** Since L is regular, there must be a DFA D that recognizes L.
- We will use this to construct a DFA D^c that recognizes L^c.
- This is actually quite simple!
  - We simply start with D, and then we flip the accept and reject states.

Closure of regular languages under complement

Technique: Use the formal definition of a DFA to construct the complement DFA

- **Proof idea:** Since L is regular, there must be a DFA D that recognizes L.
- We will use this to construct a DFA D^c that recognizes L^c.
- This is actually quite simple!
  - We simply start with D, and then we flip the accept and reject states.
- Now let's try to give an airtight proof!

Closure of regular languages under complement

- Suppose L is regular

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c
  - $Q^c = Q$ (same states)

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c
  - $Q^c = Q$ (same states)
  - $\Sigma^c = \Sigma$ (same alphabet)

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c
  - $Q^c = Q$ (same states)
  - $\Sigma^c = \Sigma$ (same alphabet)
  - $\delta^c = \delta$ (same transitions)

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c
  - $Q^c = Q$ (same states)
  - $\Sigma^c = \Sigma$ (same alphabet)
  - $\delta^c = \delta$ (same transitions)
  - $q_s^c = q_s$ (same start state)

Closure of regular languages under complement

- Suppose L is regular
- There is a DFA $D = (Q, \Sigma, \delta, q_s, F)$ that recognizes L
- We will construct a DFA $D^c = (Q^c, \Sigma^c, \delta^c, q_s^c, F^c)$ to recognize L^c
  - $Q^c = Q$ (same states)
  - $\Sigma^c = \Sigma$ (same alphabet)
  - $\delta^c = \delta$ (same transitions)
  - $q_s^c = q_s$ (same start state)
  - $\mathrm{F^c = Q \setminus F}$ (flip the accept/reject states)

Reversal of a language

Reversal of a language

- Let Σ be an alphabet, $L \subseteq \Sigma^*$ be a formal language

Reversal of a language

- Let Σ be an alphabet, $L \subseteq \Sigma^*$ be a formal language
- Let $w \in \Sigma^*$ be a string. Then w^r is the *reversal* of w, i.e. all the characters of w backwards

Reversal of a language

- Let Σ be an alphabet, $L \subseteq \Sigma^*$ be a formal language
- Let $w \in \Sigma^*$ be a string. Then w^r is the *reversal* of w, i.e. all the characters of w backwards
- $L^r = \{w^r | w \in L\}$ is the *reversal* of L, i.e. the backwards version of all the strings in L

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$
  - $L^r = \{\text{w} \mid \text{w starts with b and ends with a}\}$

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$
  - $L^r = \{\text{w} \mid \text{w starts with b and ends with a}\}$
- $L = \{\text{w} \mid \text{w contains aab}\}$

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$
  - $L^r = \{\text{w} \mid \text{w starts with b and ends with a}\}$
- $L = \{\text{w} \mid \text{w contains aab}\}$
  - $L^r = \{\text{w} \mid \text{w contains baa}\}$

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$
  - $L^r = \{\text{w} \mid \text{w starts with b and ends with a}\}$
- $L = \{\text{w} \mid \text{w contains aab}\}$
  - $L^r = \{\text{w} \mid \text{w contains baa}\}$
- $L = \{\text{w} \mid \text{w is an even integer}\}$

Reversal of a language

- $L = \{\text{w} \mid \text{w starts with a and ends with b}\}$
  - $L^r = \{\text{w} \mid \text{w starts with b and ends with a}\}$
- $L = \{\text{w} \mid \text{w contains aab}\}$
  - $L^r = \{\text{w} \mid \text{w contains baa}\}$
- $L = \{\text{w} \mid \text{w is an even integer}\}$
  - $L^r = \{\text{w} \mid \text{w starts with 0, 2, 4, 6, 8}\}$

Reversal of a language

Let $\Sigma = \{0, 1\}$, and let $L = \{w | w \text{ starts with } 01\}$.
Which of the following strings are in L^r

A) 01

B) 1010

C) 0101

D) 1111110

Reversal of a language

Let $\Sigma = \{0, 1\}$, and let $L = \{w | w \text{ starts with } 01\}$.
Which of the following strings are in L^r

A) 01

B) 1010 ✓

C) 0101

D) 1111110 ✓

Reversal of a language

- Let $\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \ldots, \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} \right\}$

Reversal of a language

- Let $\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \ldots, \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} \right\}$
- Let $B = \{w \in \Sigma^* \mid$ the top row $+$ the middle row $=$ the bottom row$\}$

Reversal of a language

- Let $\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \ldots, \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} \right\}$
- Let $B = \{w \in \Sigma^* \mid \text{the top row} + \text{the middle row} = \text{the bottom row}\}$
- $\begin{bmatrix} 4 \\ 3 \\ 7 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 6 \end{bmatrix} \in B$: $425 + 301 = 726$

Reversal of a language

- Let $\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \ldots, \begin{bmatrix} 9 \\ 9 \\ 9 \end{bmatrix} \right\}$
- Let $B = \{w \in \Sigma^* \mid$ the top row + the middle row = the bottom row$\}$
- $\begin{bmatrix} 4 \\ 3 \\ 7 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 6 \end{bmatrix} \in B$: $425 + 301 = 726$
- $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 0 \end{bmatrix} \notin B$: $119 + 041 \neq 150$

Reversal of a language

Let $B = \{w \in \Sigma^* \mid$ the top row + the middle row = the bottom row$\}$

Which of the following strings are in B?

