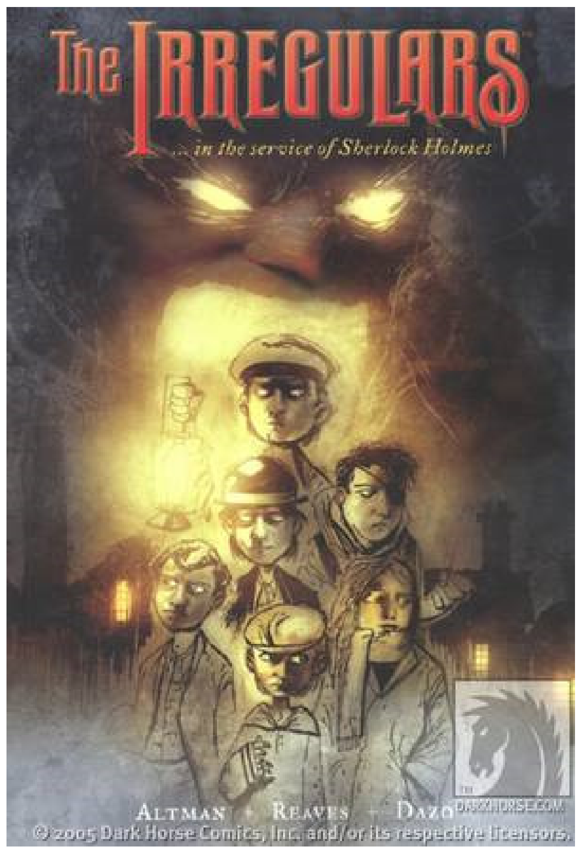
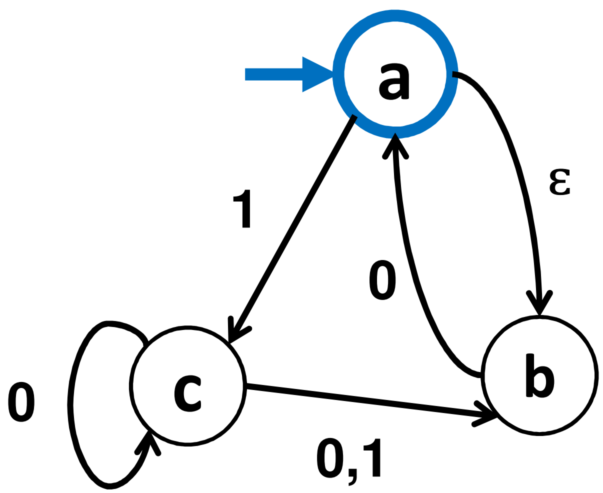
CSE 311: Foundations of Computing

Lecture 25: Languages vs Representations: Limitations of Finite Automata and Regular Expressions

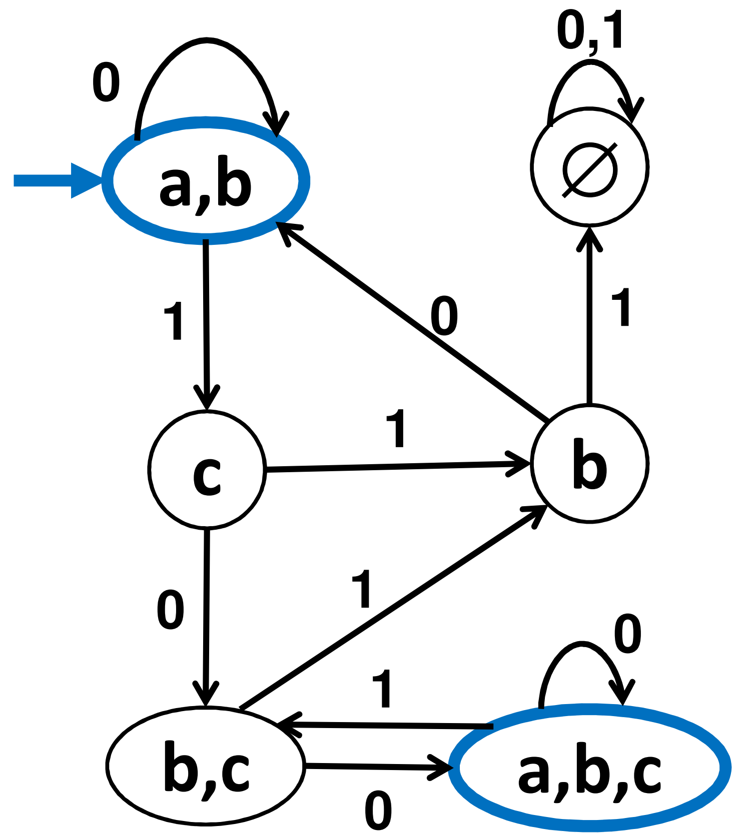


Pick HW 6 sol'ns if you don't have them

Last time: 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

Last time: DFAs ≡ NFAs ≡ 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 (or NFA) if and only if it has a regular expression

You need to know this fact but you don't need to know and we won't ask you anything about the construction for the "only if" direction from DFA/NFA to regular expression.

