CSE 311: Foundations of Computing

Lecture 27: Irregularity

Recap from last lecture

REs $\subseteq$ **CFGs**

$\equiv$

DFAs $\equiv$ **NFAs**

Transform n-state NFA to 2^n-state DFA:

- DFA simulates the set of reachable NFA states

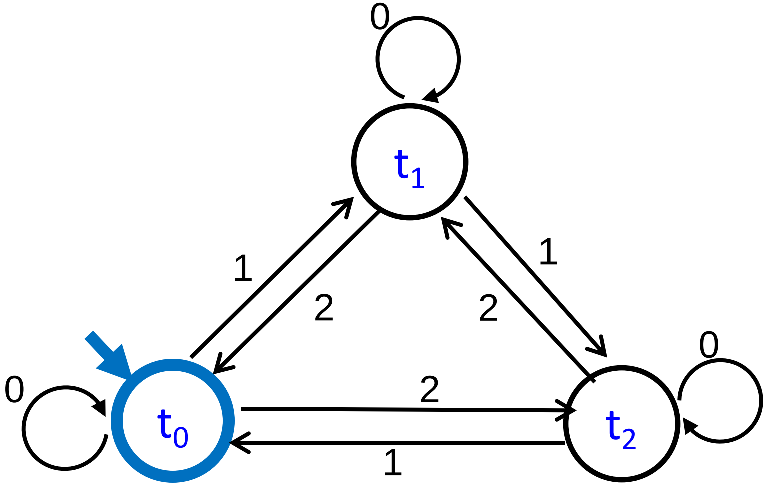
Transform NFA to RE:

- Allow generalized NFA where edges labelled with REs
- Reduce Generalized NFA one state after the other

Converting an NFA to a regular expression

Consider the DFA for the mod 3 sum

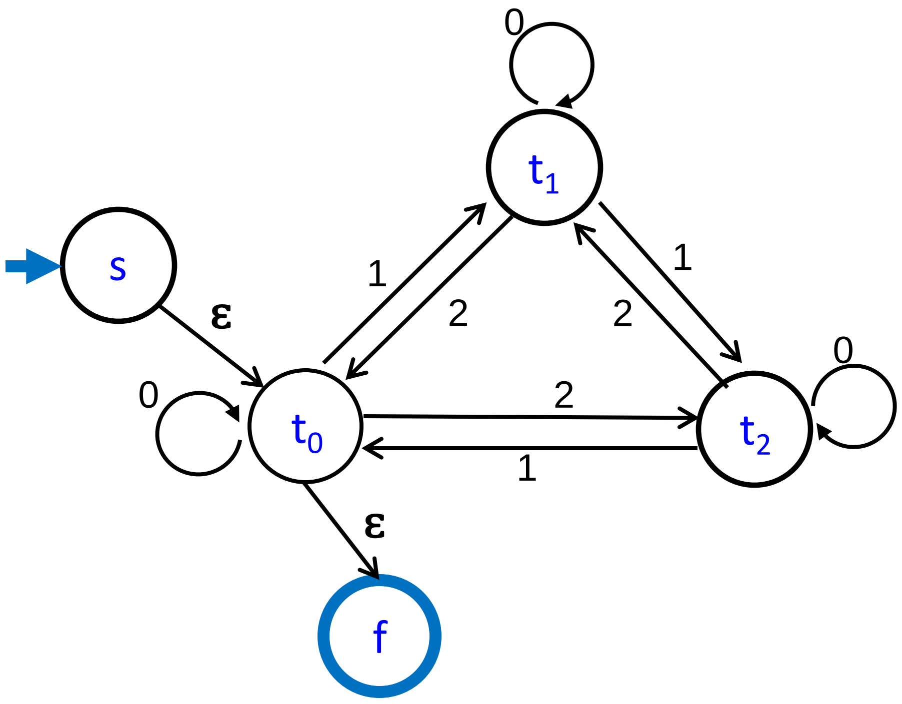
- Accept strings from {0,1,2}* where the digits mod 3 sum of the digits is 0



