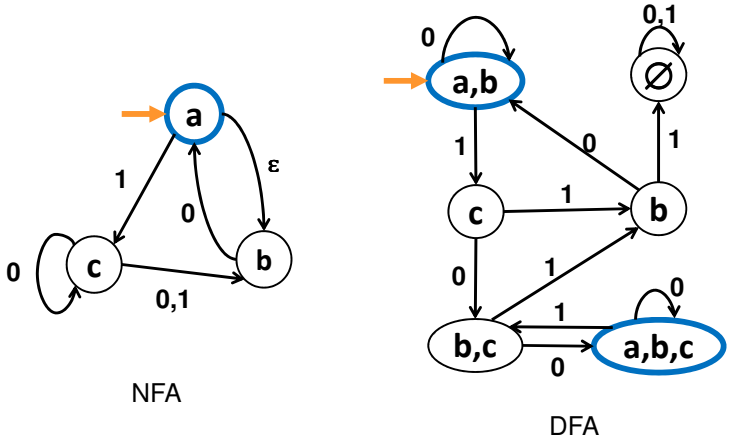
CSE 311: Foundations of Computing

Fall 2013

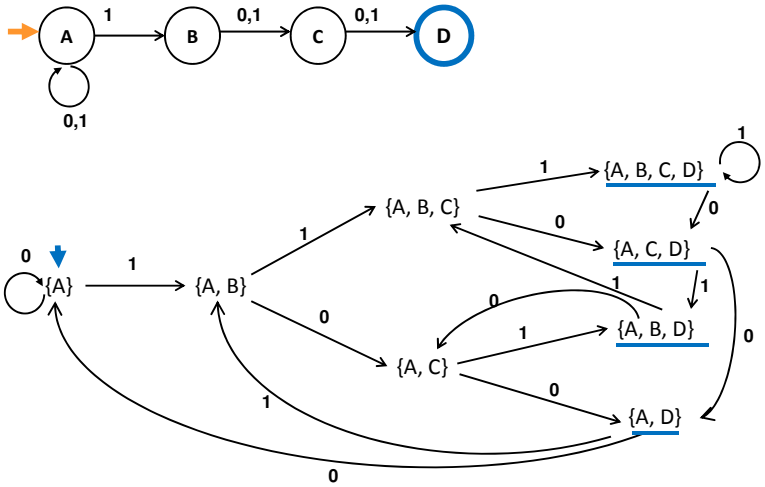
Lecture 25: Non-regularity and limits of FSMs

