Administrivia

- Exercises, 90 minutes every week, starting this week:
  - 2x Thursday 14:15-15:45
  - 2x Thursday 16:15-17:45
  - 1x Friday 14:15-15:45
- Exercise sheet #1 will be discussed in these meetings
- Exercise sheet #2 assigned today
- Optional Exam #1: Wednesday October 29, 9:00-10:00
- Reading: Chapters 0 and 1
- Last time: DFAs, NFAs, regular languages
- Today: regular expressions, non-regular languages, Pumping Lemma

RE $\Longleftrightarrow$ FAs

Theorem

A language is regular if and only if some regular expression describes it.

Proof.

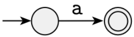
$\Leftarrow$ Consider some regular expression R that describes language A. We show how to convert R into an NFA recognizing A. (We showed this last time, by example and gave a formal proof using the inductive definition of a RE.)

Rightarrow We show how to get a RE from a FA. □

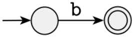
Example of RE to NFA conversion

Let $R = (ab \cup a)^*$

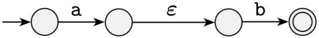
a



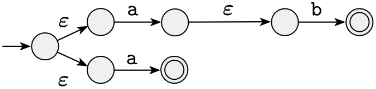
b



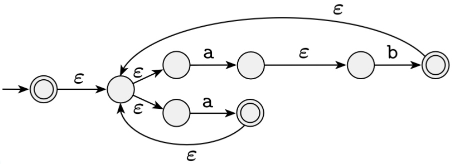
ab



ab ∪ a



(ab ∪ a)*



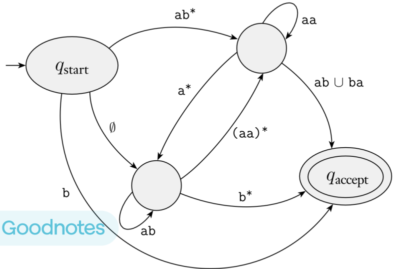
RE $\Longleftrightarrow$ FAs

Theorem

A language is regular if and only if some regular expression describes it.

Proof.

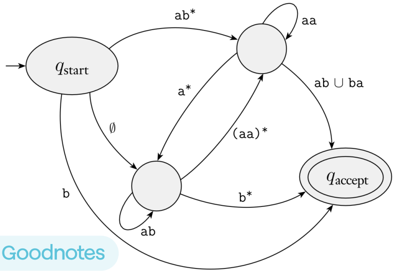
$\Rightarrow$ Given some regular language A, there exists a DFA that recognizes A. We show how to extract an equivalent regular expression, by means of a *generalized nondeterministic finite automaton (GNFA)*. □



Example of DFA to RE conversion

Start with the given NFA and convert into GNFA:

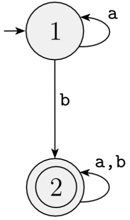
- start state has transition arrows to every other state
- start state has no arrows coming in from any other state
- there is only a single accept state, and it has arrows coming in from every other state
- accept state has no arrows going to any other state
- except for start and accept states, one arrow goes from every state to every other state; also from each state to itself



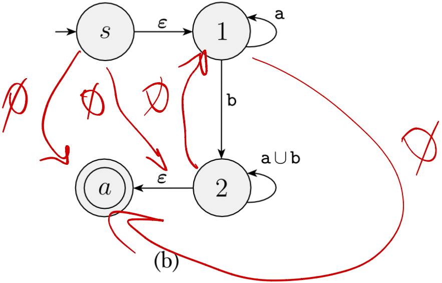
Example of DFA to RE conversion

This is easy to do:

- add new start state with ϵ arrow to the old start state
- add new accept state with ϵ arrows from old accept states
- If any arrows have multiple labels (or if there are multiple arrows going between the same two states in the same direction), we replace each with a single arrow whose label is the union of the previous labels.
- add arrows labeled $\emptyset$ between states that had no arrows



(a)

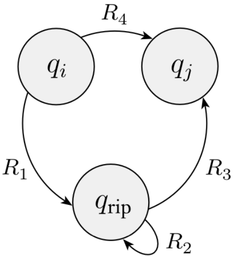


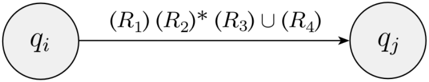
(b)