A. $\begin{bmatrix}5\\4\\9\end{bmatrix} \begin{bmatrix}5\\4\\9\end{bmatrix} \begin{bmatrix}5\\4\\9\end{bmatrix}$

C. $\begin{bmatrix}9\\1\\0\end{bmatrix} \begin{bmatrix}9\\1\\0\end{bmatrix} \begin{bmatrix}9\\1\\0\end{bmatrix}$

B. $\begin{bmatrix}0\\7\\8\end{bmatrix} \begin{bmatrix}3\\7\\0\end{bmatrix} \begin{bmatrix}1\\1\\2\end{bmatrix}$

D. $\begin{bmatrix}1\\2\\3\end{bmatrix} \begin{bmatrix}4\\5\\6\end{bmatrix} \begin{bmatrix}7\\8\\9\end{bmatrix}$

Reversal of a language

Let $B = \{w \in \Sigma^* \mid$ the top row + the middle row = the bottom row$\}$

Which of the following strings are in B?

A. $\begin{bmatrix}5\\4\\9\end{bmatrix}\begin{bmatrix}5\\4\\9\end{bmatrix}\begin{bmatrix}5\\4\\9\end{bmatrix}$ ✓

B. $\begin{bmatrix}0\\7\\8\end{bmatrix}\begin{bmatrix}3\\7\\0\end{bmatrix}\begin{bmatrix}1\\1\\2\end{bmatrix}$ ✓

C. $\begin{bmatrix}9\\1\\0\end{bmatrix}\begin{bmatrix}9\\1\\0\end{bmatrix}\begin{bmatrix}9\\1\\0\end{bmatrix}$

D. $\begin{bmatrix}1\\2\\3\end{bmatrix}\begin{bmatrix}4\\5\\6\end{bmatrix}\begin{bmatrix}7\\8\\9\end{bmatrix}$

Reversal of a language

- $B^r = \{w^r | w \in B\}$

Reversal of a language

- $B^r = \{w^r | w \in B\}$
- That is, $B^r = \{w \in \Sigma^* \mid$ the top row reversed $+$ the middle row reversed $=$ the bottom row reversed$\}$

Reversal of a language

- $B^r = \{w^r | w \in B\}$
- That is, $B^r = \{w \in \Sigma^* \mid$ the top row reversed $+$ the middle row reversed $=$ the bottom row reversed$\}$
- $\begin{bmatrix} 5 \\ 1 \\ 6 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 7 \end{bmatrix} \in B^r$: $425 + 301 = 726$

Reversal of a language

- $B^r = \{w^r | w \in B\}$
- That is, $B^r = \{w \in \Sigma^* \mid$ the top row reversed $+$ the middle row reversed $=$ the bottom row reversed$\}$
- $\begin{bmatrix}5\\1\\6\end{bmatrix} \begin{bmatrix}2\\0\\2\end{bmatrix} \begin{bmatrix}4\\3\\7\end{bmatrix} \in B^r$: $425 + 301 = 726$
- $\begin{bmatrix}9\\1\\0\end{bmatrix} \begin{bmatrix}1\\4\\5\end{bmatrix} \begin{bmatrix}1\\0\\1\end{bmatrix} \notin B^r$: $119 + 041 \neq 150$

Reversal of a language

$B^r = \{w \in \Sigma^* \mid$ the top row reversed + the middle row reversed = the bottom row reversed$\}$

Which of the following strings are in B^r?

A. $\begin{bmatrix}5\\4\\9\end{bmatrix} \begin{bmatrix}5\\4\\9\end{bmatrix} \begin{bmatrix}5\\4\\9\end{bmatrix}$

C. $\begin{bmatrix}2\\8\\0\end{bmatrix} \begin{bmatrix}6\\3\\0\end{bmatrix} \begin{bmatrix}7\\1\\9\end{bmatrix}$

B. $\begin{bmatrix}0\\7\\8\end{bmatrix} \begin{bmatrix}3\\7\\0\end{bmatrix} \begin{bmatrix}1\\1\\2\end{bmatrix}$

D. $\begin{bmatrix}9\\8\\7\end{bmatrix} \begin{bmatrix}6\\5\\2\end{bmatrix} \begin{bmatrix}1\\2\\4\end{bmatrix}$

Reversal of a language

$B^r = \{w \in \Sigma^* \mid$ the top row reversed + the middle row reversed = the bottom row reversed$\}$

Which of the following strings are in B^r?

A. $\begin{bmatrix}5\\4\\9\end{bmatrix}\begin{bmatrix}5\\4\\9\end{bmatrix}\begin{bmatrix}5\\4\\9\end{bmatrix}$ ✓

C. $\begin{bmatrix}2\\8\\0\end{bmatrix}\begin{bmatrix}6\\3\\0\end{bmatrix}\begin{bmatrix}7\\1\\9\end{bmatrix}$ ✓

B. $\begin{bmatrix}0\\7\\8\end{bmatrix}\begin{bmatrix}3\\7\\0\end{bmatrix}\begin{bmatrix}1\\1\\2\end{bmatrix}$

D. $\begin{bmatrix}9\\8\\7\end{bmatrix}\begin{bmatrix}6\\5\\2\end{bmatrix}\begin{bmatrix}1\\2\\4\end{bmatrix}$ ✓

Reversal of a language

Let's prove B^r is a regular language

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r
- **Proof idea:** construct at DFA that goes column by column, performs addition on that column

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r
- **Proof idea:** construct at DFA that goes column by column, performs addition on that column
  - Use the states to keep track of the carry

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r
- **Proof idea:** construct at DFA that goes column by column, performs addition on that column
  - Use the states to keep track of the carry
  - If at any point top row + middle row + carry $\neq$ bottom row, we move to a reject state and loop

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r
- **Proof idea:** construct at DFA that goes column by column, performs addition on that column
  - Use the states to keep track of the carry
  - If at any point top row $+$ middle row $+$ carry $\neq$ bottom row, we move to a reject state and loop
  - Otherwise if we reach the end of the input and carry $= 0$, we accept

Reversal of a language

Let's prove B^r is a regular language

- What do we want to show?
  - Want to show there is a DFA D^r to recognize B^r
- **Proof idea:** construct at DFA that goes column by column, performs addition on that column
  - Use the states to keep track of the carry
  - If at any point top row + middle row + carry $\neq$ bottom row, we move to a reject state and loop
  - Otherwise if we reach the end of the input and carry $= 0$, we accept
- **Proof:** see board