Regular Languages Rational Languages

Application of FSMs: Pattern matching

- Given
  - a string s of n characters
  - a pattern p of m characters
  - usually $m \ll n$
- Find
  - all occurrences of the pattern p in the string s

- Obvious algorithm:
  - try to see if p matches at each of the positions in s stop at a failed match and try matching at the next position: $O(mn)$ running time.

Application of FSMs: Pattern Matching

- With DFAs can do this in $O(m + n)$ time.

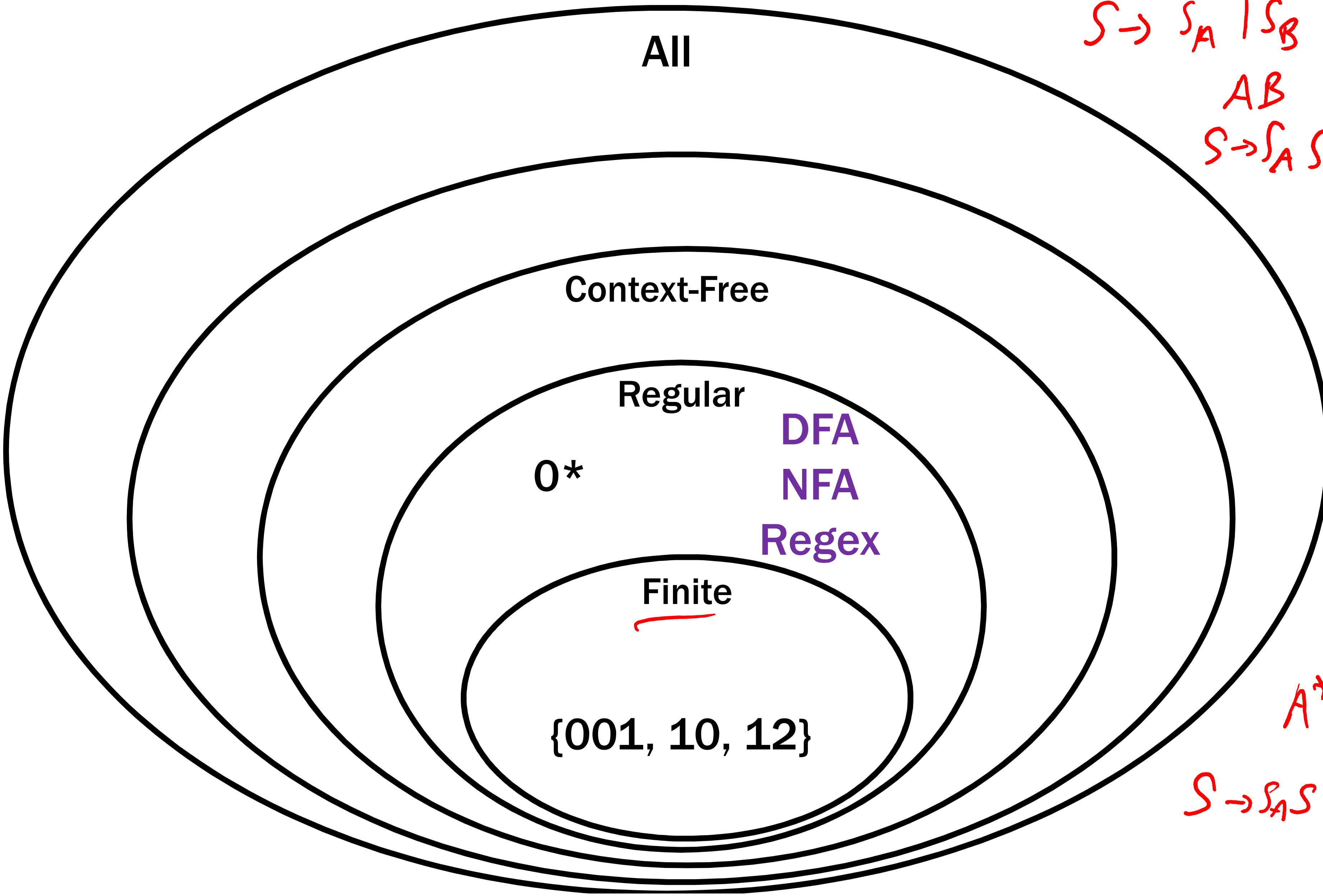
- Even more general idea in practice: implemented in regular expression pattern matchers like grep:
  - Convert regular expression pattern to an NFA
  - Start building the equivalent DFA from the NFA using the subset construction but do this "on the fly": only add arcs that are actually followed by the input text

- See Extra Credit problem on HW8 for some ideas of how to do it.

What languages have DFAs? CFGs?