Example of DFA to RE conversion

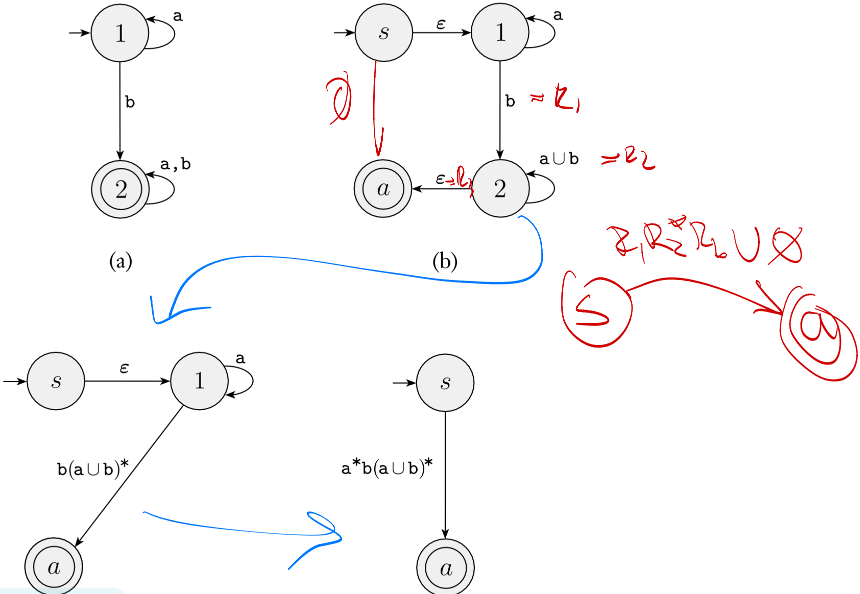
Now we remove one state at a time from the GNFA, while ensuring it still recognizes the same language, until only the start state and the accept state remain:

- let q_{rip} be the state we remove
- consider every pair of states (q_i, q_j)
- update (q_i, q_j) transition label to account for removal of q_{rip}



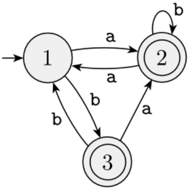


Example of DFA to RE conversion

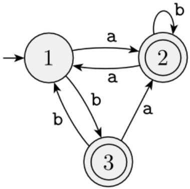


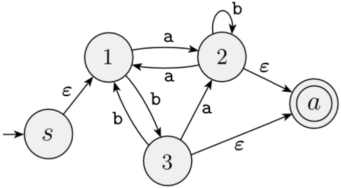
(a) (b) (c) (d)

