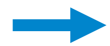
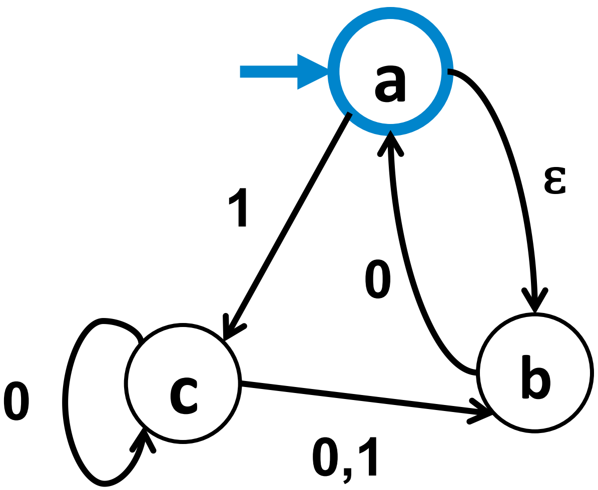
CSE 311: Foundations of Computing

Fall 2014

Lecture 25: Non-regularity and limits of FSMs

Example: NFA to DFA

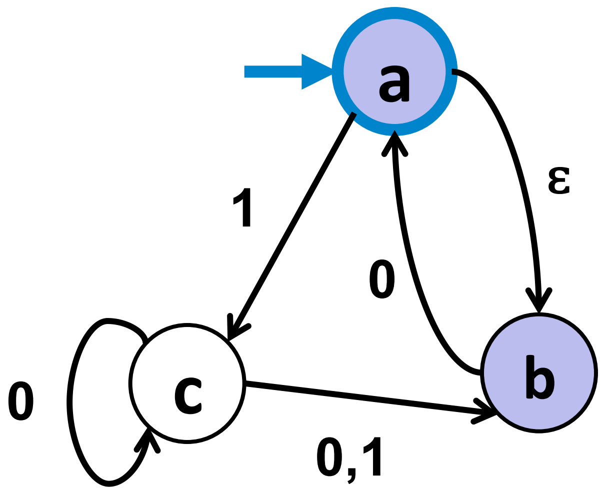




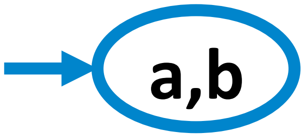
NFA

DFA

Example: NFA to DFA

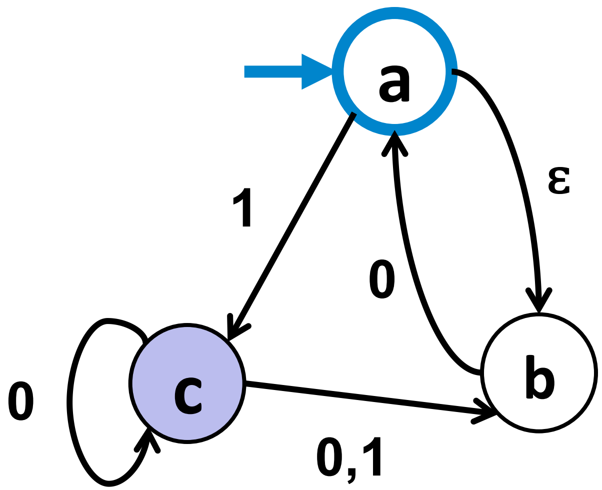


NFA

