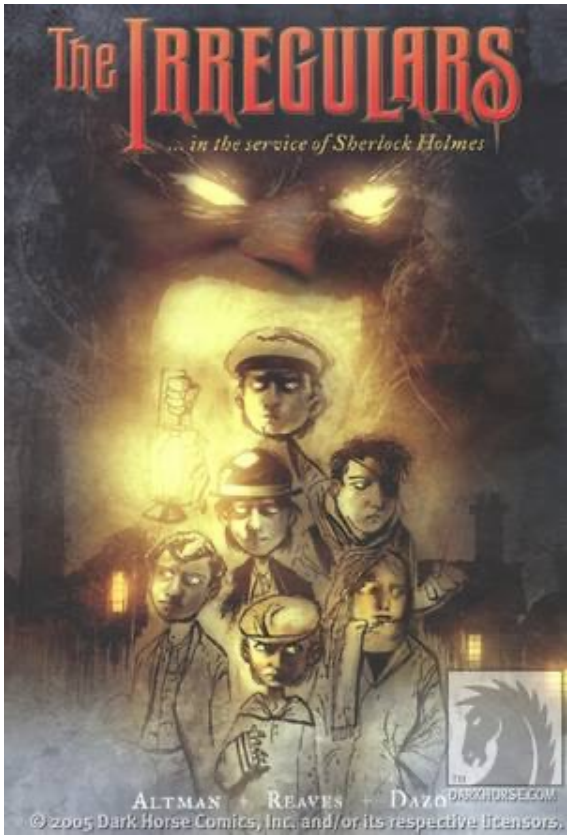
CSE 311: Foundations of Computing

Lecture 26: Languages vs Representations: Limitations of Finite Automata and Regular Expressions



Last time: Algorithms for Regular Languages

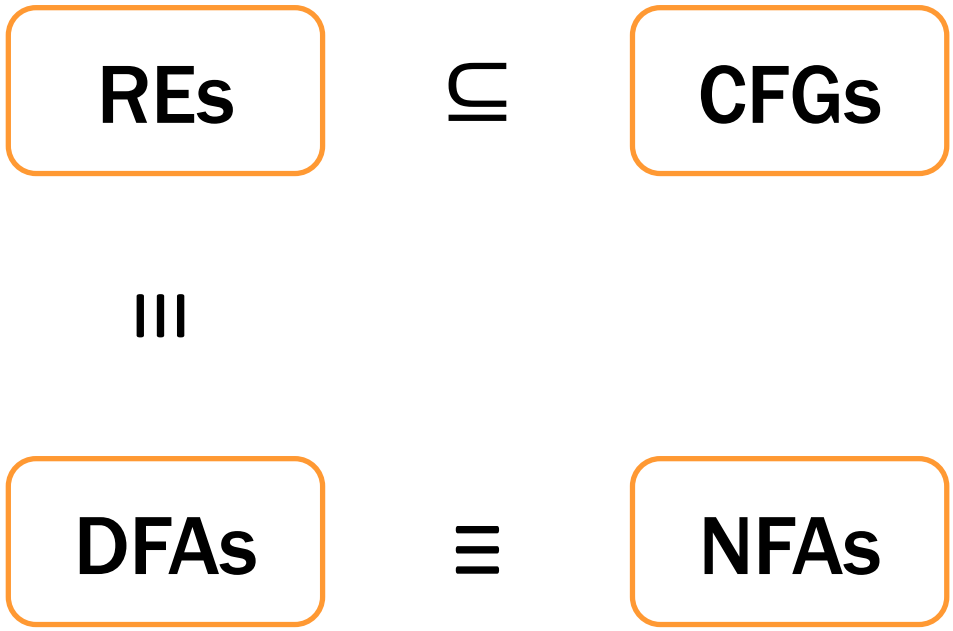
We have seen algorithms for

- RE to NFA
- NFA to DFA
- DFA/NFA to RE (not tested)
- DFA minimization

Practice three of these in HW.

(May also be on the final.)

The story so far...