All of them?

Languages and Representations!



$A \cup B$

$S \to S_A \mid S_B$

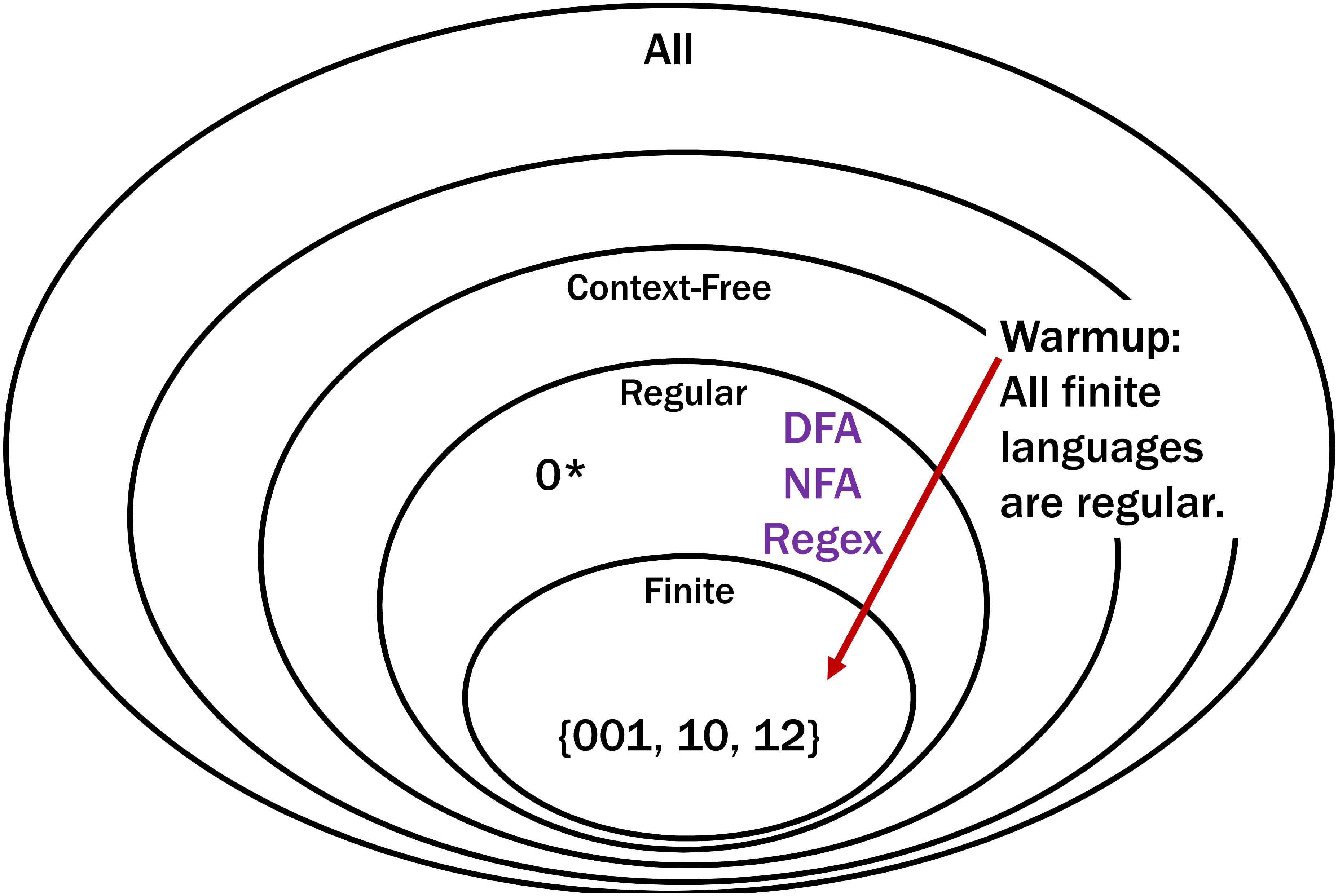
AB

