Adam Blank Spring 2016

CSE 311

Foundations of Computing I

* All slides are a combined effort between previous instructors of the course

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

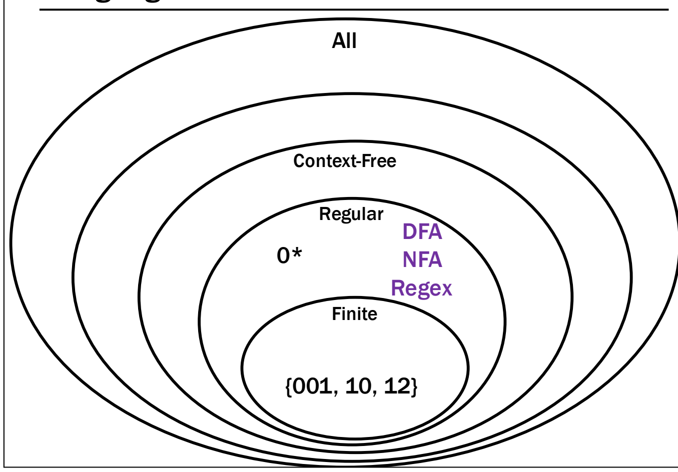
Theorem: A language is recognized by a DFA if and only if it has a regular expression

The second direction will be completely untested. I'm happy to discuss it with you at office hours, but we have more important things to discuss today.

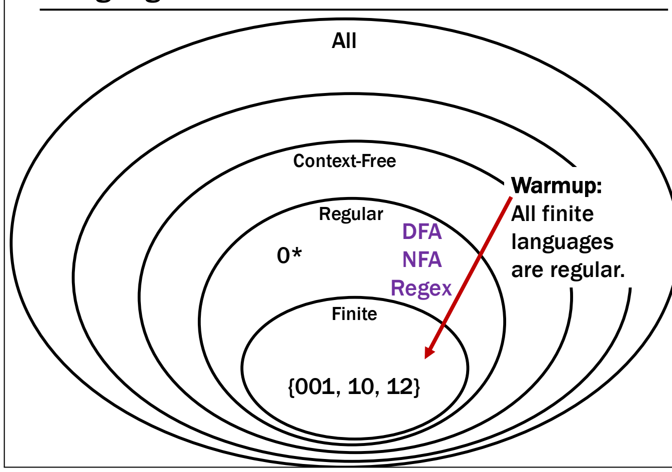
CSE 311: Foundations of Computing

Lecture 25: Limits of FSMs

Languages and Machines!



Languages and Machines!

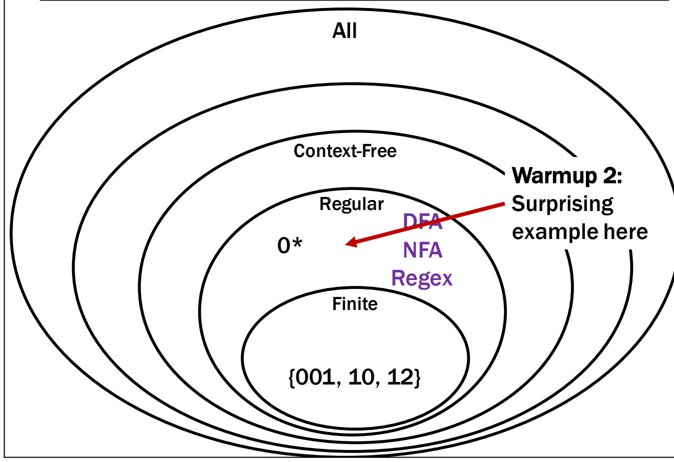


DFAs Recognize Any Finite Language

Construct DFAs for each string in the language.

Then, put them together using the union construction.

Languages and Machines!



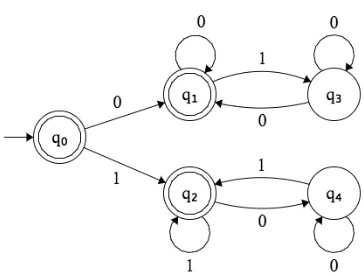
An Interesting Infinite Regular Language

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

L is infinite.

0, 00, 000, …

L is regular.



The language of "Binary Palindromes" is Context-Free

$S \to \varepsilon$ | 0 | 1 | 0S0 | 1S1

We good?

The language of "Binary Palindromes" is Regular

Is it though?