Languages represented by DFA, NFAs, or regular expressions are called **Regular Languages**

Regular expressions ≡ NFAs ≡ DFAs

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

Thus, we could now implement a RegExp library

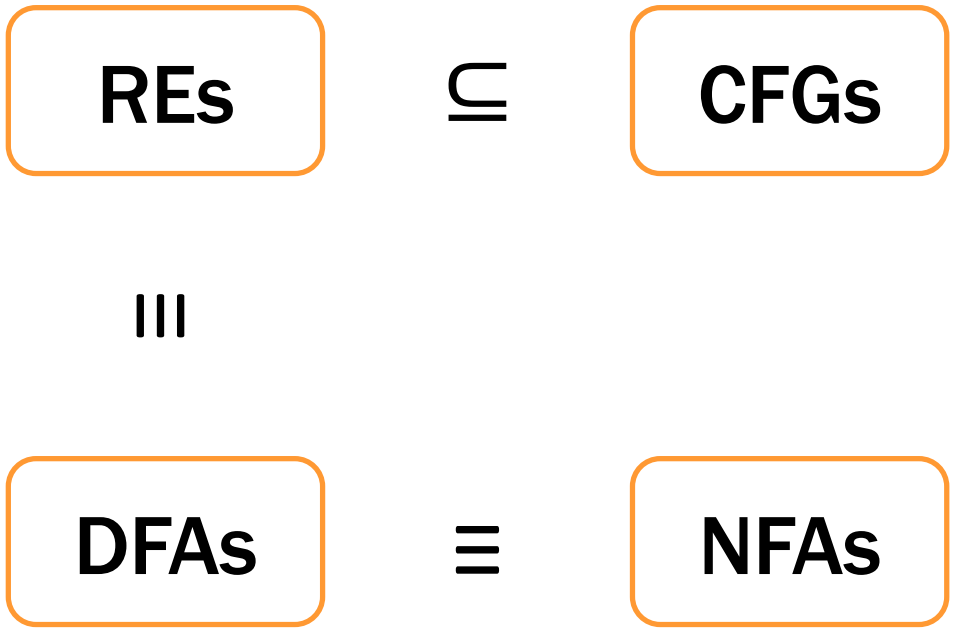
- most RegExp libraries actually simulate the NFA
- (even better: one can combine the two approaches: apply DFA minimization lazily while simulating the NFA)

Example Corollary of These Results

Corollary: If $\mathbf{A}$ is the language of a regular expression, then $\overline{\mathbf{A}}$ is the language of a regular expression*.

(This is the complement with respect to the universe of all strings over the alphabet, i.e., $\overline{\mathbf{A}} = \boldsymbol{\Sigma}^* \setminus \mathbf{A}$.)