Reversal of a language

Proposition: Regular languages are closed under reversal

Reversal of a language

Proposition: Regular languages are closed under reversal

- That is, if L is regular, then L^r is regular

Reversal of a language

Proposition: Regular languages are closed under reversal

- That is, if L is regular, then L^r is regular
- You will prove this on a future homework

Perfect shuffle

Perfect shuffle

- Let A, B be regular languages

Perfect shuffle

- Let A, B be regular languages
- The *perfect shuffle* of A and B is $\mathrm{L} = \{\mathrm{w} \mid \mathrm{w} = a_1 b_1 a_2 b_2 \ldots a_n b_n$ where $a_1 a_2 \ldots a_n \in A$ and $b_1 b_2 \ldots b_n \in B\}$

Perfect shuffle

- Let A, B be regular languages
- The *perfect shuffle* of A and B is $\mathrm{L} = \{\mathrm{w} \mid \mathrm{w} = a_1 b_1 a_2 b_2 \dots a_n b_n$ where $a_1 a_2 \dots a_n \in A$ and $b_1 b_2 \dots b_n \in B\}$
- The odd characters form a string in A

Perfect shuffle

- Let A, B be regular languages
- The *perfect shuffle* of A and B is $\mathrm{L} = \{\mathrm{w} \mid \mathrm{w} = a_1 b_1 a_2 b_2 \ldots a_n b_n$ where $a_1 a_2 \ldots a_n \in A$ and $b_1 b_2 \ldots b_n \in B\}$
- The odd characters form a string in A
- The even characters form a string in B

Perfect shuffle example

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)

Perfect shuffle example

- Let $A = \{0, 00, 000, \dots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \dots\}$ (i.e. all 1's, no 0's)

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \ldots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \ldots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$

Perfect shuffle example

- Let $A = \{0, 00, 000, \dots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \dots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$
  - $111 \in B$

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \ldots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$
  - $111 \in B$
- $010100 \notin \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \ldots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$
  - $111 \in B$
- $010100 \notin \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$

Perfect shuffle example

- Let $A = \{0, 00, 000, \ldots\}$ (i.e. all 0's, no 1's)
- Let $B = \{1, 11, 111, \ldots\}$ (i.e. all 1's, no 0's)
- $010101 \in \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$
  - $111 \in B$
- $010100 \notin \mathrm{PERFECT\text{-}SHUFFLE(A, B)}$
  - $000 \in A$
  - $110 \notin \mathrm{B}$

Perfect shuffle example

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $\text{PERFECT-SHUFFLE}(A, B)$?

A) aababaaa
B) bababab b
C) baabbaabbb
D) aabab

Perfect shuffle example

Let $\Sigma = \{a, b\}$.
Let $A = \{w|w \text{ has an even number of a's}\}$
Let $B = \{w|w \text{ ends with b}\}$
Which of the following strings are in $\text{PERFECT-SHUFFLE}(A, B)$?

A) aababaaa
B) babababb ✓
C) baabbaabbb ✓
D) aabab

Perfect shuffle closure

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation
  - This means that if A and B are regular, then $\text{PERFECT-SHUFFLE}(A, B)$ is regular

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation
  - This means that if A and B are regular, then $\text{PERFECT-SHUFFLE}(A, B)$ is regular
  - What do we know about A and B?

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation
  - This means that if A and B are regular, then $\text{PERFECT-SHUFFLE}(A, B)$ is regular
  - What do we know about A and B?
    - There exist DFAs D_A and D_B to recognize A and B, respectively

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation
  - This means that if A and B are regular, then $\text{PERFECT-SHUFFLE}(A, B)$ is regular
  - What do we know about A and B?
    - There exist DFAs D_A and D_B to recognize A and B, respectively
  - What do we want to show for $\text{PERFECT-SHUFFLE}(A, B)$?

Perfect shuffle closure

- **Proposition:** Regular languages are closed under the perfect shuffle operation
  - This means that if A and B are regular, then $\text{PERFECT-SHUFFLE}(A, B)$ is regular
  - What do we know about A and B?
    - There exist DFAs D_A and D_B to recognize A and B, respectively
  - What do we want to show for $\text{PERFECT-SHUFFLE}(A, B)$?
    - There exists a DFA D that recognizes $\text{PERFECT-SHUFFLE}(A, B)$

Perfect shuffle

- **Proposition:** Regular languages are closed under the perfect shuffle operation

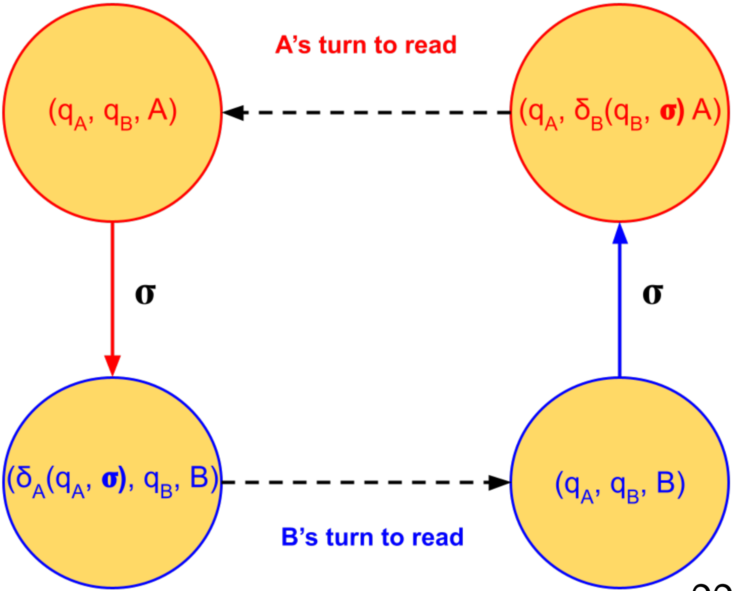
Perfect shuffle

- **Proposition:** Regular languages are closed under the perfect shuffle operation
- **Proof idea:** Using D_A and D_B, we will construct a DFA runs the two machines and checks if A accepts the odd characters and B accepts the even characters