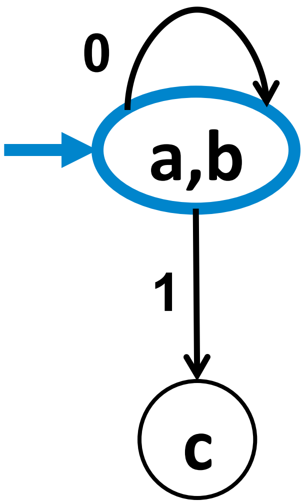


DFA

Example: NFA to DFA

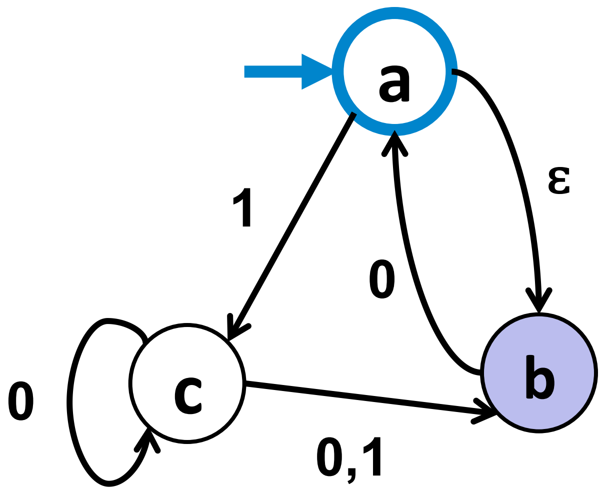


NFA

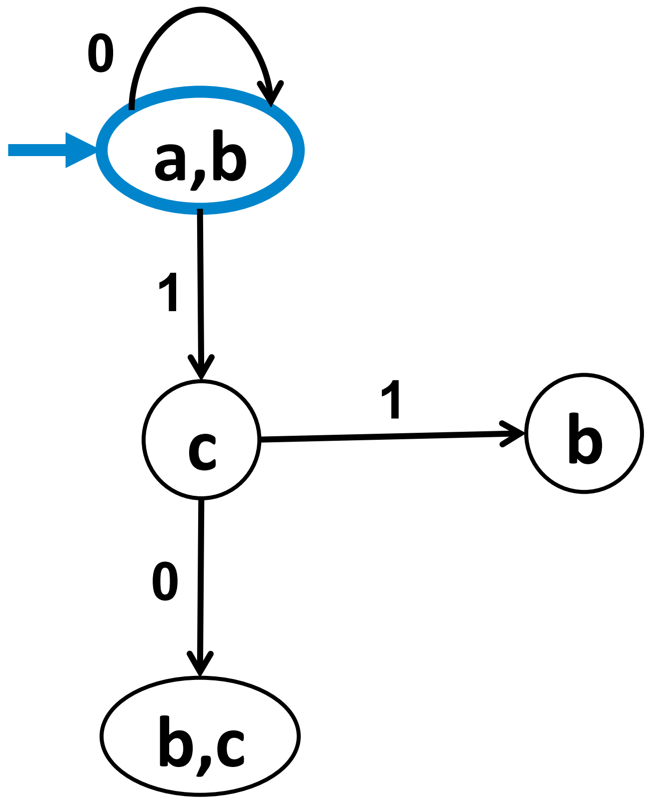


DFA

Example: NFA to DFA

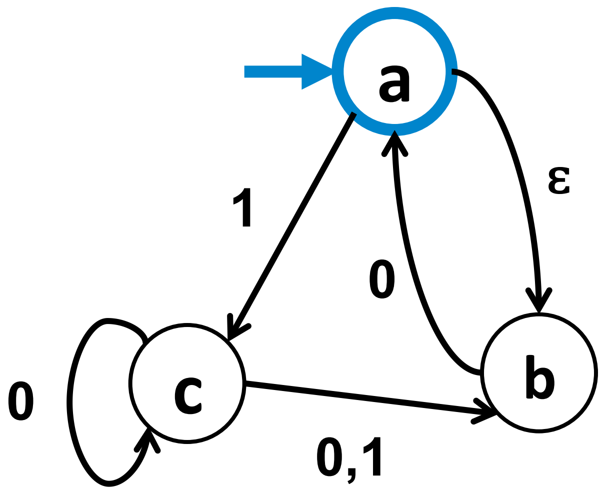


NFA

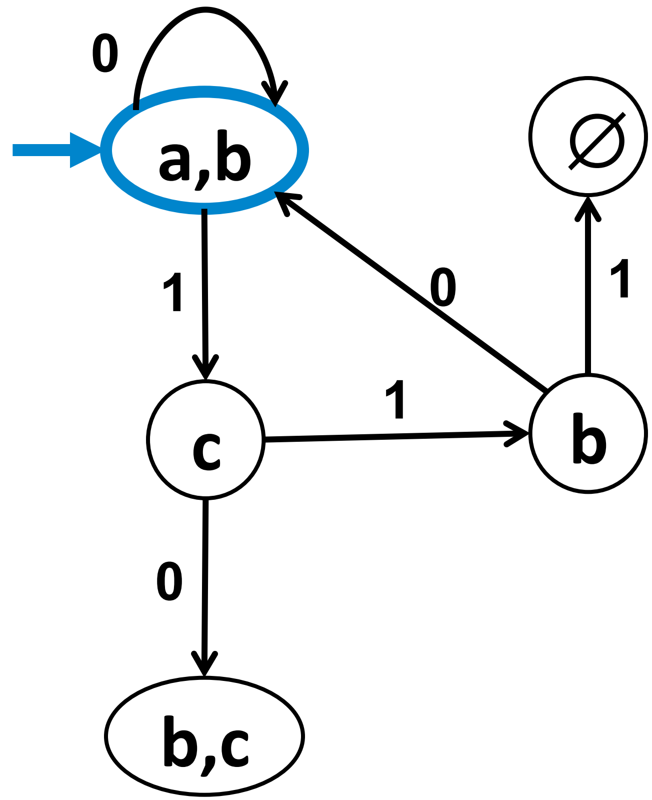


DFA

Example: NFA to DFA

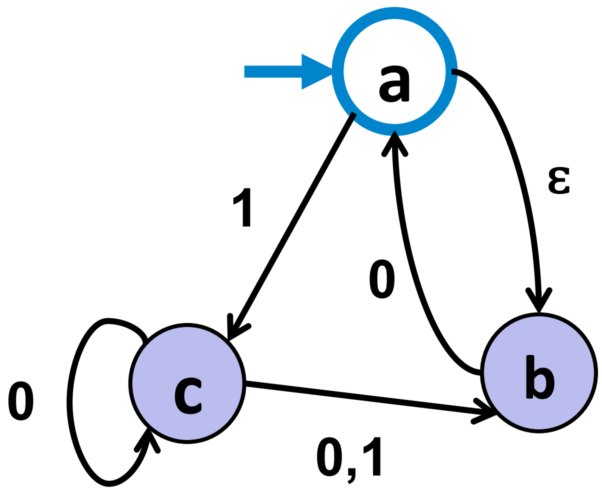


NFA

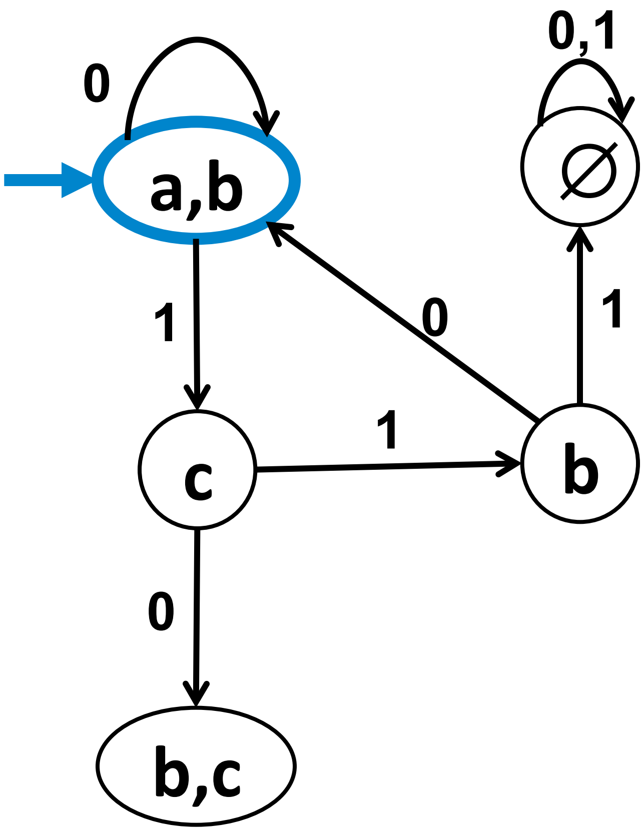