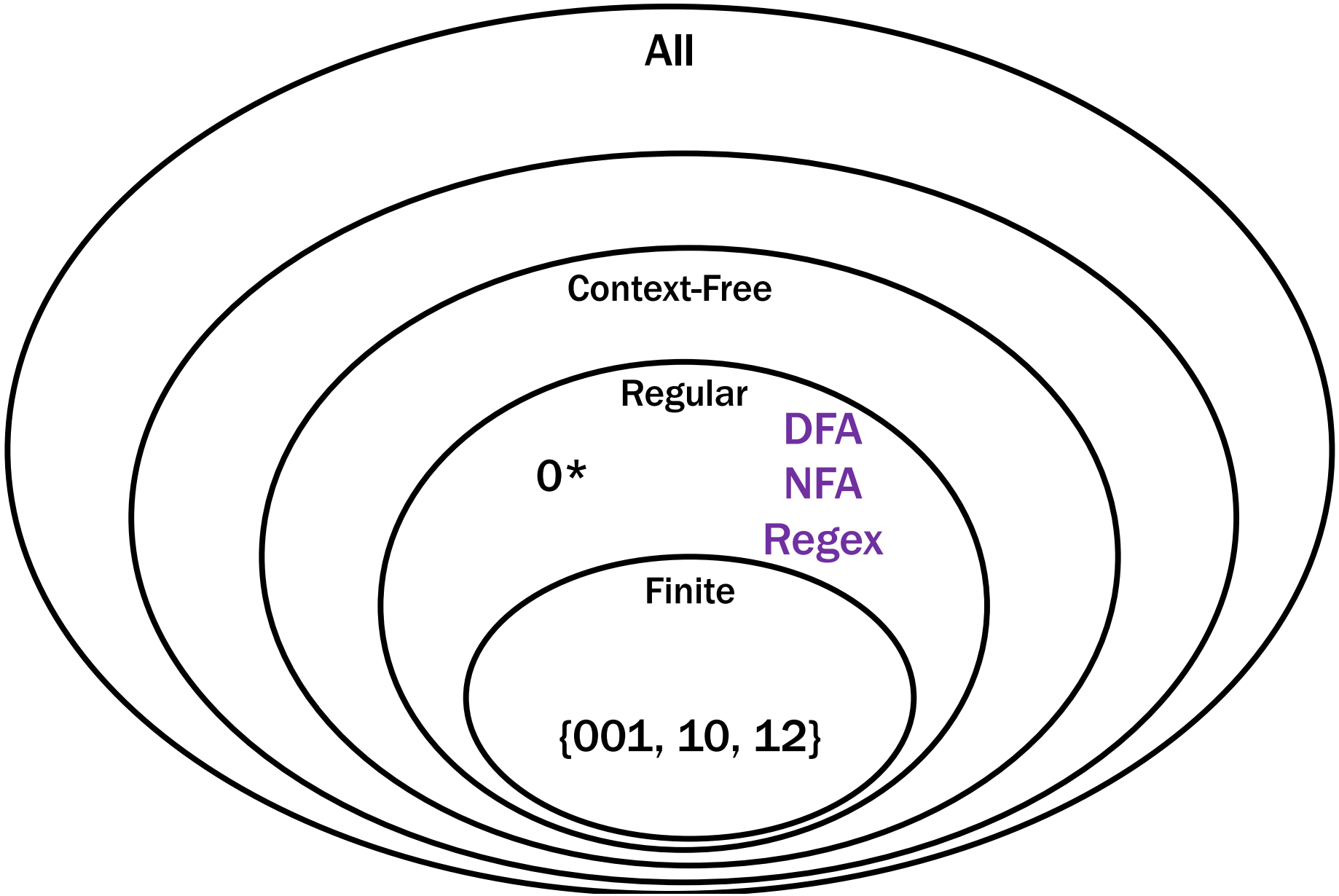
The story so far...