Perfect shuffle

- **Proposition:** Regular languages are closed under the perfect shuffle operation
- **Proof idea:** Using D_A and D_B, we will construct a DFA runs the two machines and checks if A accepts the odd characters and B accepts the even characters
- **Technique:** Run two DFAs *in alternation* using Cartesian product, and an extra variable to keep track of turns

Perfect shuffle idea



Perfect shuffle closure

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
- $\delta((q_A, q_B, B), \sigma) = (q_A, \delta_B(q_B, \sigma), A)$

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
- $\delta((q_A, q_B, B), \sigma) = (q_A, \delta_B(q_B, \sigma), A)$
- $q_S = (q_{s_A}, q_{s_B}, A)$

Perfect shuffle closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
- $\delta((q_A, q_B, B), \sigma) = (q_A, \delta_B(q_B, \sigma), A)$
- $q_S = (q_{s_A}, q_{s_B}, A)$
- $F = F_A \times F_B \times \{A\}$

Perfect shuffle closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$

Perfect shuffle closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- Each state is a combination of 3 elements:

Perfect shuffle closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- Each state is a combination of 3 elements:
  - A state $q_A \in Q_A$

Perfect shuffle closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- Each state is a combination of 3 elements:
  - A state $q_A \in Q_A$
  - A state $q_B \in Q_B$

Perfect shuffle closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $Q = Q_A \times Q_B \times \{A, B\}$
- Each state is a combination of 3 elements:
  - A state $q_A \in Q_A$
  - A state $q_B \in Q_B$
  - A variable A or B that keeps track of turns

Perfect shuffle closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$

Perfect shuffle closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
  - When it's A's turn, we transition A's state, keep B's state the same, and switch to B's turn to read

Perfect shuffle closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
  - When it's A's turn, we transition A's state, keep B's state the same, and switch to B's turn to read
- $\delta((q_A, q_B, B), \sigma) = (q_A, \delta_B(q_B, \sigma), A)$

Perfect shuffle closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $\delta((q_A, q_B, A), \sigma) = (\delta_A(q_A, \sigma), q_B, B)$
  - When it's A's turn, we transition A's state, keep B's state the same, and switch to B's turn to read
- $\delta((q_A, q_B, B), \sigma) = (q_A, \delta_B(q_B, \sigma), A)$
  - When it's B's turn, we transition B's state, keep A's state the same, and switch to A's turn to read

Perfect shuffle closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $q_s = (q_{s_A}, q_{s_B}, A)$

Perfect shuffle closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $q_s = (q_{s_A}, q_{s_B}, A)$
- We start out in A's start state

Perfect shuffle closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\mathrm{PERFECT\text{-}SHUFFLE}(A, B)$

- $q_s = (q_{s_A}, q_{s_B}, A)$
- We start out in A's start state
- We start out in B's start state

Perfect shuffle closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $q_s = (q_{s_A}, q_{s_B}, A)$
- We start out in A's start state
- We start out in B's start state
- Initially, it's A's turn to read

Perfect shuffle closure - accept states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $F = F_A \times F_B \times \{A\}$

Perfect shuffle closure - accept states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $F = F_A \times F_B \times \{A\}$
- A's state should be one of its accept states.

Perfect shuffle closure - accept states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $F = F_A \times F_B \times \{A\}$
- A's state should be one of its accept states.
- B's state should also be one of its accept states

Perfect shuffle closure - accept states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

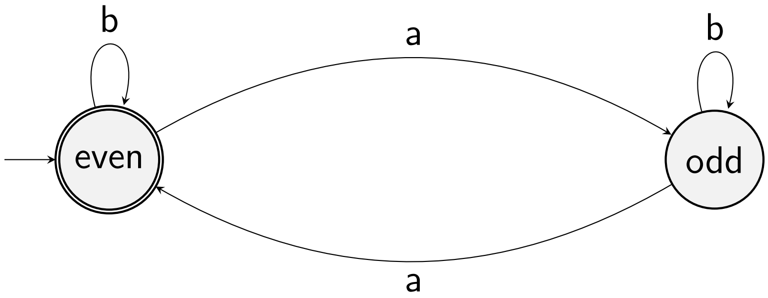
Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $\text{PERFECT-SHUFFLE}(A, B)$

- $F = F_A \times F_B \times \{A\}$
- A's state should be one of its accept states.
- B's state should also be one of its accept states
- At the end it should be A's turn to read.