DFA

Example: NFA to DFA

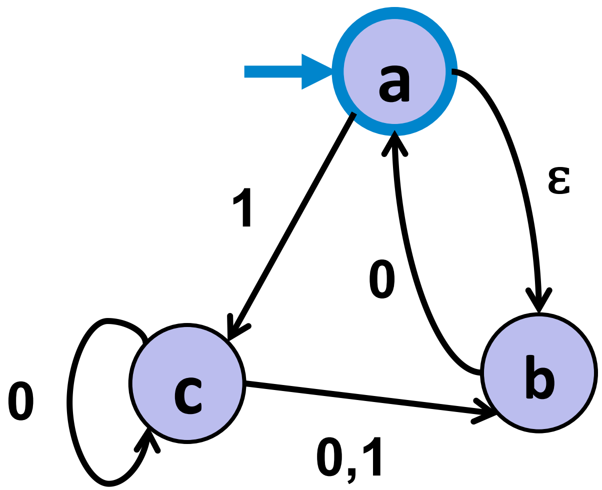


NFA

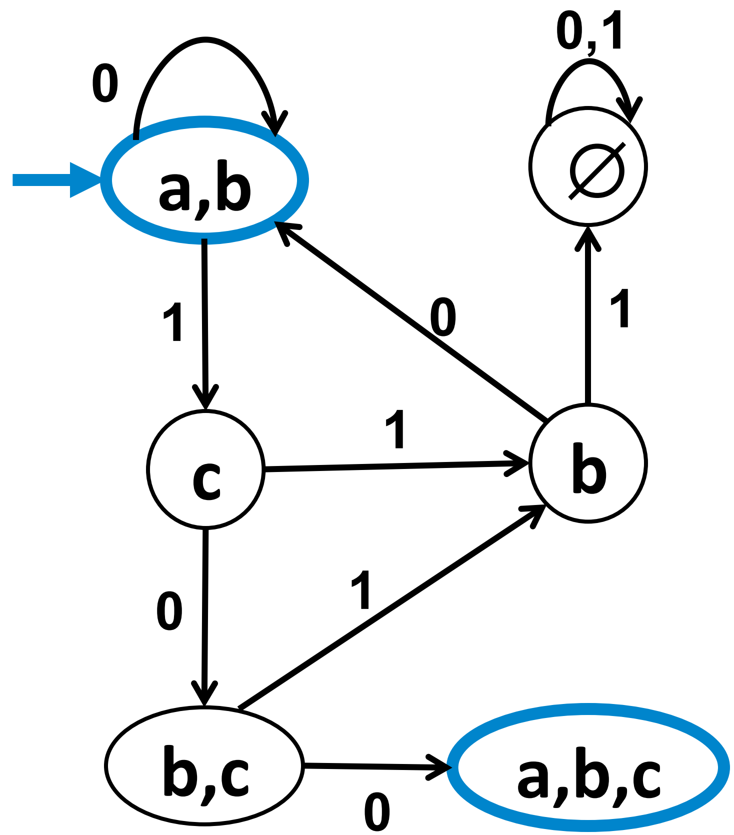


DFA

Example: NFA to DFA

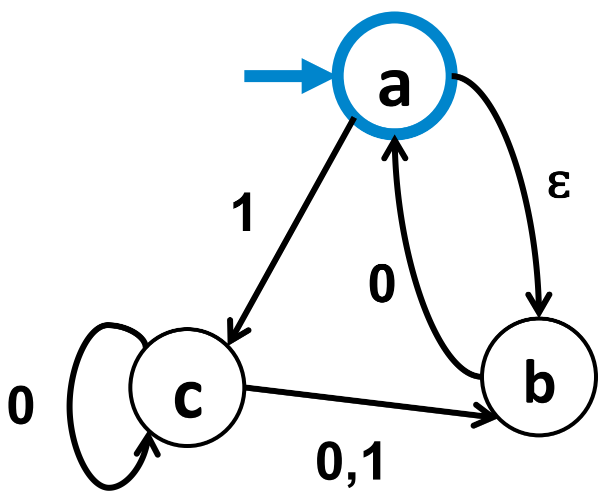


NFA

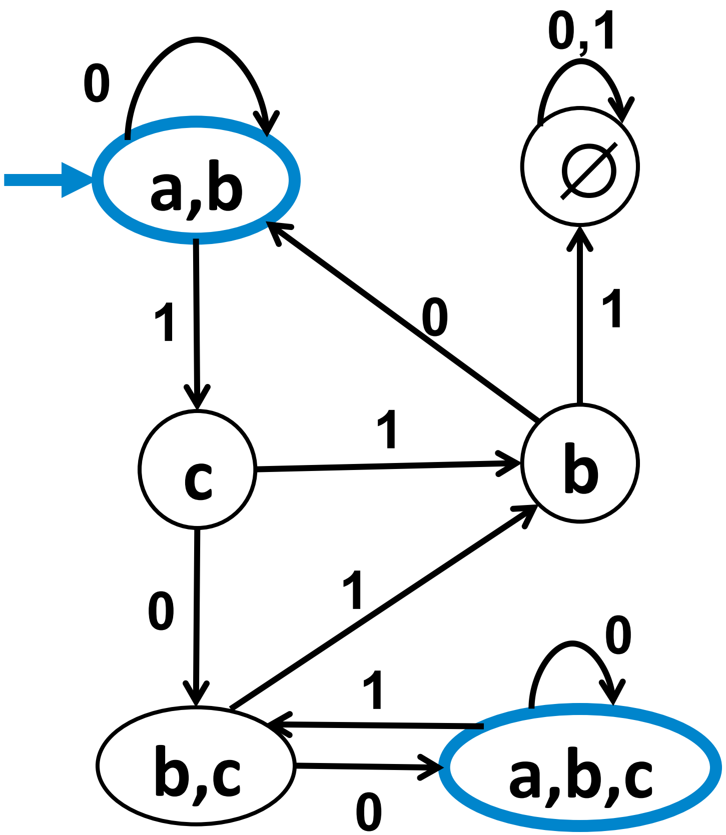


DFA

Example: NFA to DFA



NFA



DFA

Exponential Blow-up in Simulating Nondeterminism

- In general the DFA might need a state for every subset of states of the NFA
  - Power set of the set of states of the NFA
  - n-state NFA yields DFA with at most 2^n states
  - We saw an example where roughly 2^n is necessary

    Is the n^{th} char from the end a 1?

- The famous "P=NP?" question asks whether a similar blow-up is always necessary to get rid of nondeterminism for polynomial-time algorithms