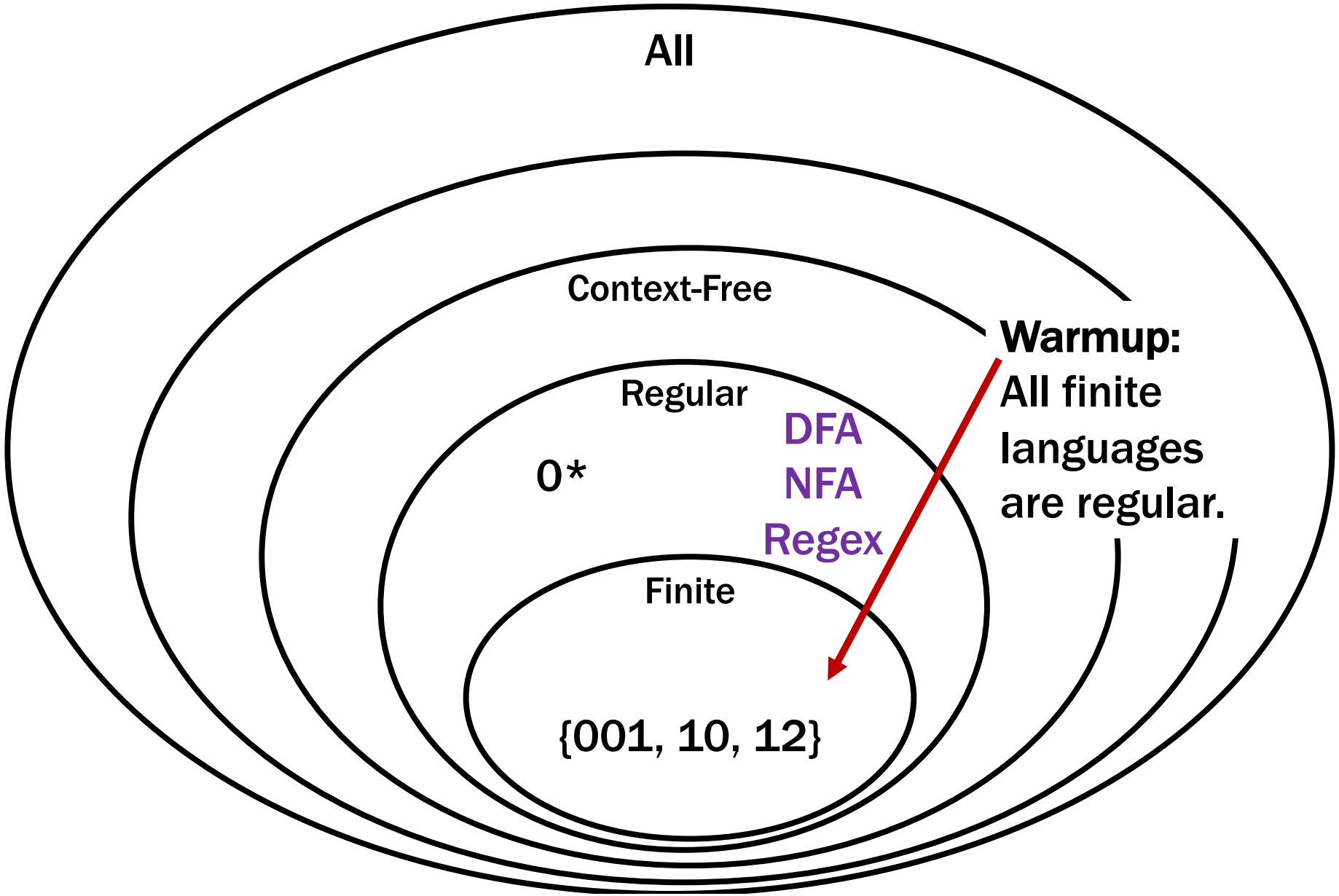
What languages have DFAs? CFGs?

All of them?

Languages and Representations!



Languages and Representations!



