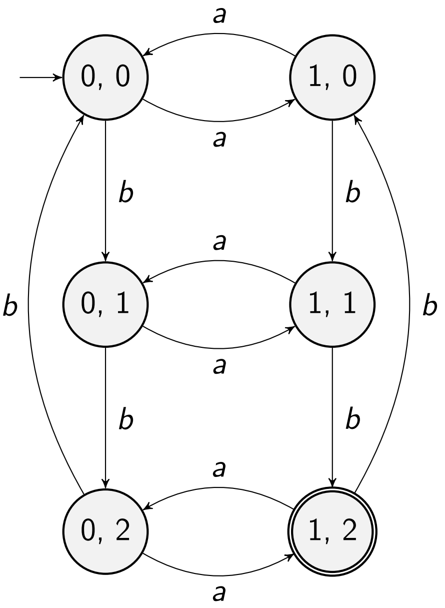
Review for CMSC 452 Final

Deterministic Finite Automata (DFA)

$$\{w : \#_a(w) \equiv 1 \pmod 2 \wedge \#_b(w) \equiv 2 \pmod 3\}$$

$\{w : \#_a(w) \equiv 1 \pmod 2 \wedge \#_b(w) \equiv 2 \pmod 3\}$



Nondeterministic Finite Automata (NFA)

NFA's Intuitively

1. An NFA is a DFA that can guess.
2. NFAs do not really exist.
3. Good for $\cup$ since can guess which one.
4. An NFA accepts iff SOME guess accepts.

Every NFA-lang a DFA-lang!

Thm If L is accepted by an NFA then L is accepted by a DFA.
Pf Sketch L is accepted by NFA $(Q, \Sigma, \Delta, s, F)$ where

1. Get rid of e-transitions using reachability.
2. Get rid of non-determinism by using power sets. Possibly 2^n blowup.

Regular Expressions

Examples

1. $b^*(ab^*ab^*)^*ab^*$
2. $b^*(ab^*ab^*ab^*)^*$
3. $(b^*(ab^*ab^*)^*ab^*) \cup (b^*(ab^*ab^*ab^*)^*)$

DFA = NFA = REGEX

NFA $\subseteq$ DFA: Use Power Set Construction. Exp Blowup.

DFA = NFA = REGEX

NFA $\subseteq$ DFA: Use Power Set Construction. Exp Blowup.

DFA $\subseteq$ REGEX: Use $R(i,j,k)$ construction.

DFA = NFA = REGEX

NFA $\subseteq$ DFA: Use Power Set Construction. Exp Blowup.

DFA $\subseteq$ REGEX: Use $R(i, j, k)$ construction.

REGEX $\subseteq$ NFA: Induction on formation of regex. Linear.

Closure Properties

Summary of Proofs of Closure Properties

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$
Def means by Definition, e.g., $L_1 \cup L_2$ for regex.

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$

Def means by Definition, e.g., $L_1 \cup L_2$ for regex.

Swap means swapping final and non-final states.

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$

Def means by Definition, e.g., $L_1 \cup L_2$ for regex.

Swap means swapping final and non-final states.

***e*-trans** means by using *e*-transitions, e.g., $L_1 \cdot L_2$ for NFAs.

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$

Def means by Definition, e.g., $L_1 \cup L_2$ for regex.

Swap means swapping final and non-final states.

***e*-trans** means by using e-transitions, e.g., $L_1 \cdot L_2$ for NFAs.

X means hard to prove, e.g., $\overline{L}$ for NFA.

Summary of Proofs of Closure Properties

Prod means product construction where you use $Q_1 \times Q_2$

Def means by Definition, e.g., $L_1 \cup L_2$ for regex.

Swap means swapping final and non-final states.

***e*-trans** means by using *e*-transitions, e.g., $L_1 \cdot L_2$ for NFAs.

X means hard to prove, e.g., $\overline{L}$ for NFA.

Property	DFA	NFA	regex
$L_1 \cup L_2$	Prod	*e*-trans	Def
$L_1 \cap L_2$	Prod	Prod	X
$\overline{L}$	Swap	X	X
$L_1 \cdot L_2$	X	*e*-trans	Def
L^*	X	*e*-trans	Def

Summary of Blowup for Closure Properties

X means **Can't Prove Easily**

Summary of Blowup for Closure Properties

X means **Can't Prove Easily**

n_1, n_2 are number of states in a DFA or NFA.

Summary of Blowup for Closure Properties

X means **Can't Prove Easily**

n_1, n_2 are number of states in a DFA or NFA.

$\ell_, \ell_2$ are length of regex.