$S \to S_A S_B$

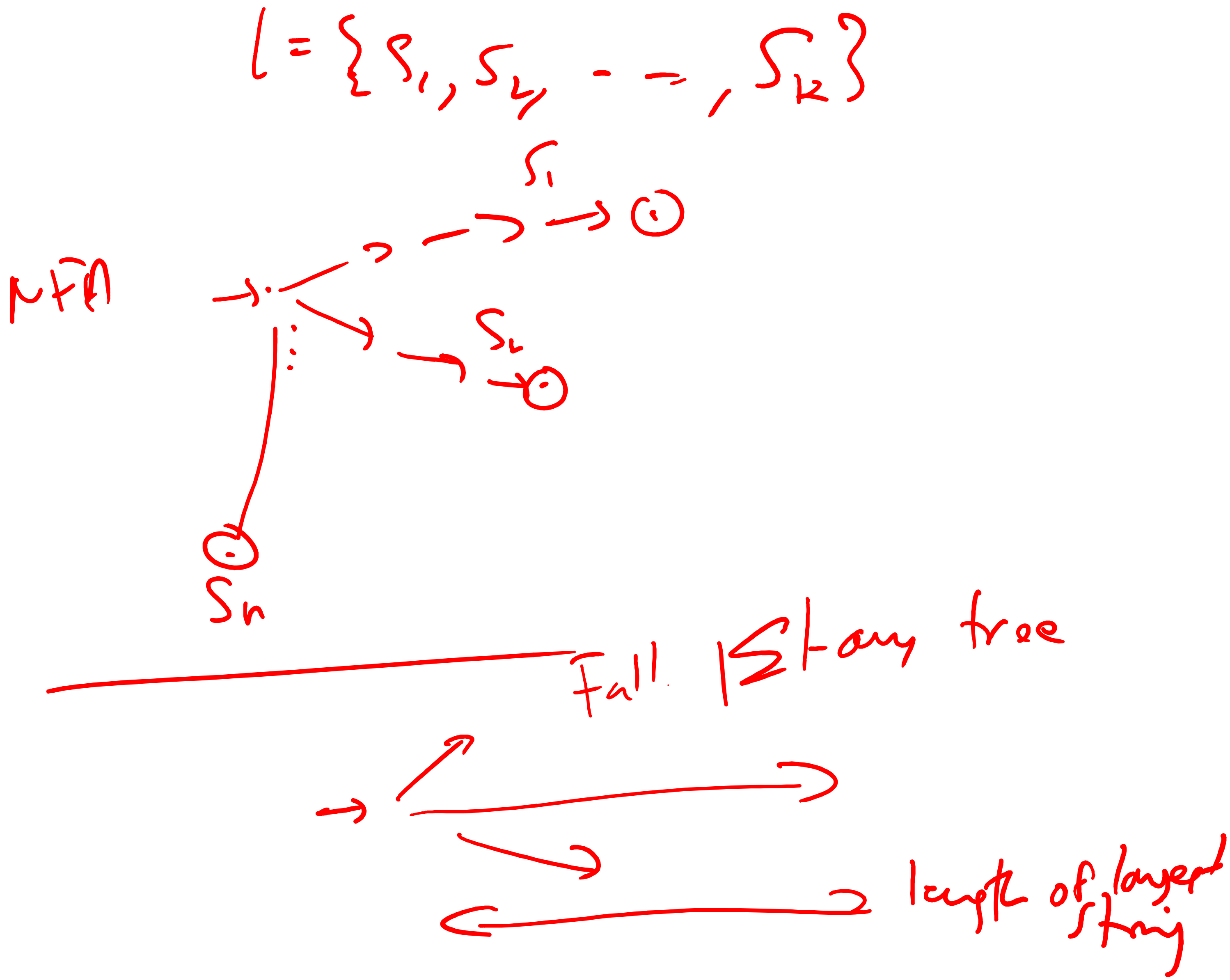
A^*

$S \to S_A S \mid \varepsilon$

Languages and Representations!