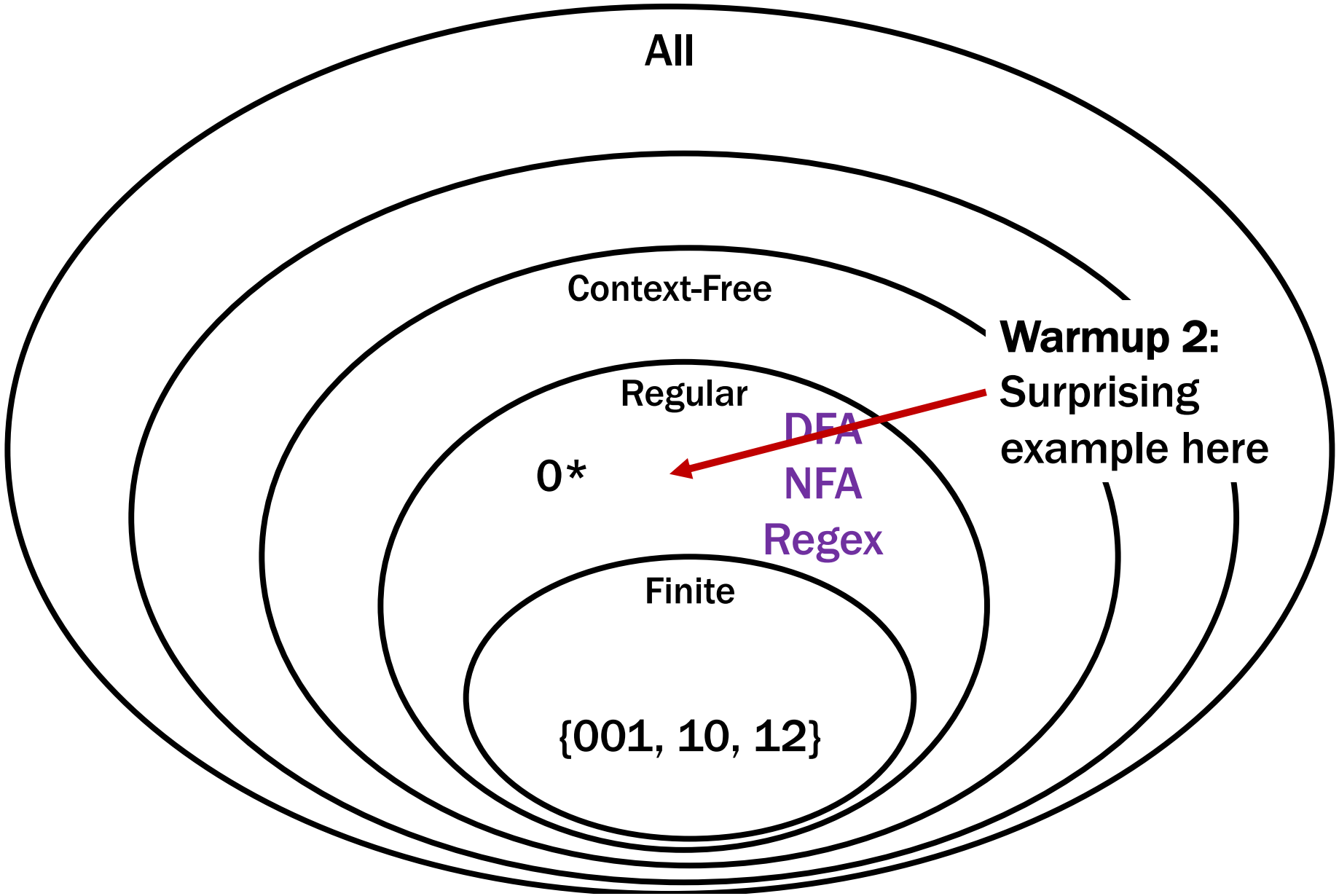
DFAs Recognize Any Finite Language

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?

That seems to require comparing counts…

- easy for a CFG
- but seems hard for DFAs!

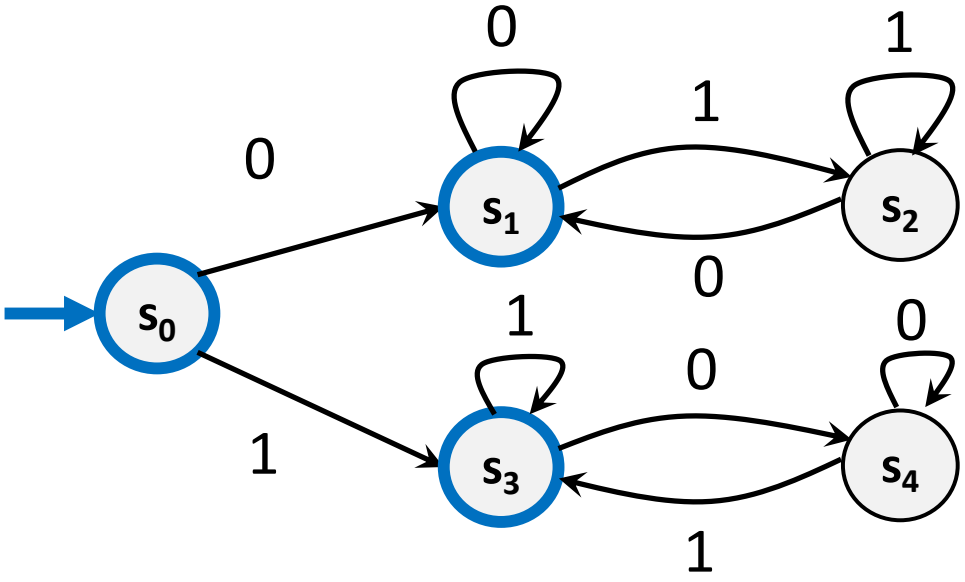
An Interesting Infinite Regular Language