Subset Construction

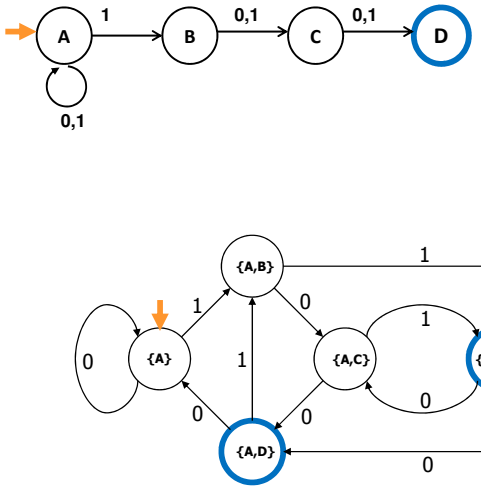
"Subset construction": NFA to DFA



1 in third position from end



Redrawing



DFAs ≡ Regular expressions

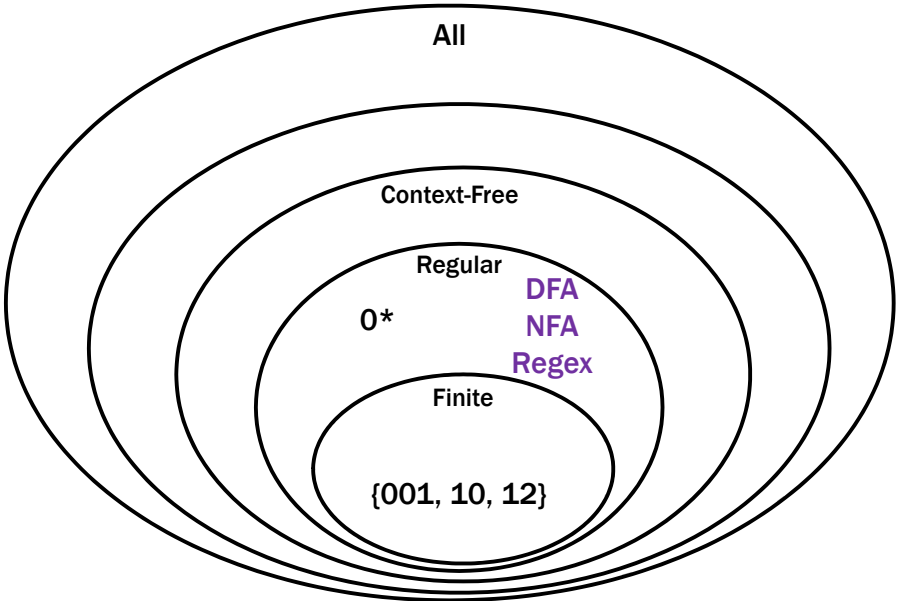
We have shown how to build an optimal DFA for every regular expression

- Build NFA
- Convert NFA to DFA using subset construction
- Minimize resulting DFA

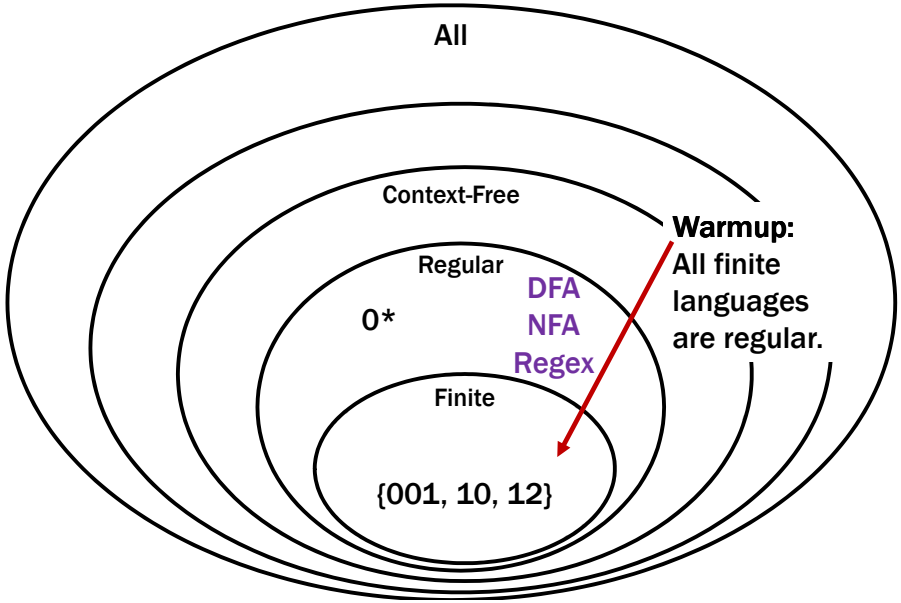
Theorem: A language is recognized by a DFA if and only if it has a regular expression

We show the other direction of the proof at the end of these lecture slides

Languages and Machines!