DFAs ≡ Regular expressions

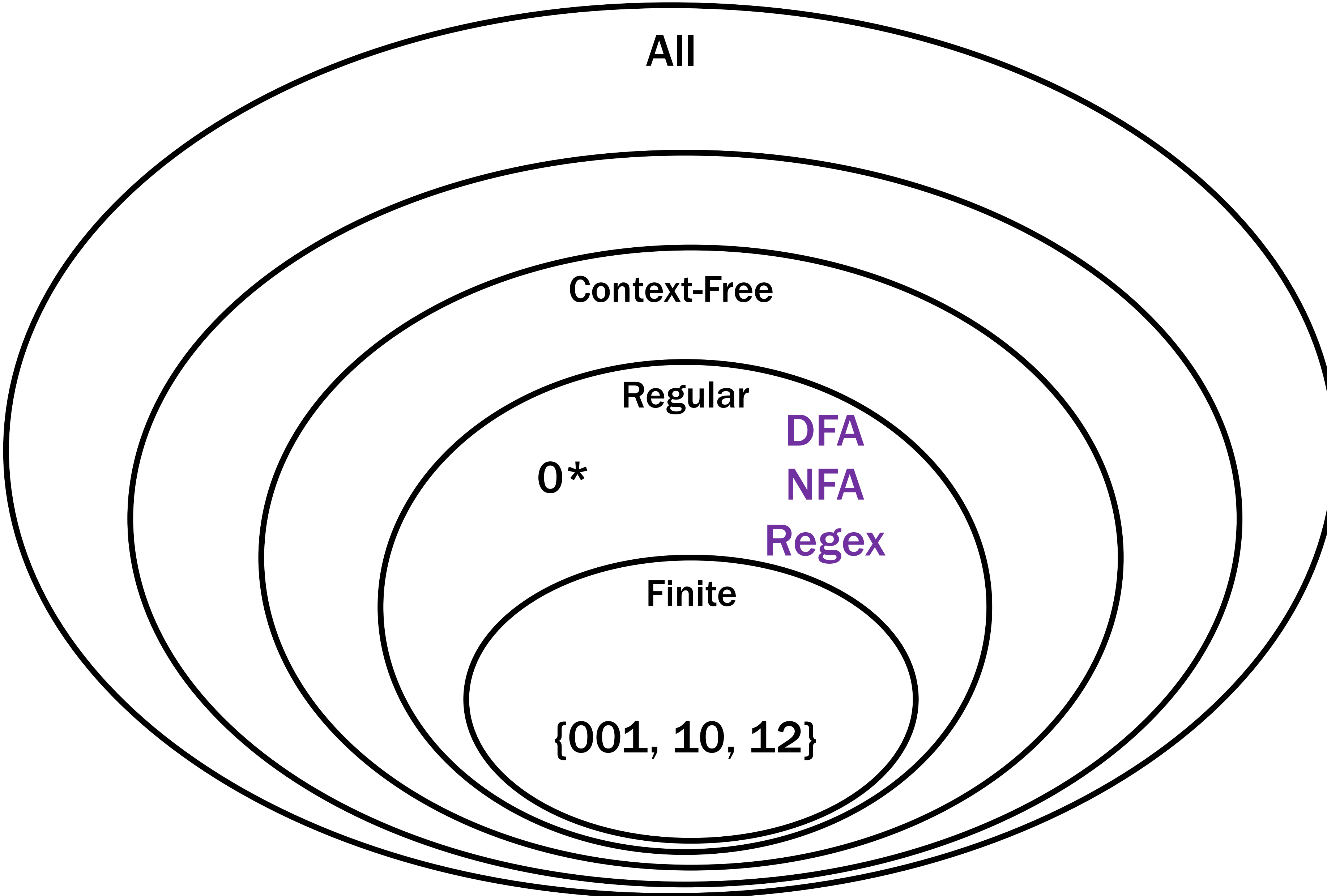
We have shown how to build an optimal DFA for every regular expression