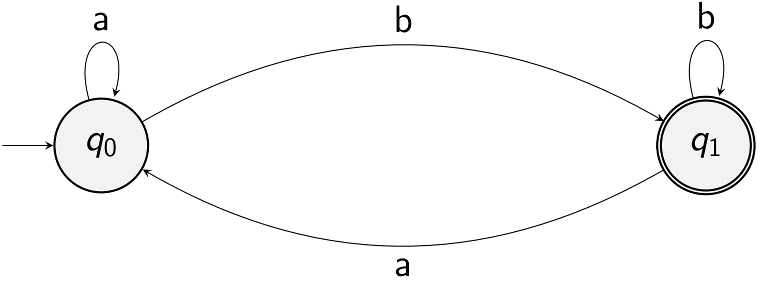
Perfect shuffle example

DFA for $A = \{w | w \text{ has an even number of a's}\}$



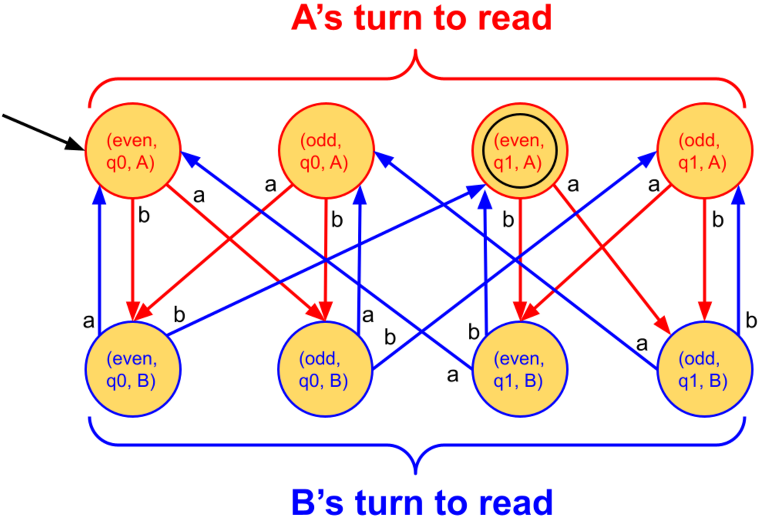
Perfect shuffle example

DFA for $B = \{w | w \text{ ends with b}\}$



Perfect shuffle example

DFA for $\text{PERFECT-SHUFFLE}(A, B)$



Regular operations

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$
  - w can be split into two substrings; the first substring is in A, the second substring is in B

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$
  - w can be split into two substrings; the first substring is in A, the second substring is in B
- **(Kleene) Star:**
  $A^* = \{\epsilon\} \cup \{w = w_1 w_2 \dots w_n | w_i \in A\}$

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$
  - w can be split into two substrings; the first substring is in A, the second substring is in B
- **(Kleene) Star:**
  $A^* = \{\epsilon\} \cup \{w = w_1 w_2 \ldots w_n | w_i \in A\}$
  - w can be split into n substrings; each substring is in A

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$
  - w can be split into two substrings; the first substring is in A, the second substring is in B
- **(Kleene) Star:**
  $A^* = \{\epsilon\} \cup \{w = w_1 w_2 \ldots w_n | w_i \in A\}$
  - w can be split into n substrings; each substring is in A
  - 0 or more “copies” of A

Regular operations

- **Union:**
  $A \cup B = \{w | w \in A \text{ or } w \in B\}$
- **Concatenation:**
  $A \circ B = \{w = w_1 w_2 | w_1 \in A, w_2 \in B\}$
  - w can be split into two substrings; the first substring is in A, the second substring is in B
- **(Kleene) Star:**
  $A^* = \{\epsilon\} \cup \{w = w_1 w_2 \ldots w_n | w_i \in A\}$
  - w can be split into n substrings; each substring is in A
  - 0 or more "copies" of A
  - Note that A^* *always* includes empty string ϵ

Union Operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $A \cup B$?

A) aaaaaa

B) baaaab

C) ab

D) aabaaba

Union Operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $A \cup B$?

A) aaaaaa ✓

B) baaaab ✓

C) ab ✓

D) aabaaba

Concatenation operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $A \circ B$?

A) aaab

B) aabaa

C) bba

D) bbbaaaa

Concatenation operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $A \circ B$?

A) aa|ab ✓

B) aabaa

C) bba

D) bbbaaaa

Concatenation operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w|w \text{ has an even number of a's}\}$
Let $B = \{w|w \text{ ends with b}\}$
Which of the following strings are in $B \circ A$?

A) aaab

B) aabaa

C) bba

D) bbbaaaa

Concatenation operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in $B \circ A$?

A) aaab| ✓

B) aab|aa ✓

C) bba

D) bbb|aaaa✓

Kleene star operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in A^*?

A) ϵ

B) aaaababab

C) aabaaa

D) bba

E) bbbaaaa

Kleene star operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w|w \text{ has an even number of a's}\}$
Let $B = \{w|w \text{ ends with b}\}$
Which of the following strings are in A^*?

A) ϵ ✓

B) aaaa|babab| ✓

C) aabaaa

D) bba

E) bbb|aaaa ✓

Kleene star operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w \text{ has an even number of a's}\}$
Let $B = \{w | w \text{ ends with b}\}$
Which of the following strings are in B^*?

A) ϵ

B) aaaababab

C) aabaaa

D) bba

E) bbbaaaa

Kleene star operation

Let $\Sigma = \{a, b\}$.
Let $A = \{w | w$ has an even number of a's$\}$
Let $B = \{w | w$ ends with b$\}$
Which of the following strings are in B^*?

A) ϵ ✓

B) aaaab|ab|ab| ✓

C) aabaaa

D) bba

E) bbbaaaa

Closure of regular languages under union

Closure of regular languages under union

Proposition: Regular languages are closed under union

Closure of regular languages under union

Proposition: Regular languages are closed under union

- This means that if L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular

Closure of regular languages under union

Proposition: Regular languages are closed under union

- This means that if L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular
- What do we know about L_1 and L_2?

Closure of regular languages under union

Proposition: Regular languages are closed under union

- This means that if L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular
- What do we know about L_1 and L_2?
  - There exist DFAs D_1, D_2 that recognizes L_1 and L_2, respectively

Closure of regular languages under union

Proposition: Regular languages are closed under union

- This means that if L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular
- What do we know about L_1 and L_2?
  - There exist DFAs D_1, D_2 that recognizes L_1 and L_2, respectively
- What do we want to show for $L_1 \cup L_2$?