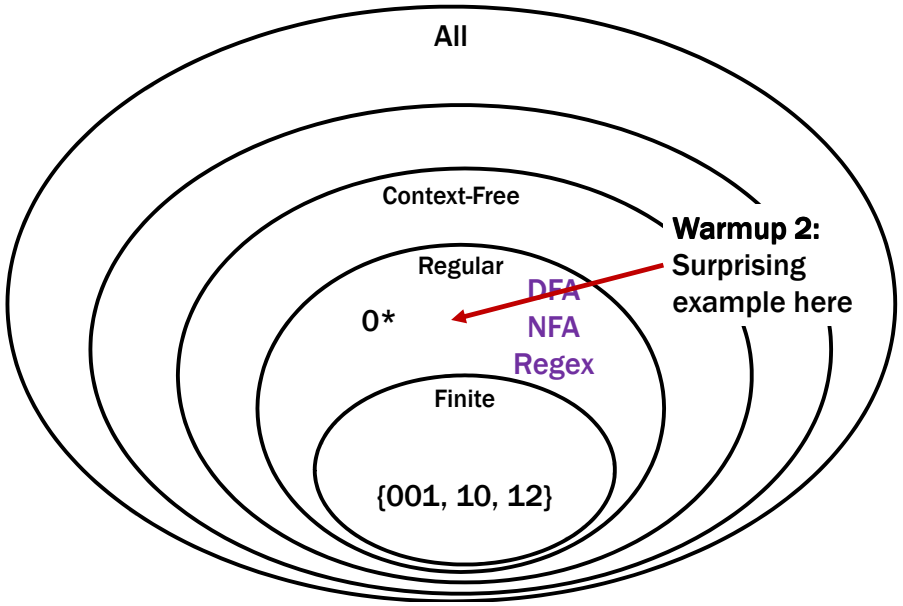


Languages and Machines!



DFAs Recognize Any Finite Language

Languages and Machines!



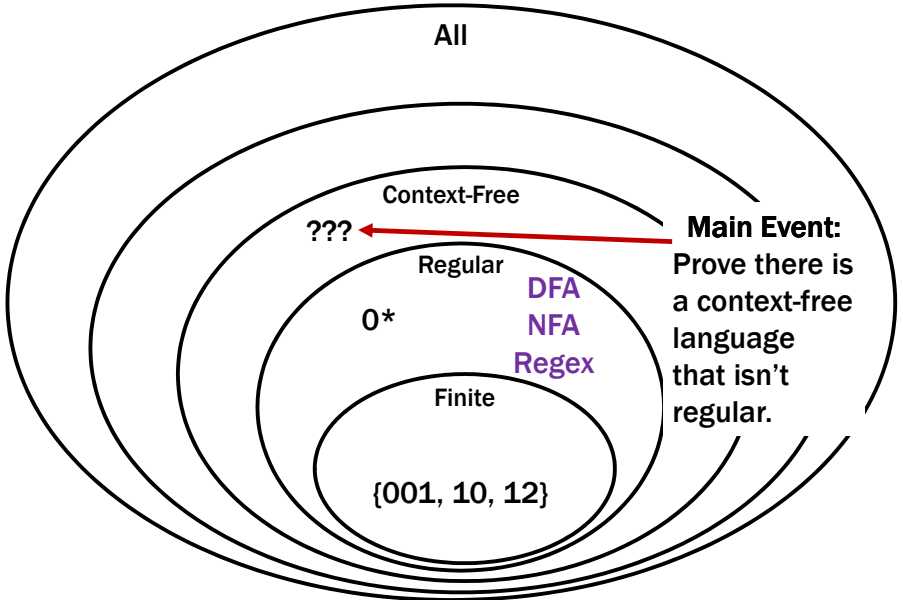
An Interesting Infinite Regular Language

L = {x∈ {0, 1}*: x has an equal number of substrings 01 and 10}.

L is infinite.

L is regular.

Languages and Machines!



Irregular Language!

B = {binary palindromes} can't be recognized by any DFA

Why is this language not regular?

Intuition (NOT A PROOF!):

Q: What would a DFA need to keep track of to decide the language?

A: It would need to keep track of the "first part" of the input in order to check the second part against it

...but there are an infinite # of possible first parts and we only have finitely many states.