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

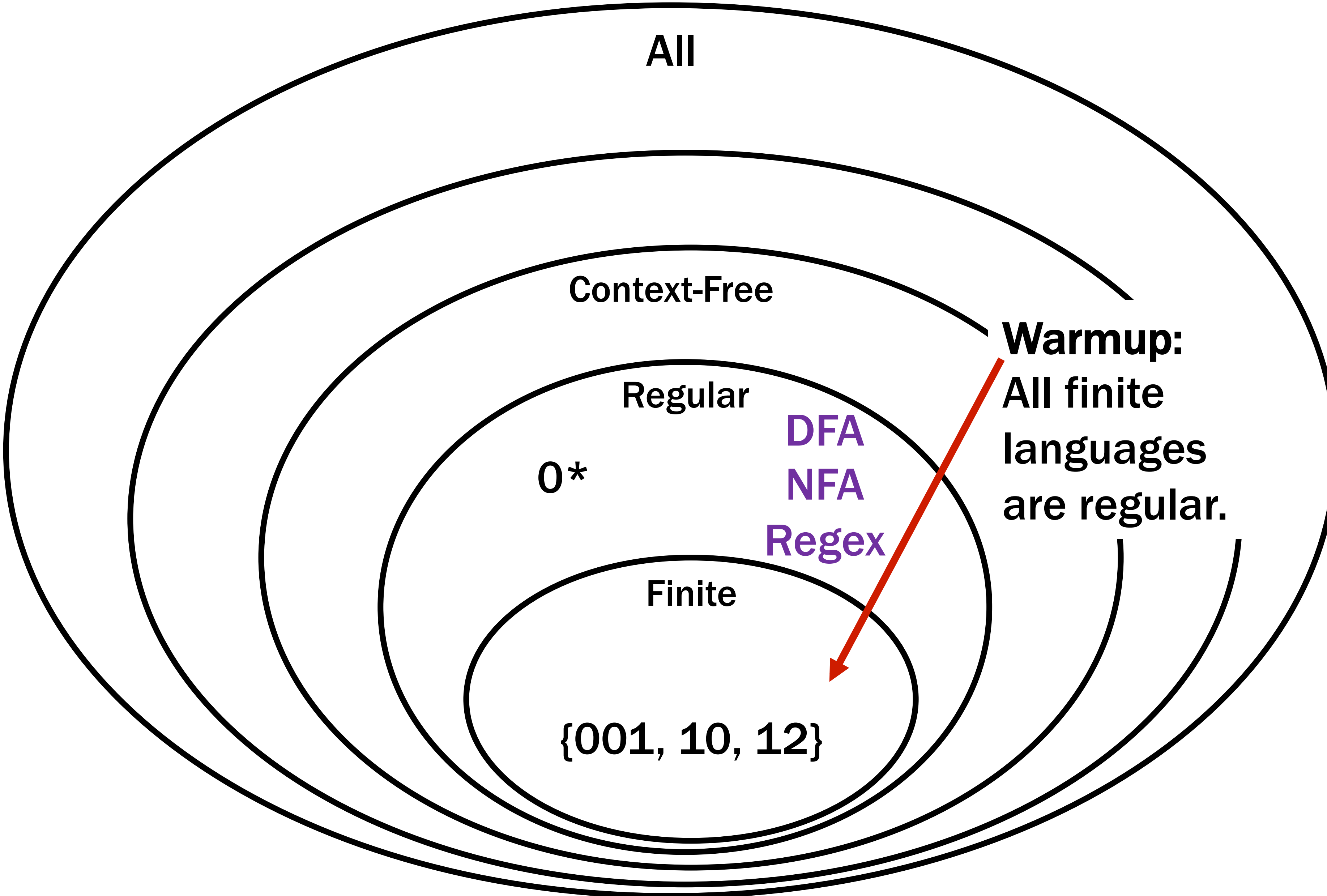
Theorem: A language is recognized by a DFA if and only if it has a regular expression

This direction will be completely untested. We will post slides, but we have more important things to discuss today.

Languages and Machines!



Languages and Machines!