DFAs Recognize Any Finite Language

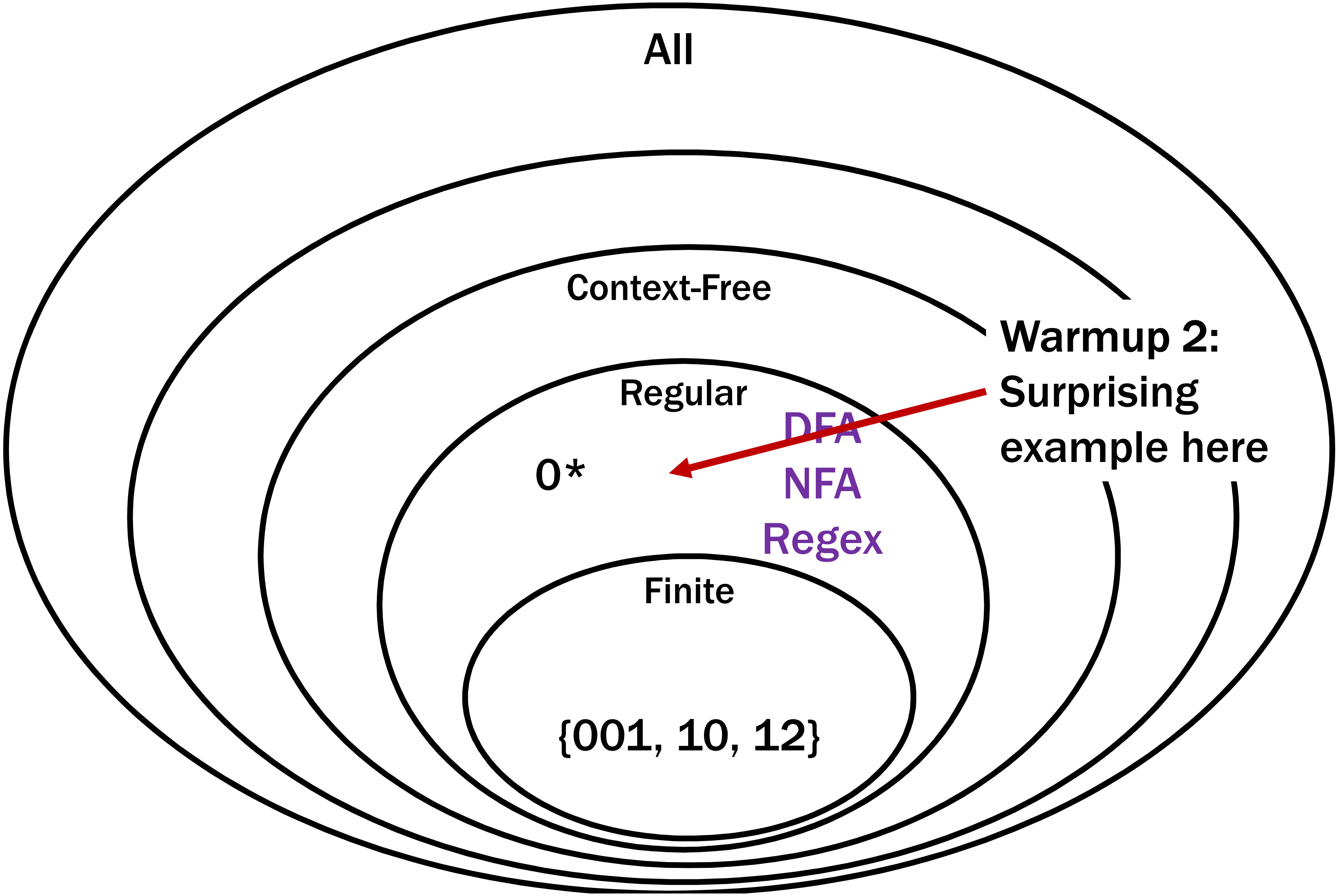


DFAs Recognize Any Finite Language

Construct a DFA 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. How could this be?

(It seems to be comparing counts and counting seems hard for DFAs.)

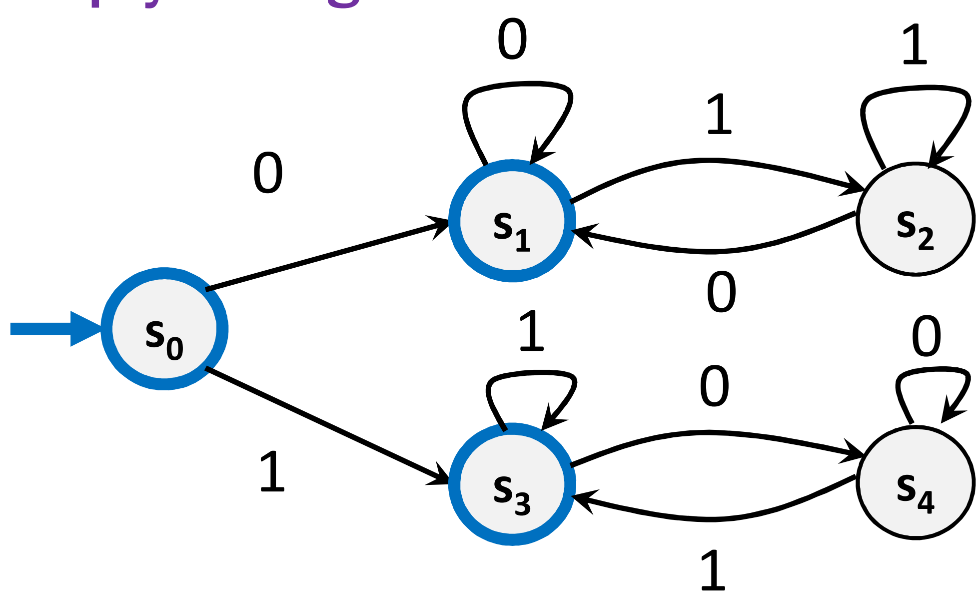
An Interesting Infinite Regular Language