Example of DFA to RE conversion

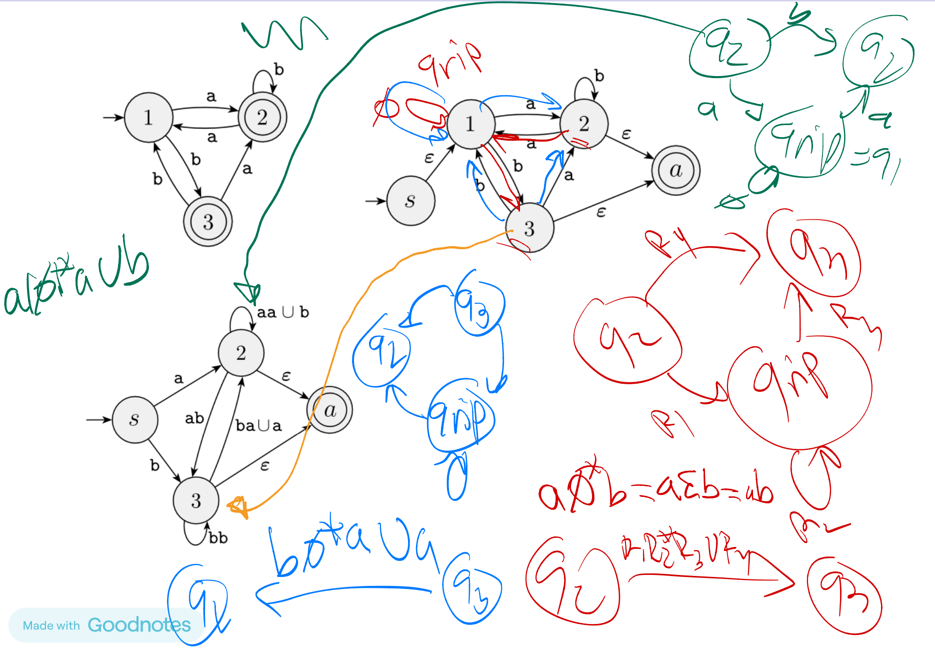


Example of DFA to RE conversion

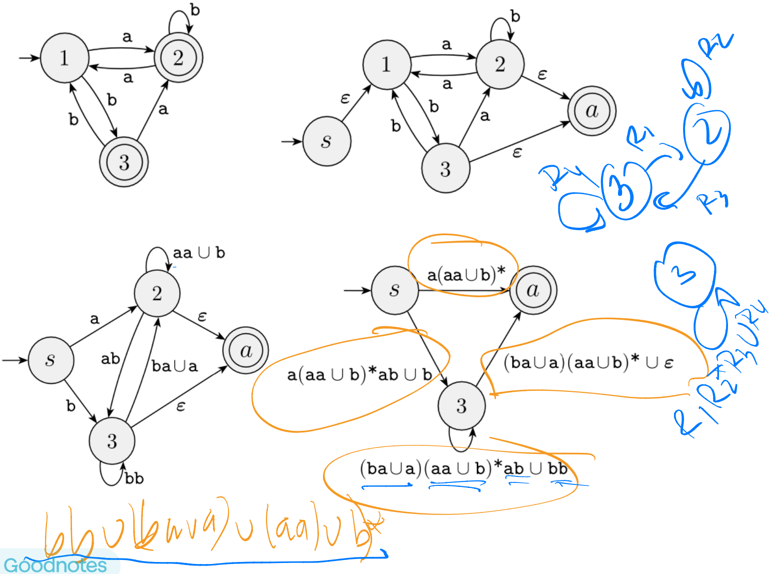




Example of DFA to RE conversion

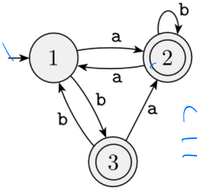


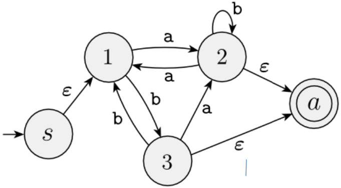
Example of DFA to RE conversion

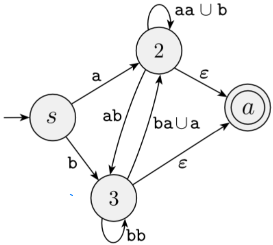


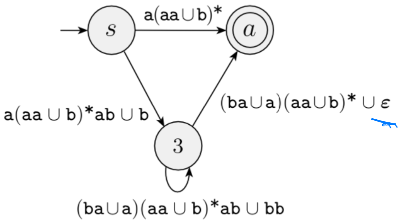
bb ∪ (ba∪a) ∪ (aa) ∪ b*

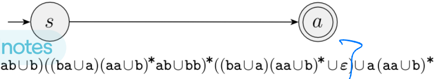
Example of DFA to RE conversion










s
a
(a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)* ∪ ε) ∪ a(aa∪b)*

RE $\Longleftrightarrow$ FAs

This completes the two parts of a fairly complicated argument summarized again below.

Theorem

A language is regular if and only if some regular expression describes it.

Proof.

$\Rightarrow$ Start with an NFA for A and convert into an equivalent GNFA. Remove all states one by one, except the unique start state and the unique accept state. The expression on the arc from the start to the accept state is the regular expression that describes A.

$\Leftarrow$ We showed this using the inductive definition of a regular expression $(a, \epsilon, \emptyset, R_1 \cup R_2, R_1 \circ R_2, R^*)$ to convert the given regular expression for A into an NFA for A. □

What Makes a Language Regular

Consider the languages

- $L = \{0^*1^*\}$
- $L = \{01\}$
- $L = \{0011\}$
- $L = \{\epsilon, 01, 0011\}$
- $L = \{0^n1^n | n \geq 0\}$

Let's try to build FAs for them...

The Pumping Lemma (PL) for Regular Languages

0^*1^* $P=2$

00
01
11
000

Theorem

(PL) If A is a regular language, then there exists a number p (the pumping length) *so that if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:*

1. $xy^iz \in A$, *for each* $i \geq 0$,
2. $|y| > 0$, *and*
3. $|xy| \leq p$.

$s = \underset{x}{\varepsilon}\,\underset{y}{0}\,\underset{z}{0}$

$i=0$ $xy^0z = 0$

$i=1$ $xyz = 00$

$xy^2z = 000$

Made with Goodnotes

The Pumping Lemma (PL) for Regular Languages

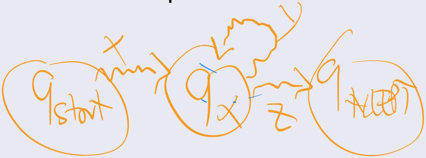
Theorem

If A is regular, then $\exists p$ so that if $s \in A$ and $|s| > p$, then s may be divided into three pieces, $s = xyz$, satisfying the following:

1. $xy^iz \in A$, *for each* $i \geq 0$,
2. $|y| > 0$, *and*
3. $|xy| \leq p$.