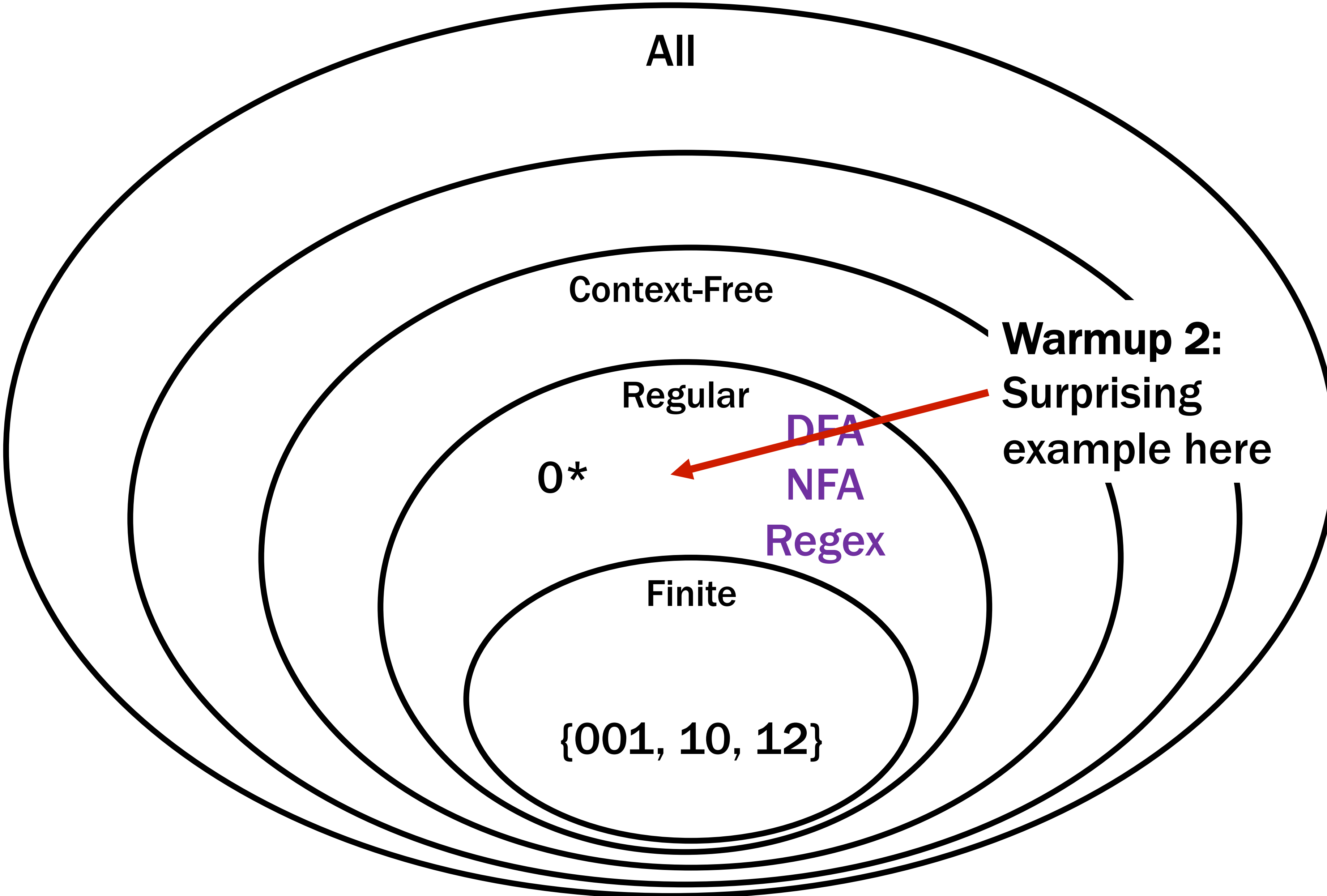


DFAs Recognize Any Finite Language

Construct DFAs for each string in the language.

Then, put them together using the union construction.

Languages and Machines!



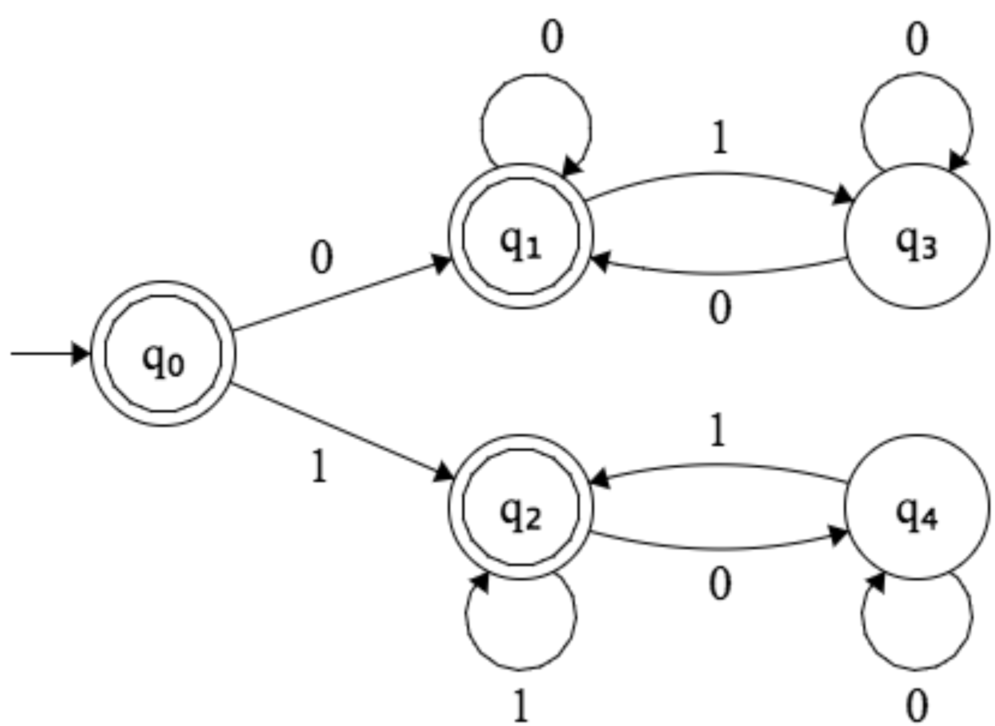
An Interesting Infinite Regular Language