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

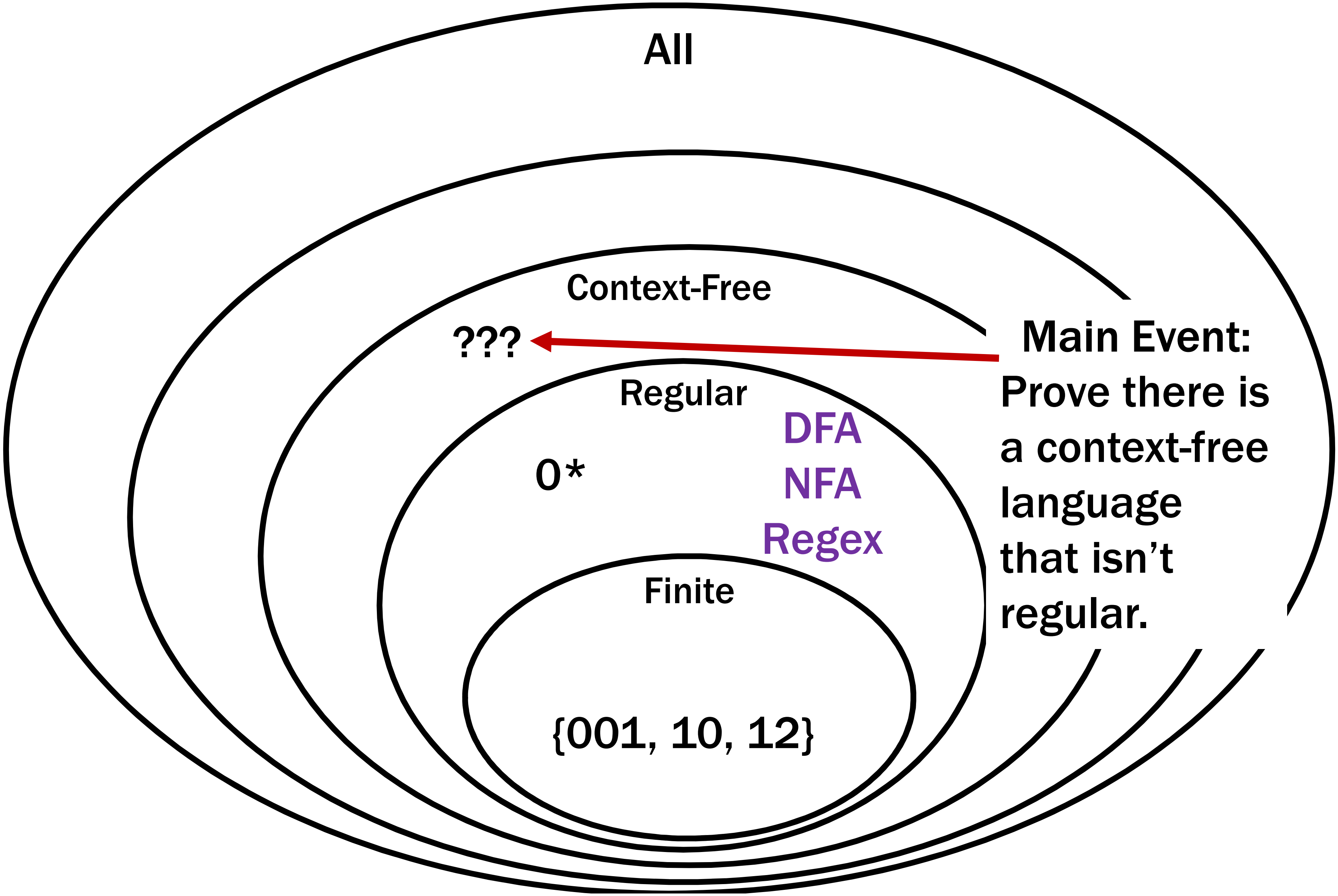
L is infinite.

0, 00, 000, ...

L is regular. How could this be? It is just the set of binary strings that are empty or begin and end with the same character!



Languages and Representations!



Main Event: Prove there is a context-free language that isn't regular.

The language of “Binary Palindromes” is Context-Free

$$S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$$

Is the language of “Binary Palindromes” Regular ?

Is the language of “Binary Palindromes” 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.

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

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B

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

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B
- Our goal is to show that M must be "confused"… we want to show that it "does the wrong thing".

How can a DFA be "wrong"?

- when it accepts or rejects a string it shouldn't.

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

The general proof strategy is:

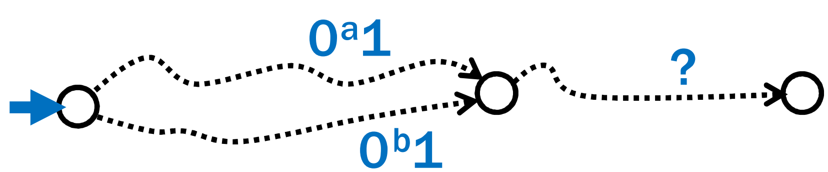
- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B
- Our goal is to show that M must be "confused"… we want to show that it "~~does the wrong thing~~" accepts or rejects a string it shouldn't.

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

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B
- We want to show: M accepts or rejects a string it shouldn't.

Key Idea 1: If two strings "collide" at any point, a DFA can no longer distinguish between them!