Summary of Blowup for Closure Properties

X means **Can't Prove Easily**

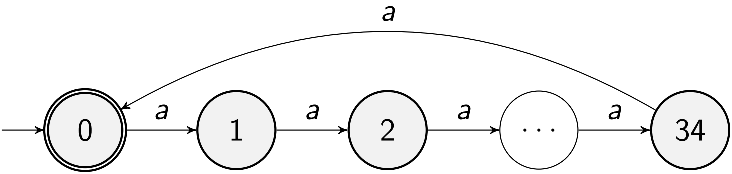
n_1, n_2 are number of states in a DFA or NFA.

$\ell_, \ell_2$ are length of regex.

Closure Property	DFA	NFA	Regex
$L_1 \cup L_2$	$n_1 n_2$	$n_1 + n_2 + 1$	$\ell_1 + \ell_2$
$L_1 \cap L_2$	$n_1 n_2$	$n_1 n_2$	X
$L_1 \cdot L_2$	X	$n_1 + n_2$	$\ell_1 + \ell_2$
$\overline{L}$	n	X	X
L^*	X	$n + 1$	$\ell + 1$

Number of States for DFAs and NFAs

Minimal DFA for $L_1 = \{a^i : i \equiv 0 \pmod{35}\}$



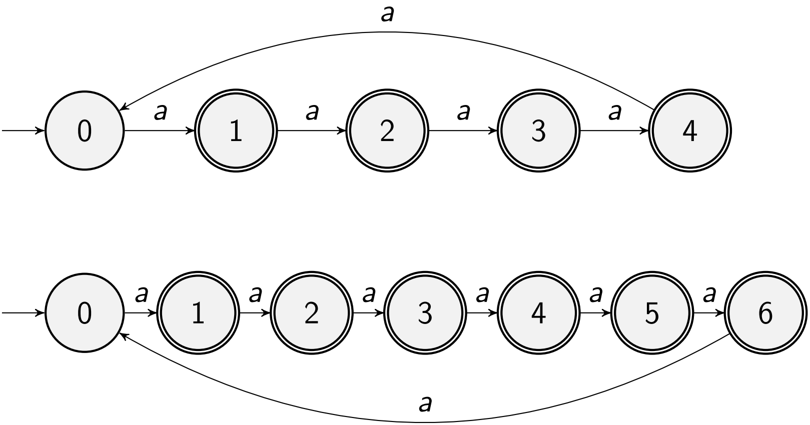
Min DFA for $L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

Min DFA for $L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

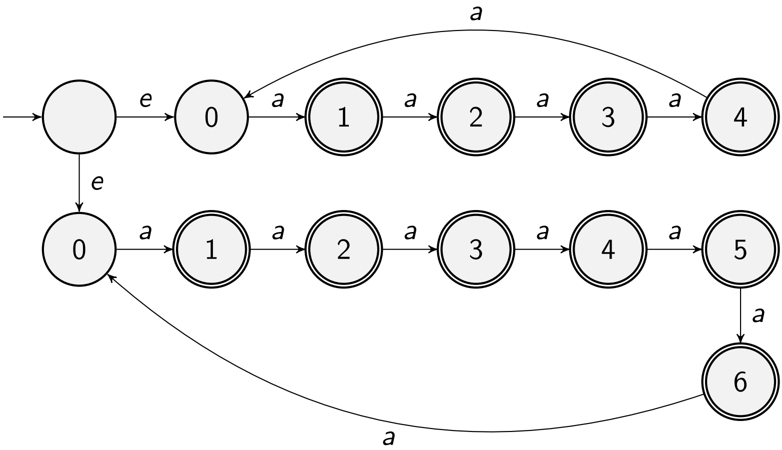
$\exists$ DFA for L_2: 35 states: swap final-$\overline{\text{final}}$ states in DFA for L_1.

Small NFA for $L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

Need these two NFA's.



Small NFA for $L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$



$L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

$L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

DFA for L_2 requires 35 states.

$L_2 = \{a^i : i \not\equiv 0 \pmod{35}\}$

DFA for L_2 requires 35 states.

NFA for L_2 can be done with $1 + 5 + 7 = 13$ states.

Proving That a Language Is Not Regular

Pumping Lemma

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy| \leq n_1$ (or can take $|yz| \leq n_1$ but not both.)

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy| \leq n_1$ (or can take $|yz| \leq n_1$ but not both.)
3. For all $i \geq 0$, $xy^i z \in L$.

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy| \leq n_1$ (or can take $|yz| \leq n_1$ but not both.)
3. For all $i \geq 0$, $xy^iz \in L$.