Closure of regular languages under union

Proposition: Regular languages are closed under union

- This means that if L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular
- What do we know about L_1 and L_2?
  - There exist DFAs D_1, D_2 that recognizes L_1 and L_2, respectively
- What do we want to show for $L_1 \cup L_2$?
  - Want to show that there is a DFA D_3 that recognizes $L_1 \cup L_2$

Closure of regular languages under union

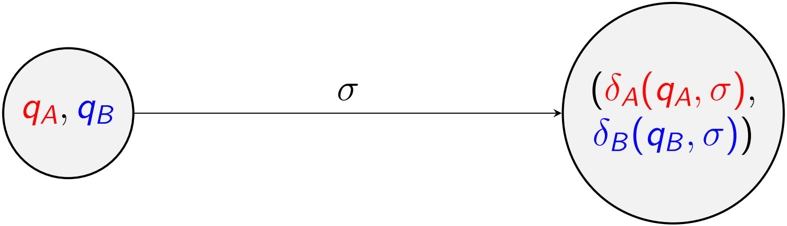
Closure of regular languages under union

- **Proof idea:** Using D_1 and D_2, we will construct a DFA that runs both machines simultaneously and accepts if either machine accepts.

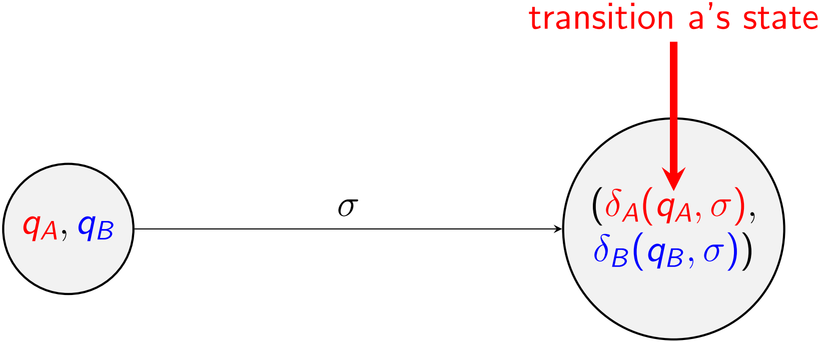
Closure of regular languages under union

- **Proof idea:** Using D_1 and D_2, we will construct a DFA that runs both machines simultaneously and accepts if either machine accepts.
- **Technique:** Run two DFAs *in parallel* using the cartesian product construction

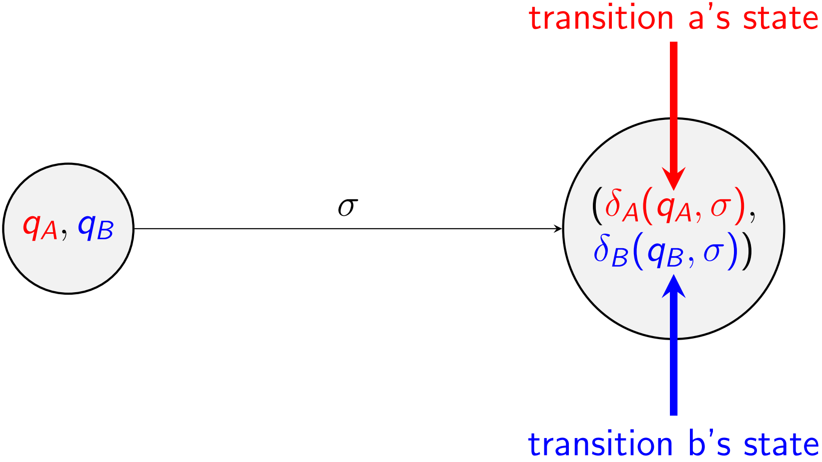
Union idea



Union idea


transition a's state
q_A, q_B
σ
$(\delta_A(q_A, \sigma),$
$\delta_B(q_B, \sigma))$

Union idea



transition a's state
q_A, q_B
σ
$(\delta_A(q_A, \sigma),$
$\delta_B(q_B, \sigma))$
transition b's state

Union closure

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- $q_s = (q_{s_A}, q_{s_B})$

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- $q_s = (q_{s_A}, q_{s_B})$
- $F = \{(q_A, q_B) \in Q | q_A \in F_A \text{ or } q_B \in F_B\}$

Union closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$

Union closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- Each state is a combination of 2 elements:

Union closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- Each state is a combination of 2 elements:
  - A state $q_A \in Q_A$

Union closure - states

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $Q = Q_A \times Q_B$
- Each state is a combination of 2 elements:
  - A state $q_A \in Q_A$
  - A state $q_B \in Q_B$

Union closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$

Union closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- We transition A to its next state

Union closure - transition function

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- We transition A to its next state
- We simultaneously transition B to its next state

Union closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $q_s = (q_{s_A}, q_{s_B})$

Union closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $q_s = (q_{s_A}, q_{s_B})$
- We start out in A's start state

Union closure - start state

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $q_s = (q_{s_A}, q_{s_B})$
- We start out in A's start state
- We start out in B's start state

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $F = \{(q_A, q_B) \in Q | q_A \in F_A$ or $q_B \in F_B\}$

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $F = \{(q_A, q_B) \in Q | q_A \in F_A \text{ or } q_B \in F_B\}$
- Either A's state should be one of its accept states...

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $F = \{(q_A, q_B) \in Q | q_A \in F_A \text{ or } q_B \in F_B\}$
- Either A's state should be one of its accept states...
- ... or B's state should be one of its accept states

Union closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

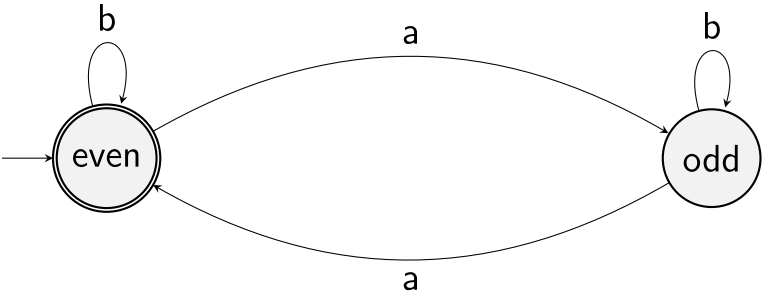
Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cup B$

- $F = \{(q_A, q_B) \in Q | q_A \in F_A \text{ or } q_B \in F_B\}$
- Either A's state should be one of its accept states...
- ... or B's state should be one of its accept states
- (or perhaps both!)