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

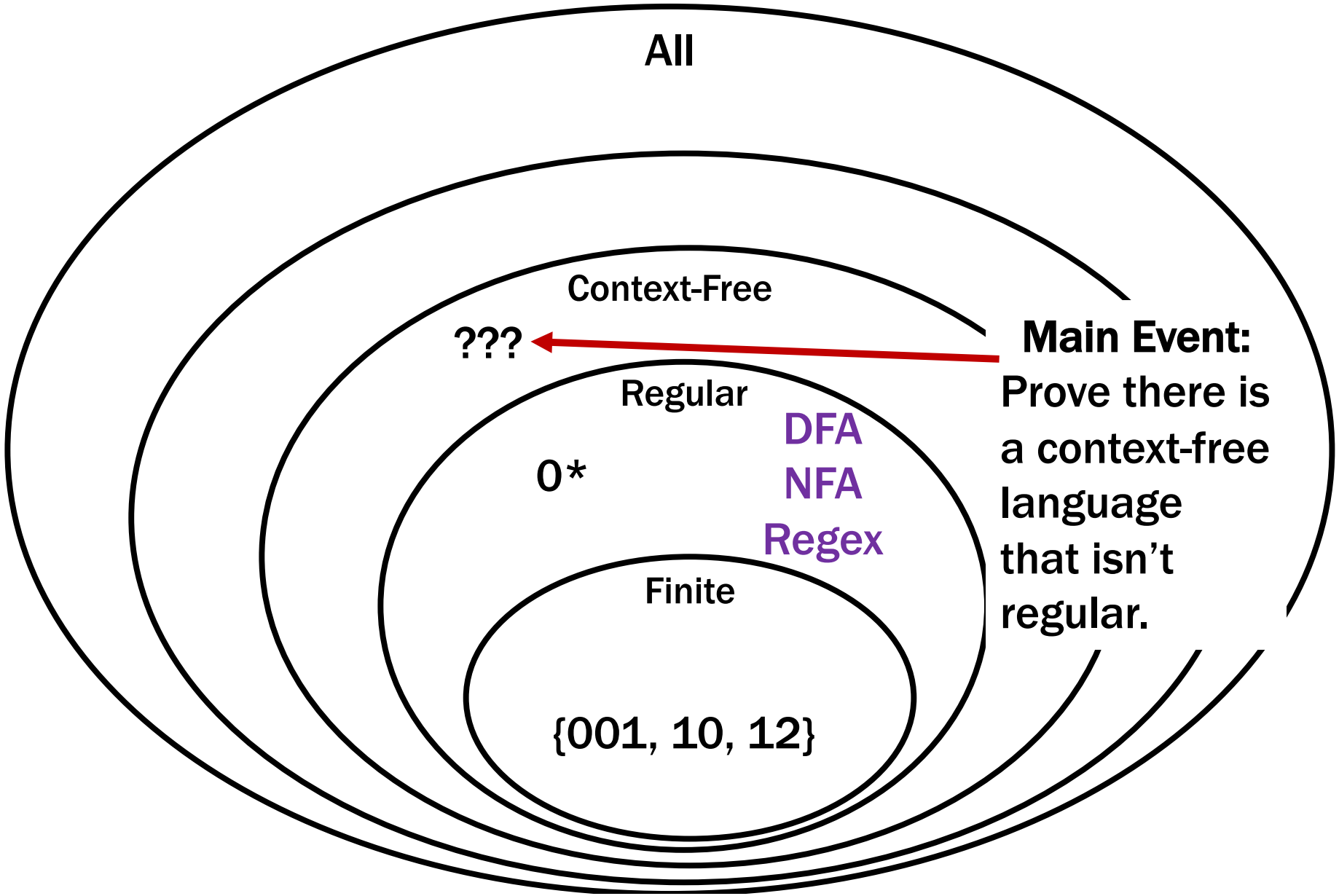
L is infinite.

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!



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 ?

Intuition (NOT A PROOF!):

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

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

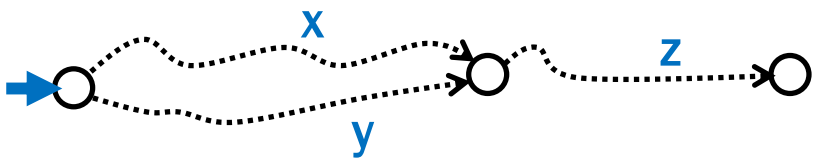
...but there are an infinite # of possible first parts and we only have finitely many states.