Proof.

Let $M = (Q, \Sigma, \delta, q_s, F)$ be a DFA for language A. Then we set $p = |Q|$. Consider some string $s \in A$ such that $|s| > p$. While computing, M must repeat some state (by the pigeonhole principle). Let q_x be the first state that is repeated. Then this picture completes the proof:



□

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Claim

$L = \{0^n 1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Claim

$L = \{0^n1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.
- Let p be the pumping length.

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

$\overbrace{00\ldots0}^{p}11\ldots1$

Claim

$L = \{0^n1^n | n \geq 0\}$ is not a regular language

Proof.

- Suppose L is regular. Then the PL applies. $S = xyz$
- Let p be the pumping length.
- Let's pick a string $s = 0^p1^p$ from L and longer than p. $|xy| \leq p$

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Claim

$L = \{0^n 1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.
- Let p be the pumping length.
- Let's pick a string $s = 0^p 1^p$ from L and longer than p.
- Now we must consider all possible ways to break s into three parts $s = xyz$ subject to PL conditions $|y| > 0$ and $|xy| \leq p$.

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL $\overbrace{000\ldots0}^{p}11\ldots1$

Claim

$L = \{0^n1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.
- Let p be the pumping length.
- Let's pick a string $s = 0^p1^p$ from L and longer than p.
- Now we must consider all possible ways to break s into three parts $s = xyz$ subject to PL conditions $|y| > 0$ and $|xy| \leq p$.
- Clearly, y must contain one or more zeros (from conditions 2 and 3).

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Claim

$L = \{0^n 1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.
- Let p be the pumping length.
- Let's pick a string $s = 0^p 1^p$ from L and longer than p.
- Now we must consider all possible ways to break s into three parts $s = xyz$ subject to PL conditions $|y| > 0$ and $|xy| \leq p$.
- Clearly, y must contain one or more zeros (from conditions 2 and 3).
- Condition 1 of the PL requires that $xy^i z \in L$ for each $i \geq 0$.

Proving a Language Non-Regular

- To prove a language is regular: build a FA or give a RE
- To prove a language is non-regular, we use the PL

Claim

$L = \{0^n 1^n | n \geq 0\}$ *is not a regular language*

Proof.

- Suppose L is regular. Then the PL applies.
- Let p be the pumping length.
- Let's pick a string $s = 0^p 1^p$ from L and longer than p.
- Now we must consider all possible ways to break s into three parts $s = xyz$ subject to PL conditions $|y| > 0$ and $|xy| \leq p$.
- Clearly, y must contain one or more zeros (from conditions 2 and 3).
- Condition 1 of the PL requires that $xy^i z \in L$ for each $i \geq 0$.
- But this is not possible: “pumping” y up or down results in strings that are not in L, $xz \notin L$ (fewer 0s than 1s), $xyyz \notin L$ (more 0s than 1s).

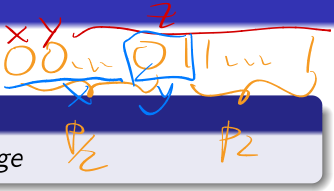
□

Proving a Language Non-Regular

Use PL to prove a language is non-regular. Think of it as a game:

- I **give** you the language L
- I **give** you a value p
- You **choose** a string s longer than p
- Choose carefully in order to break the claim that s satisfies the 3 conditions of the PL
- Note that you **do not choose** how s is broken into 3 parts $s = xyz$
- You must **argue that no matter how I break** s into 3 parts there is always a contradiction.
- Choosing s poorly can make your job harder or impossible.

Proving a Language Non-Regular



Claim

$L = \{0^n 1^n | n \geq 0\}$ *is not a regular language*

$|xy| \leq p$

Now, let's choose a different string s.

Proof.

Suppose L is regular. Then the PL applies. Let p be the pumping length. Then consider the string $s = 0^{\lceil p/2 \rceil} 1^{\lceil p/2 \rceil}$ and all possible ways to break s down into $s = xyz$ subject to the 3 PL conditions.

- y contains one or more 0s; then xy^2z contains more 0s than 1s
- y contains one or more 1s; then xy^2z contains more 1s than 0s
- y contains 0s and 1s; then xy^2z alternates b/n 0s and 1s more than once.

□