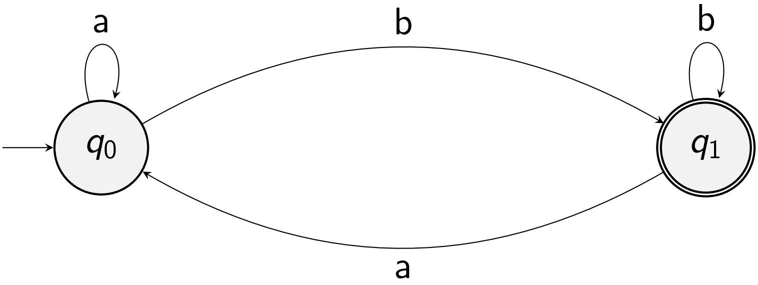
Union example

DFA for $A = \{w | w \text{ has an even number of a's}\}$

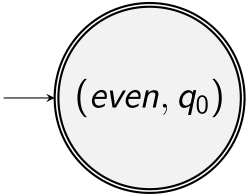


Union example

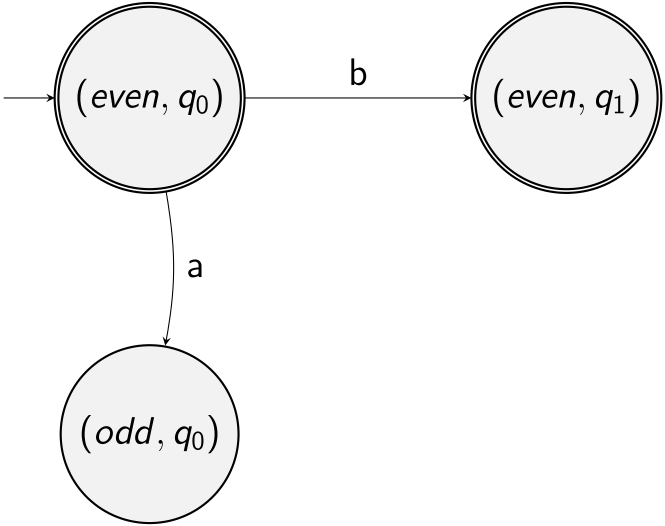
DFA for $B = \{w | w \text{ ends with b}\}$



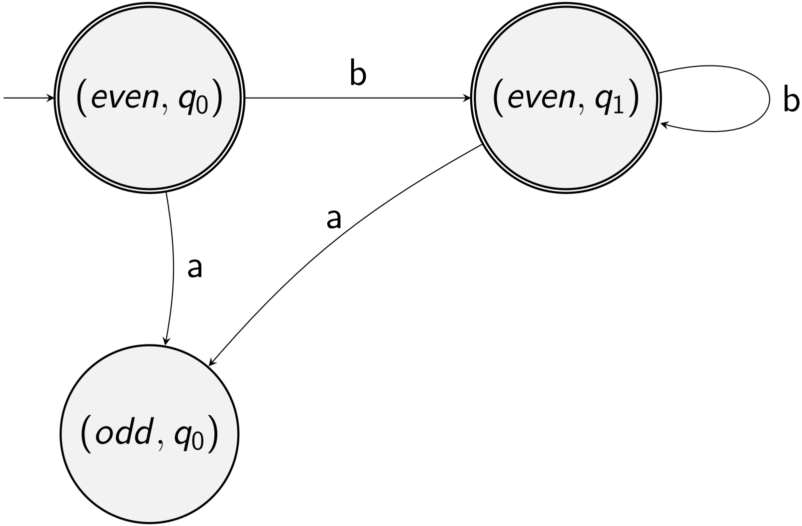
Union example



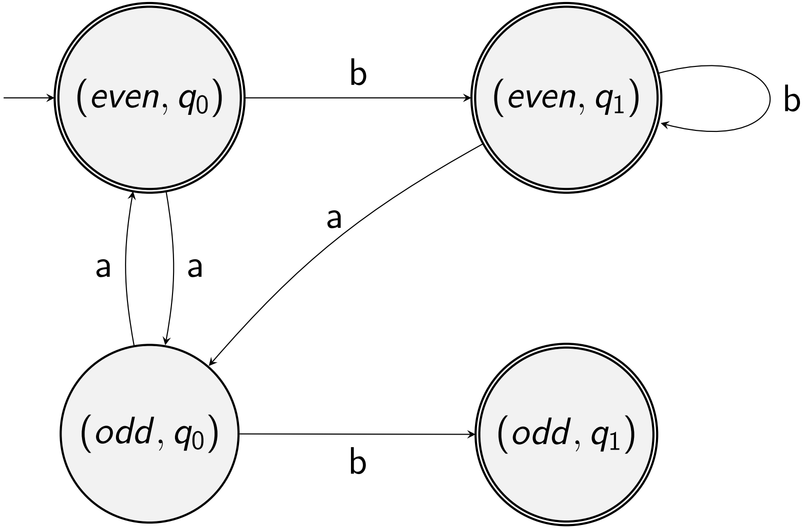
Union example



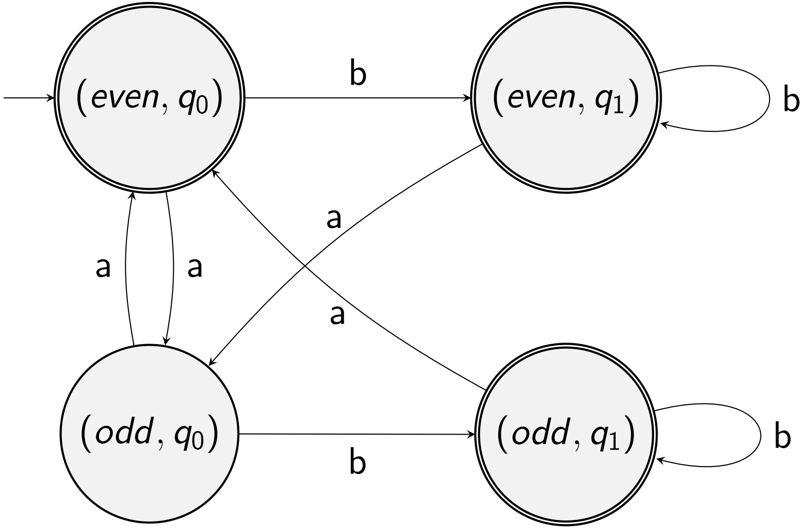
Union example



Union example



Union example



Closure of regular languages under intersection

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$
- Q: How do we prove closure under intersection?