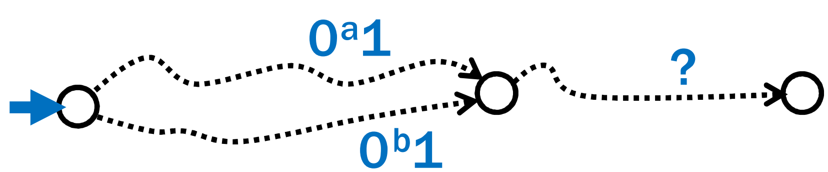


B = {binary palindromes} can't be recognized by any DFA

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B
- We want to show: M accepts or rejects a string it shouldn't.

Key Idea 1: If two strings "collide" at any point, a DFA can no longer distinguish between them!



Key Idea 2: Our machine M has a finite number of states which means if we have infinitely many strings, two of them must collide!

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

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that recognizes B
- We want to show: M accepts or rejects a string it shouldn't.

We choose an INFINITE set S of "partial strings" (which we intend to complete later). It is imperative that for *every pair* of strings in our set there is an "accept" completion that the two strings DO NOT SHARE.

1_____ ε
01_____ 0
001_____ 00
0001_____ 000
00001_____ 0000
....

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

Suppose for contradiction that some DFA, M, recognizes B.

We show M accepts or rejects a string it shouldn't.

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

Key Idea 2: Our machine has a finite number of states which means if we have infinitely many strings, two of them must collide!

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

Suppose for contradiction that some DFA, M, recognizes B.

We show M accepts or rejects a string it shouldn't.

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

Since there are finitely many states in M and infinitely many strings in S***, there exist strings*** $0^a1 \in S$ ***and*** $0^b1 \in S$ ***with*** $a \neq b$ ***that end in the same state of M.***

SUPER IMPORTANT POINT: You do not get to choose what a **and** b **are. Remember, we've just proven they exist…we have to take the ones we're given!**

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

Suppose for contradiction that some DFA, M, recognizes B.

We show M accepts or rejects a string it shouldn't.

Consider $S = \{0^n1 : n \geq 0\}$.

Since there are finitely many states in M and infinitely many strings in S, there exist strings $0^a1 \in S$ and $0^b1 \in S$ with $a \neq b$ that end in the same state of M.

SUPER IMPORTANT POINT: You do not get to choose what a and b are. Remember, we've just proven they exist…we have to take the ones we're given!

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

Suppose for contradiction that some DFA, M, accepts B.

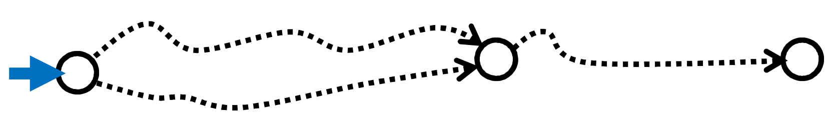
We show M accepts or rejects a string it shouldn't.

Consider $S = \{0^n1 : n \geq 0\}$.

Since there are finitely many states in M and infinitely many strings in S, there exist strings $0^a1 \in S$ and $0^b1 \in S$ with $a \neq b$ that end in the same state of M.

Now, consider appending 0^a to both strings.

Key Idea 1: If two strings "collide" at any point, a DFA can no longer distinguish between them!



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

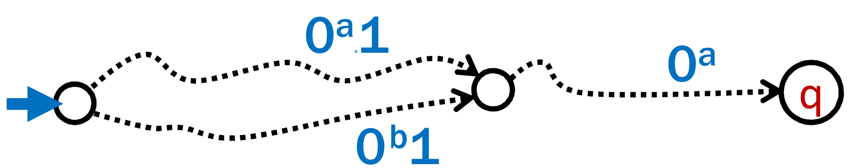
Suppose for contradiction that some DFA, M, recognizes B.

We show M accepts or rejects a string it shouldn't.

Consider $S = \{0^n 1 : n \geq 0\}$.

Since there are finitely many states in M and infinitely many strings in S, there exist strings $0^a 1 \in S$ **and** $0^b 1 \in S$ **with** $a \neq b$ **that end in the same state of M.**

Now, consider appending 0^a **to both strings.**



Then, since $0^a 1$ ***and*** $0^b 1$ ***end in the same state,*** $0^a 1 0^a$ ***and*** $0^b 1 0^a$ ***also end in the same state, call it q. But then M must make a mistake: q needs to be an accept state since*** $0^a 1 0^a \in B$***, but then M would accept*** $0^b 1 0^a \notin B$ ***which is an error.***

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

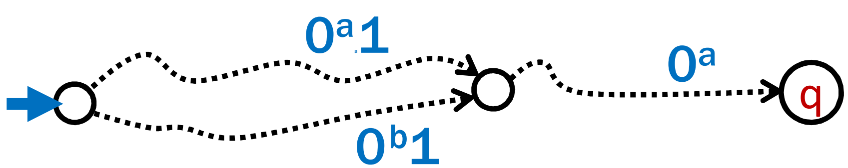
Suppose for contradiction that some DFA, M, recognizes B.