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

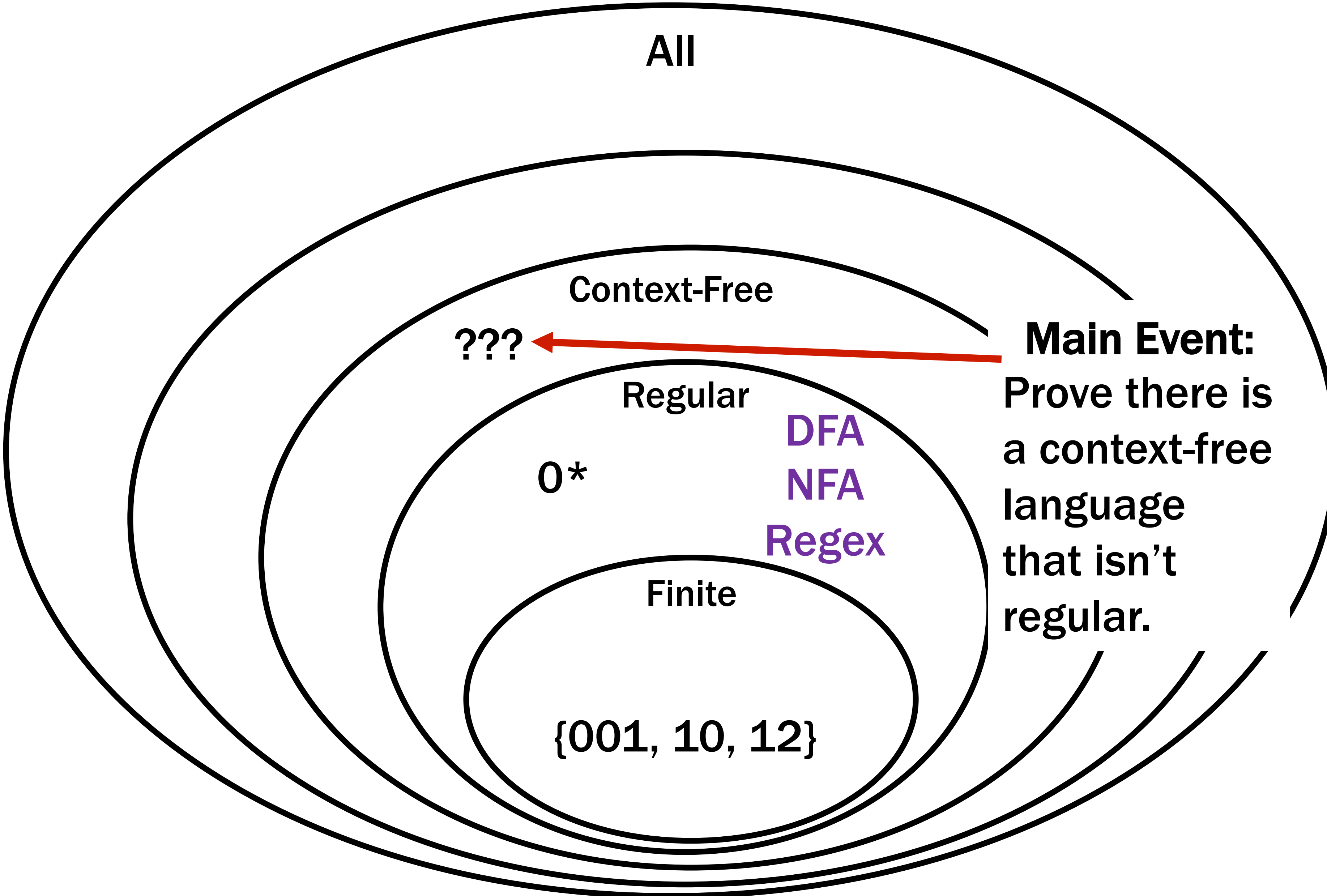
L is infinite.

0, 00, 000, ...

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 some arbitrary DFA. We want to show it doesn't work for our language.

Consider the infinite set of strings

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

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 $x \in B$ and some $y \notin B$ 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

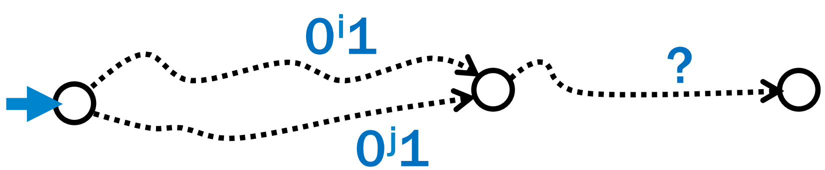
Suppose we are given an arbitrary DFA **M**.

Consider the infinite set of strings

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

- 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 ∈ **B** and 0^j10^i ∉ **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**=$\{s_0, s_1, \ldots, s_n, \ldots\}$ 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**."

Prove $A = \{0^n 1^n : n \geq 0\}$ is not regular

Let M be an arbitrary DFA.

Let $S = \{0^n : n \geq 0\}$. Since S is infinite and M has finitely many states, there must be two strings, 0^i and 0^j (for some $i \neq j$) that end in the same state in M.

Consider appending 1^i to both strings. Note that $0^i 1^i \in A$, but $0^j 1^i \notin A$ since $i \neq j$. But they both end up in the same state of M. Since that state can't be both an accept and reject state, M does not recognize A.

Since M was arbitrary, no DFA recognizes A.

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 $(012)^i(102)^i \in L$ but $(012)^j(102)^i \notin L$ 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 $(012)^i(102)^i \in L$ and $(012)^j(102)^i \notin L$, **M** does not recognize **L.**
5. Since **M** was arbitrary, no DFA recognizes **L.**