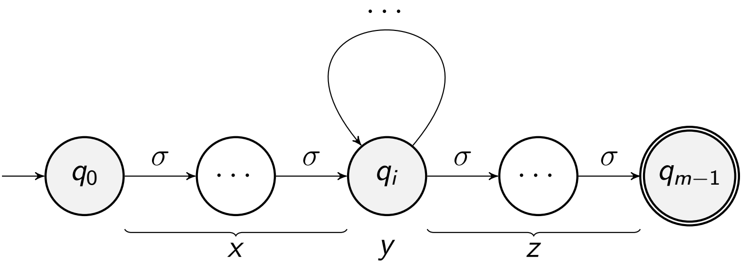
Proof by picture

Pumping Lemma

Pumping Lemma (PL) If L is regular then there exist n_0 and n_1 such that the following holds:
For all $w \in L$, $|w| \geq n_0$ there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy| \leq n_1$ (or can take $|yz| \leq n_1$ but not both.)
3. For all $i \geq 0$, $xy^iz \in L$.

Proof by picture



How We Use the PL

How We Use the PL

We restate it in the way that we use it.

How We Use the PL

We restate it in the way that we use it.

PL If L is reg then **for large enough strings w in L** there exist x, y, z such that:

How We Use the PL

We restate it in the way that we use it.

PL If L is reg then **for large enough strings w in L** there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.

How We Use the PL

We restate it in the way that we use it.

PL If L is reg then **for large enough strings w in L** there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy|$ **is short**.

How We Use the PL

We restate it in the way that we use it.

PL If L is reg then **for large enough strings w in L** there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy|$ **is short**.
3. for all i, $xy^iz \in L$.

How We Use the PL

We restate it in the way that we use it.

PL If L is reg then **for large enough strings w in L** there exist x, y, z such that:

1. $w = xyz$ and $y \neq e$.
2. $|xy|$ **is short**.
3. for all i, $xy^iz \in L$.

We then find some i such that $xy^iz \notin L$ for the contradiction.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n$$

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n \;=\; a^{n+k(i-1)} b^n$$

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n \;=\; a^{n+k(i-1)} b^n$$

are in L_1.

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n \;=\; a^{n+k(i-1)} b^n$$

are in L_1.
Take $i = 2$ to get

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n \;=\; a^{n+k(i-1)} b^n$$

are in L_1.
Take $i = 2$ to get

$$a^{n+k} b^n \in L_1$$

REDO: $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ is Not Regular

Assume L_1 is regular.
By PL, for long $a^n b^n \in L_1$, $\exists x, y, z$:

1. $y \neq e$.
2. $|xy|$ is short.
3. For all $i \geq 0$, $xy^i z \in L_1$.

Take w long enough so that the xy part only has a's.
$x = a^j$, $y = a^k$, $z = a^{n-j-k} b^n$. Note $k \geq 1$.
By the PL, all of the words

$$a^j \left(a^k\right)^i a^{n-j-k} b^n \; = \; a^{n+k(i-1)} b^n$$

are in L_1.
Take $i = 2$ to get

$$a^{n+k} b^n \in L_1$$

Contradiction since $k \geq 1$.

$L_3 = \{w : \#_a(w) \neq \#_b(w)\}$ is Not Regular

$L_3 = \{w : \#_a(w) \neq \#_b(w)\}$ is Not Regular

PL Does Not Help. When you increase the number of y's there is no way to control it so carefully to make the number of a's EQUAL the number of b's.

$L_3 = \{w : \#_a(w) \neq \#_b(w)\}$ is Not Regular

PL Does Not Help. When you increase the number of y's there is no way to control it so carefully to make the number of a's EQUAL the number of b's.

So what do to?

$L_3 = \{w : \#_a(w) \neq \#_b(w)\}$ is Not Regular

PL Does Not Help. When you increase the number of y's there is no way to control it so carefully to make the number of a's EQUAL the number of b's.

So what do to?

If L_3 is regular then $L_2 = \overline{L_3}$ is regular. But we know that L_2 is not regular. DONE!

$L_4 = \{a^{n^2} : n \in \mathbb{N}\}$ is Not Regular

$L_4 = \{a^{n^2} : n \in \mathbb{N}\}$ is Not Regular

Intuition Perfect squares keep getting further apart.

$L_4 = \{a^{n^2} : n \in \mathbb{N}\}$ is Not Regular

Intuition Perfect squares keep getting further apart.
PL says you can always add some constant k to produce a word in the language.
We omit details.

Applications of DFAs

Applications of DFAs

1. Lexical Analyzer for compilers (we didn't do this).

Applications of DFAs

1. Lexical Analyzer for compilers (we didn't do this).
2. Pattern Matching Algorithms like grep (we didn't do this).

Applications of DFAs

1. Lexical Analyzer for compilers (we didn't do this).
2. Pattern Matching Algorithms like grep (we didn't do this).
3. Decidability of WS1S (we did this).