Proof idea: any machine that does not remember the entire first half will be wrong for some inputs

Useful Lemmas about DFAs

Lemma 1: If DFA **M** takes $\mathbf{x}, \mathbf{y} \in \Sigma^*$ to the same state, then for every $\mathbf{z} \in \Sigma^*$, M accepts $\mathbf{x} \bullet \mathbf{z}$ iff it accepts $\mathbf{y} \bullet \mathbf{z}$.

M can't remember that the input was $\mathbf{x}$, not $\mathbf{y}$.



$x \bullet z = x_1 x_2 \ldots x_n z_1 z_2 \ldots z_k$

$y \bullet z = y_1 y_2 \ldots y_m z_1 z_2 \ldots z_k$

Useful Lemmas about DFAs

Lemma 2: If DFA **M** has **n** states and a set **S** contains *more* than **n** strings, then **M** takes at least two strings from **S** to the same state.

M can't take n+1 or more strings to different states because it doesn't have n+1 different states.

So, some pair of strings must go to the same state.

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

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

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

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

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

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

We will 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, by Lemma 2, 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 must take the ones we're given!

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

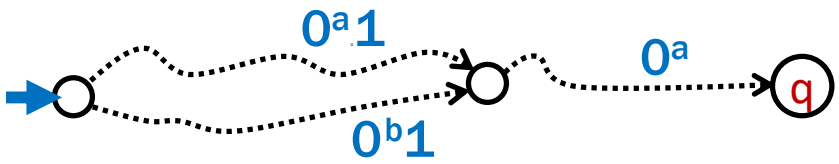
Suppose for contradiction that some DFA, M, accepts B.

We will 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, by Lemma 2, 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.



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

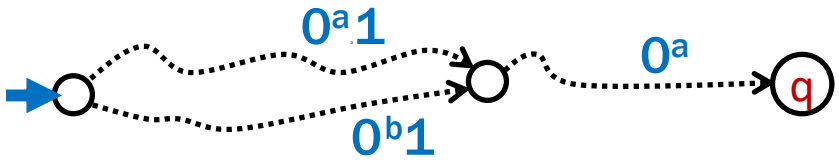
Suppose for contradiction that some DFA, M, accepts B.

We will 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, by Lemma 2, 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.



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 makes a mistake: q needs to be an accept state since $0^a10^a \in B$, but M would accept $0^b10^a \notin B$, which is an error.

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

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

We will 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, by Lemma 2, 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.

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 makes a mistake: q needs to be an accept state since $0^a10^a \in B$, but M would accept $0^b10^a \notin B$, which is an error.

This proves that M does not recognize B, contradicting our assumption that it does. Thus, no DFA 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 prefixes (which we intend to complete later).
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_at and s_bt end at the same state q of M. Since $s_at \in L$ and $s_bt \notin L$, M does not recognize L."
6. "Thus, no DFA recognizes L."

Showing that a Language L is not regular

The choice of S is the creative part of the proof

You must find an infinite set S with the property that *no two* strings can be taken to the same state

- i.e., for ***every pair*** of strings S there is an "accept" completion that the two strings DO NOT SHARE

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

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

Let **S** =

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**.

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.

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.

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.

Thus, 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.

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.

Thus, 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 prefixes (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_at and s_bt end at the same state q of M. Since $s_at \in L$ and $s_bt \notin L$, M does not recognize L."
6. "Thus, no DFA recognizes L."

Fact: This method is optimal

- Suppose that for a language L, the set S is a *largest* set of prefixes 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.

Fact: This method is optimal

- Suppose that for a language L, the set S is a *largest* set of prefixes 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.

Corollary: Our minimization algorithm was correct.

– we separated *exactly* those states for which some t would make one accept and another not accept

Important Notes

- It is not necessary for our strings **xz** with **x** $\in$ **L** to allow any string in the language
  - we only need to find a small "core" set of strings that must be distinguished by the machine

- It is **not true** that, if **L** is irregular and **L** $\subseteq$ **U**, then **U** is irregular!
  - we always have **L** $\subseteq$ **Σ*** and **Σ*** is regular!
  - our argument needs different answers: $xz \in L \nleftrightarrow yz \in L$

    for **Σ***, both strings are always in the language

Do not claim in your proof that, because L ⊆ U, U is also irregular