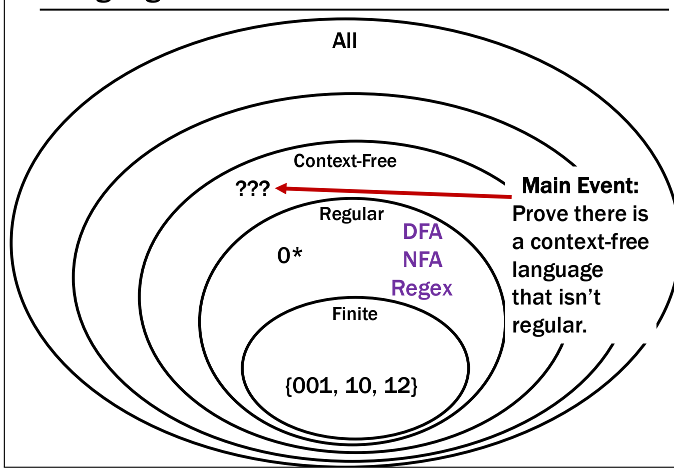
Intuition (NOT A PROOF!):

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.

Languages and Machines!



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 accepts B
- Our goal is to "confuse" M. That is, we want to show it "does the wrong thing".

How can a DFA be "wrong" or "broken"?

Just like the errors you were getting on the homework, a DFA is "broken" 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:

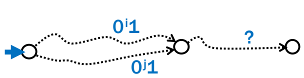
- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that accepts B
- Our goal is to "confuse" M. That is, we want to show it ~~"does the wrong thing"~~ accepts or rejects a string it shouldn't.

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 accepts B
- We want to show M accepts or rejects a string it shouldn't.

Key Idea 1: If two strings "collide" at any point, an FSM 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 accepts B
- We want to show M accepts or rejects a string it shouldn't.

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

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

The general proof strategy is:

- Assume (for contradiction) that it's possible.
- Therefore, some DFA (call it M) exists that accepts B
- We want to show M accepts or rejects a string it shouldn't.
- We choose an **INFINITE** set of "half strings" (which we intend to complete later). It is imperative that every string in our set have a **DIFFERENT, SINGLE** "accept" completion.

0____	1____	1____	0____
00____	10____	01____	01____
000____	100____	001____	101____
0000____	1000____	0001____	0101____
00000____	10000____	00001____	10101____

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={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, accepts B.

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

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

Since there are finitely many states and infinitely many strings in S, there exists strings $0^a1 \in S$ and $0^b1 \in S$ that end in the same state.

SUPER IMPORTANT POINT: You do not get to choose what a and be are. Remember, we've 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.

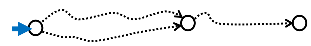
We show M accepts or rejects a string it shouldn't.

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

Since there are finitely many states and infinitely many strings in S, there exists strings $0^a1 \in S$ and $0^b1 \in S$ that end in the same state.

Now, consider appending 0^a to both strings.

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



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

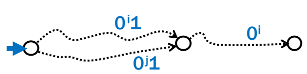
Suppose for contradiction that some DFA, M, accepts B.

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

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

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

Now, consider appending 0^a to both strings. *Then, since 0^a1 and 0^b1 are in the same state, 0^a10^a and 0^b10^a also end in the same state. Since $0^a10^a \in B$, this state must be an accept state. But, then M accepts $0^b10^a \notin B$.*



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

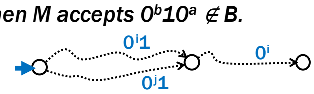
Suppose for contradiction that some DFA, M, accepts B.

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

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

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

Now, consider appending 0^a to both strings. *Then, since 0^a1 and 0^b1 are in the same state, 0^a10^a and 0^b10^a also end in the same state. Since $0^a10^a \in B$, this state must be an accept state. But, then M accepts $0^b10^a \notin B$.*



This is a contradiction, because we assumed M accepts B. Since M was arbitrary, **there is no DFA that accepts B.**

Showing a Language L is not regular

1. "Suppose for contradiction that some DFA M accepts L."
2. Consider an **INFINITE** set of "half strings" (which we intend to complete later). It is imperative that every string in our set have a **DIFFERENT, SINGLE** "accept" completion.
3. "Since **S** is infinite and **M** has finitely many states, 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**."
4. Consider appending the (correct) completion to one of the two strings.
5. "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**."
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, accepts A.

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 ≠ j) that end in the same state in M.

Consider appending 1^i to both strings. Note that $0^i1^i \in$ **A, but** $0^j1^i \notin$ **A** since i ≠ 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.**

Suppose for contradiction that some DFA, **M**, accepts **L**.

Let **S** = {ε, 012, 012012, 012012012, ...} = {$(012)^n$: n ∈ ℕ}

Since **S** is infinite and **M** is finite, there must be two strings $(012)^i$ and $(012)^j$ for some i ≠ j that end up at the same state of **M**. 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.

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**.

Since **M** was arbitrary, no DFA recognizes **L**.