We show M accepts or rejects a string it shouldn't.

Consider $S = \{0^n1 : n \geq 0\}$.

Since there are finitely many states in M and infinitely many strings in S, there exist strings $0^a1 \in S$ and $0^b1 \in S$ with $a \neq b$ that end in the same state of M.

Now, consider appending 0^a to both strings.



Then, since 0^a1 and 0^b1 end in the same state, 0^a10^a and 0^b10^a also end in the same state, call it q. But then M must make a mistake: q needs to be an accept state since $0^a10^a \in B$, but then M would accept $0^b10^a \notin B$ which is an error.

This is a contradiction, since we assumed that M recognizes B. Since M was arbitrary, there is no DFA that recognizes B. ■

Showing that a Language L is not regular

1. "Suppose for contradiction that some DFA M recognizes L."
2. Consider an INFINITE set S of "partial strings" (which we intend to complete later). It is imperative that for *every pair* of strings in our set there is an "accept" completion that the two strings DO NOT SHARE.
3. "Since S is infinite and M has finitely many states, there must be two strings s_a and s_b in S for $s_a \neq s_b$ that end up at the same state of M."
4. Consider appending the (correct) completion t to each of the two strings.
   will depend on s_a and s_b
5. "Since s_a and s_b both end up at the same state of M, and we appended the same string t, both $s_a t$ and $s_b t$ end at the same state q of M. Since $s_a t \in L$ and $s_b t \notin L$, M does not recognize L."
6. "Since M was arbitrary, no DFA recognizes L."

Prove $A = \{0^n1^n : n \geq 0\}$ is not regular

Suppose for contradiction that some DFA, **M**, recognizes **A**.

Let **S** = $\{\varepsilon, 0, 00, 000, \ldots\} = \{0^n : n \geq 0\}$

$\therefore \exists\, a \neq b$ s.t. $0^a \in S$, $0^b \in S$ end at the same state of M.

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

Suppose for contradiction that some DFA, M, recognizes A.

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

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

Suppose for contradiction that some DFA, M, recognizes A.

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

Consider appending 1^a to both strings.

$0^a 1^a \in A \qquad 0^a 1^b \notin A.$

Prove A = $\{0^n1^n : n \geq 0\}$ is not regular

Suppose for contradiction that some DFA, M, recognizes A.

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

Consider appending 1^a to both strings.

Note that $0^a1^a \in$ A, but $0^b1^a \notin$ A since $a \neq b$. But they both end up in the same state of M, call it q. Since $0^a1^a \in$ A, state q must be an accept state but then M would incorrectly accept $0^b1^a \notin$ A so M does not recognize A.

Since M was arbitrary, no DFA recognizes A.

Prove P = {balanced parentheses} **is not regular**

Suppose for contradiction that some DFA, M, accepts P.

Let S =

Prove P = {balanced parentheses} is not regular

Suppose for contradiction that some DFA, M, recognizes P.

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

$(^a)^b \notin A$

$b \neq a$

$(^b)^b \in A$

Prove P = {balanced parentheses} **is not regular**

Suppose for contradiction that some DFA, M, recognizes P.

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

Consider appending $)^a$ **to both strings.**

Prove P = {balanced parentheses} is not regular

Suppose for contradiction that some DFA, M, recognizes P.

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

Consider appending $)^a$ to both strings.

Note that $(^a)^a \in P$, but $(^b)^a \notin P$ since $a \neq b$. But they both end up in the same state of M, call it q. Since $(^a)^a \in P$, state q must be an accept state but then M would incorrectly accept $(^b)^a \notin P$ so M does not recognize P.

Since M was arbitrary, no DFA recognizes P.

Showing that a Language L is not regular

1. "Suppose for contradiction that some DFA M recognizes L."
2. Consider an INFINITE set S of "partial strings" (which we intend to complete later). It is imperative that for *every pair* of strings in our set there is an "accept" completion that the two strings DO NOT SHARE.
3. "Since S is infinite and M has finitely many states, there must be two strings s_a and s_b in S for $s_a \neq s_b$ that end up at the same state of M."
4. Consider appending the (correct) completion t to each of the two strings.
5. "Since s_a and s_b both end up at the same state of M, and we appended the same string t, both $s_a t$ and $s_b t$ end at the same state q of M. Since $s_a t \in L$ and $s_b t \notin L$, M does not recognize L."
6. "Since M was arbitrary, no DFA recognizes L."

Fact: This method is optimal

- Suppose that for a language L, the set S is a *largest* set of "partial strings" with the property that for every pair $s_a \neq s_b \in S$, there is some string t such that one of $s_a t$, $s_b t$ is in L but the other isn't.
- If S is infinite then L is not regular
- If S is finite then the minimal DFA for L has precisely $|S|$ states, one reached by each member of S.

BTW: There is another method commonly used to prove languages not regular called the Pumping Lemma that we won't use in this course. Note that it doesn't always work.