How do we prove it?

B = {binary palindromes} can't be recognized by any DFA

Consider the infinite set of strings

$S=\{1, 01, 001, 0001, 00001, ...\} = \{0^n1 : n \geq 0\}$

That's a nice set of first parts to have to remember but how can we argue that a DFA does the wrong thing for B?

- Show that some **x** ∈ B and some y ∉ B both must end up at the *same* state of the DFA

That state can't be

- a final state since then y is accepted: error on y
- a non-final state since then x is rejected: error on x

B = {binary palindromes} can't be recognized by any DFA

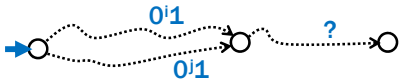
Consider the infinite set of strings

$S=\{1, 01, 001, 0001, 00001, ...\} = \{0^n1 : n \geq 0\}$

Suppose we are given an arbitrary DFA **M**.

- Goal: Show that some **x** ∈ **B** and some **y** ∉ **B** both must end up at the *same* state of **M**

Since **S** is infinite we know that two different strings in **S** must land in the same state of **M**, call them 0^i1 and 0^j1 for i≠j.



- That also must be true for 0^i1z and 0^j1z for any **z** ∈ {0,1}* !

In particular, with $z=0^i$ we get that 0^i10^i and 0^j10^i end up at the same state of **M**. Since $0^i10^i \in B$ and $0^j10^i \notin B$ (because i≠j) M does not recognize **B**. ∴ no DFA can recognize **B**.

Showing a Language L is not regular

1. Find an infinite set $S=\{s_0,s_1,...,s_n,...\}$ of string prefixes that you think will need to be remembered separately
2. "Let **M** be an arbitrary DFA. Since **S** is infinite and **M** is finite state there must be two strings s_i and s_j in **S** for some **i ≠j** that end up at the same state of **M**."

   Note: You don't get to choose which two strings s_i and s_j
3. Find a string **t** (typically depending on s_i and/or s_j) such that

   s_it is in **L**, and s_jt is not in **L**

   or s_it is not in **L**, and s_jt is in **L**
4. "Since s_i and s_j both end up at the same state of **M**, and we appended the same string **t**, both s_it and s_jt end at the same state of **M**. Since $s_it \in L$ and $s_jt \notin L$, **M** does not recognize **L**."
5. "Since **M** was arbitrary, no DFA recognizes **L**."

A=$\{0^n1^n : n \geq 0\}$ cannot be recognized by any DFA

Another Irregular Language Example

L = {x∈ {0, 1,2}*: x has an equal number of substrings 01 and 10}.

Intuition: Need to remember difference in # of **01** or **10** substrings seen, but only hard to do if these are separated by **2**'s.

1. Let **S**={ε, 012, 012012, 012012012, ...} = {(012)^n : n ∈ ℕ}
2. Let **M** be an arbitrary DFA. Since **S** is infinite and **M** is finite state there must be two strings (012)^i and (012)^j for some i ≠ j that end up at the same state of **M**.
3. Consider appending string **t** = (102)^i **to each of these strings.**
   Then (012)^i (102)^i ∈ **L** but (012)^j (102)^i ∉ **L** since i ≠ j
4. So (012)^i (102)^i and (012)^j (102)^i end up at the same state of **M** since (012)^i and (012)^j do. Since (012)^i (102)^i ∈ **L** and (012)^j (102)^i ∉ **L**, **M** does not recognize **L**.
5. Since **M** was arbitrary, no DFA recognizes **L**.

DFAs ≡ Regular expressions

Theorem: A language is recognized by a DFA if and only if it has a regular expression

Proof: Last class: RegExp → NFA → DFA

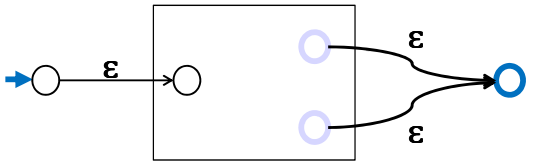
Now: NFA → RegExp
Enough since every DFA is also an NFA.