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$
- Q: How do we prove closure under intersection?
  - A: show that if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$
- Q: How do we prove closure under intersection?
  - A: show that if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular
- Q: What technique do we use to do this?

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$
- Q: How do we prove closure under intersection?
  - A: show that if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular
- Q: What technique do we use to do this?
  - **Technique 1:** Run the two machines in parallel using the Cartesian product construction

Closure of regular languages under intersection

- $L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$
- Q: How do we prove closure under intersection?
  - A: show that if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular
- Q: What technique do we use to do this?
  - **Technique 1:** Run the two machines in parallel using the Cartesian product construction
  - **Technique 2:** Express intersection in terms of other operations that we know regular languages are closed under

Intersection closure

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

- $Q = Q_A \times Q_B$

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- $q_s = (q_{s_A}, q_{s_B})$

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- $q_s = (q_{s_A}, q_{s_B})$
- $\mathrm{F} = \mathrm{F_A} \times \mathrm{F_B}$

Intersection closure

Let $D_A = (Q_A, \Sigma, q_{s_A}, \delta_A, F_A)$ recognize A

Let $D_B = (Q_B, \Sigma, q_{s_B}, \delta_B, F_B)$ recognize B

The following DFA $D = (Q, \Sigma, q_s, \delta, F)$ will recognize $A \cap B$

- $Q = Q_A \times Q_B$
- $\delta((q_A, q_B), \sigma) = (\delta_A(q_A, \sigma), \delta_B(q_B, \sigma))$
- $q_s = (q_{s_A}, q_{s_B})$
- $\mathrm{F} = \mathrm{F_A} \times \mathrm{F_B}$
  - Need an accept state for A *and* for B

Intersection closure

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under
- $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ (De Morgan's laws)

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under
- $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ (De Morgan's laws)
- Regular languages are closed under complement, so L_1^c and L_2^c are both regular

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under
- $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ (De Morgan's laws)
- Regular languages are closed under complement, so L_1^c and L_2^c are both regular
- Regular languages are closed under union, so $(L_1^c \cup L_2^c)$ is regular

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under
- $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ (De Morgan's laws)
- Regular languages are closed under complement, so L_1^c and L_2^c are both regular
- Regular languages are closed under union, so $(L_1^c \cup L_2^c)$ is regular
- Regular languages are closed under complement, so $(L_1^c \cup L_2^c)^c$ is regular

Intersection closure

- **Technique:** Express intersection in terms of other operations that we know regular languages are closed under
- $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ (De Morgan's laws)
- Regular languages are closed under complement, so L_1^c and L_2^c are both regular
- Regular languages are closed under union, so $(L_1^c \cup L_2^c)$ is regular
- Regular languages are closed under complement, so $(L_1^c \cup L_2^c)^c$ is regular
- Thus, $L_1 \cap L_2$ is regular!

Closure under concatenation

Closure under concatenation

- Let D_1 and D_2 recognize L_1 and L_2

Closure under concatenation

- Let D_1 and D_2 recognize L_1 and L_2
- We need to combine them into a machine that recognizes $L_1 \circ L_2$

Closure under concatenation

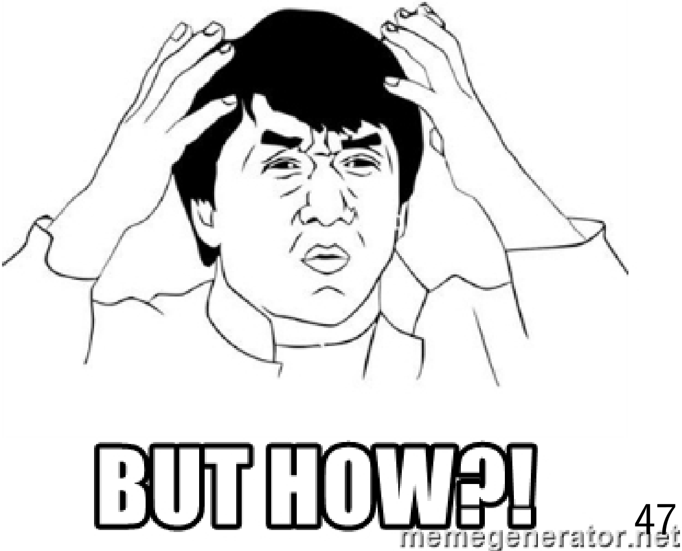
- Let D_1 and D_2 recognize L_1 and L_2
- We need to combine them into a machine that recognizes $L_1 \circ L_2$
- We can't run the machines in parallel

Closure under concatenation

- Let D_1 and D_2 recognize L_1 and L_2
- We need to combine them into a machine that recognizes $L_1 \circ L_2$
- We can't run the machines in parallel
- We need to run them *in sequence* - one after the other

Clos

-
- at
-
-



Closure under concatenation

- Let D_1 and D_2 recognize L_1 and L_2
- We need to combine them into a machine that recognizes $L_1 \circ L_2$
- We can't run the machines in parallel
- We need to run them *in sequence* - one after the other

Technique: nondeterminism!