Generalized NFAs

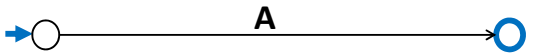
- Like NFAs but allow
  - Parallel edges
  - Regular Expressions as edge labels
    NFAs already have edges labeled ε or ***a***
- An edge labeled by **A** can be followed by reading a string of input chars that is in the language represented by **A**
- A string x is accepted iff there is a path from start to final state labeled by a regular expression whose language contains x

Starting from an NFA

Add new start state and final state



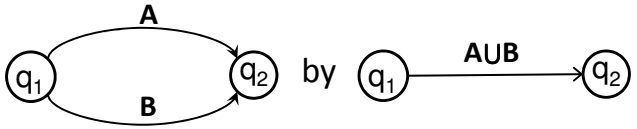
Then eliminate original states one by one, keeping the same language, until it looks like:



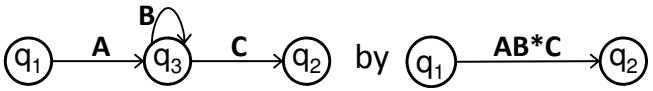
Final regular expression will be **A**

Only two simplification rules

- **Rule 1**: For any two states q_1 and q_2 with parallel edges (possibly $q_1=q_2$), replace



- **Rule 2**: Eliminate non-start/final state q_3 by replacing all

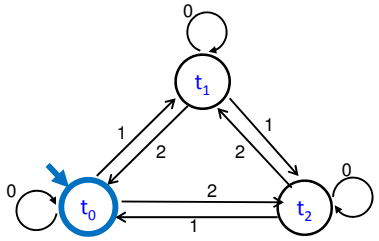


for *every* pair of states q_1, q_2 (even if $q_1=q_2$)

Converting an NFA to a regular expression

Consider the DFA for the mod 3 sum

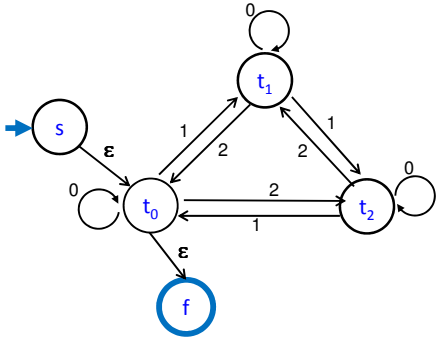
– Accept strings from {0,1,2}* where the digits mod 3 sum of the digits is 0



splicing out a node

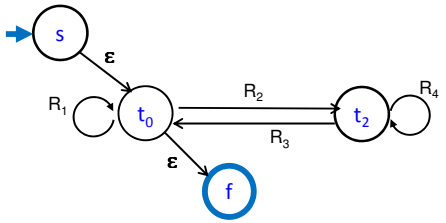
Label edges with regular expressions

$t_0 \to t_1 \to t_0$: 10*2
$t_0 \to t_1 \to t_2$: 10*1
$t_2 \to t_1 \to t_0$: 20*2
$t_2 \to t_1 \to t_2$: 20*1

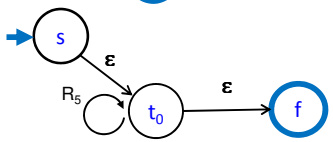


Finite automaton without t_1

R_1: 0 ∪ 10*2
R_2: 2 ∪ 10*1
R_3: 1 ∪ 20*2
R_4: 0 ∪ 20*1



R_5: $R_1 \cup R_2 R_4^* R_3$



Final regular expression:
(0 ∪ 10*2 ∪ (2 ∪ 10*1)(0 ∪ 20*1)*(1 ∪ 20*2))*