Splicing out a state t_1

Regular expressions to add to edges

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



Splicing out a state t_1

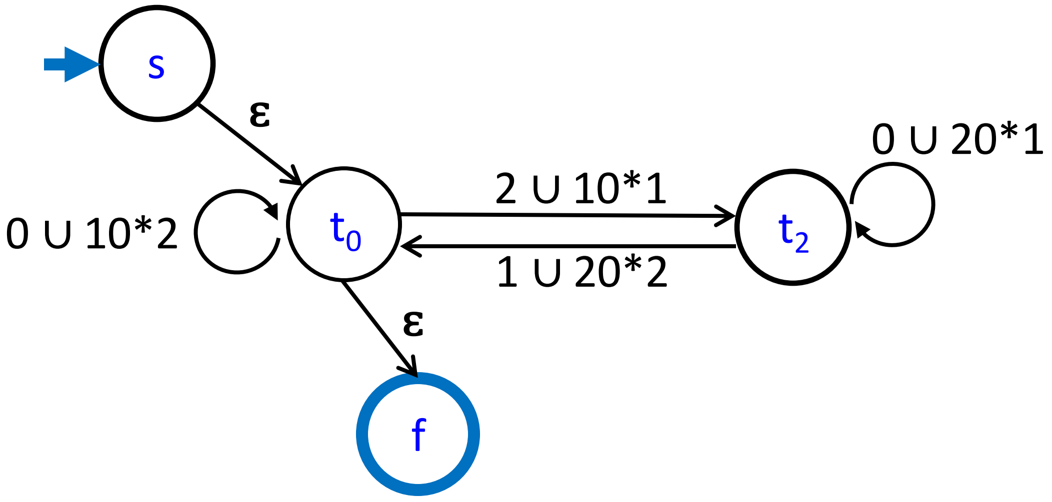
Regular expressions to add to edges

$t_0 \to t_1 \to t_0$: 10*2

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

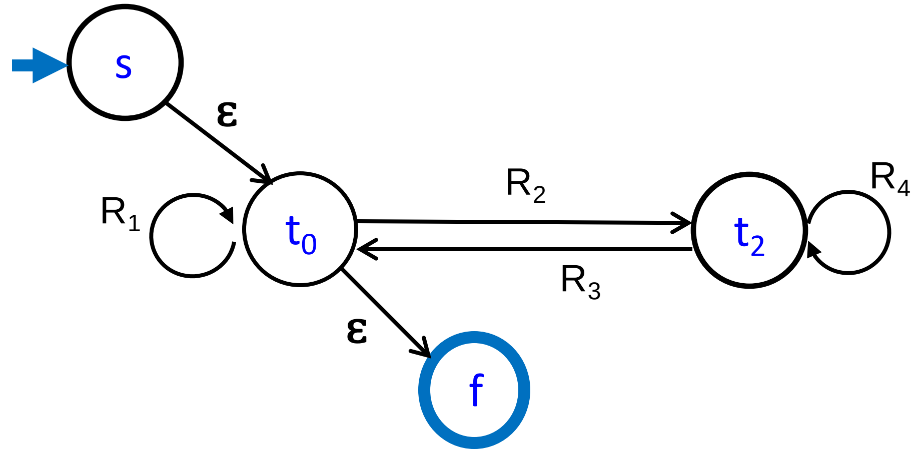
$t_2 \to t_1 \to t_0$: 20*2

$t_2 \to t_1 \to t_2$: 20*1

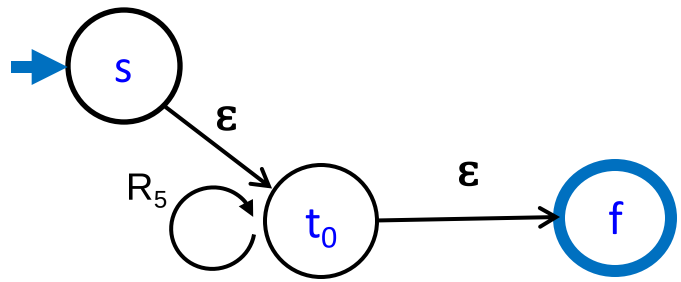


Splicing out state t_2 (and then t_0)

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



R_5: R_1 ∪ $R_2R_4{}^*R_3$



Final regular expression: R_5*=
(0 ∪ 10*2 ∪ (2 ∪ 10*1)(0 ∪ 20*1)*(1 ∪ 20*2))*

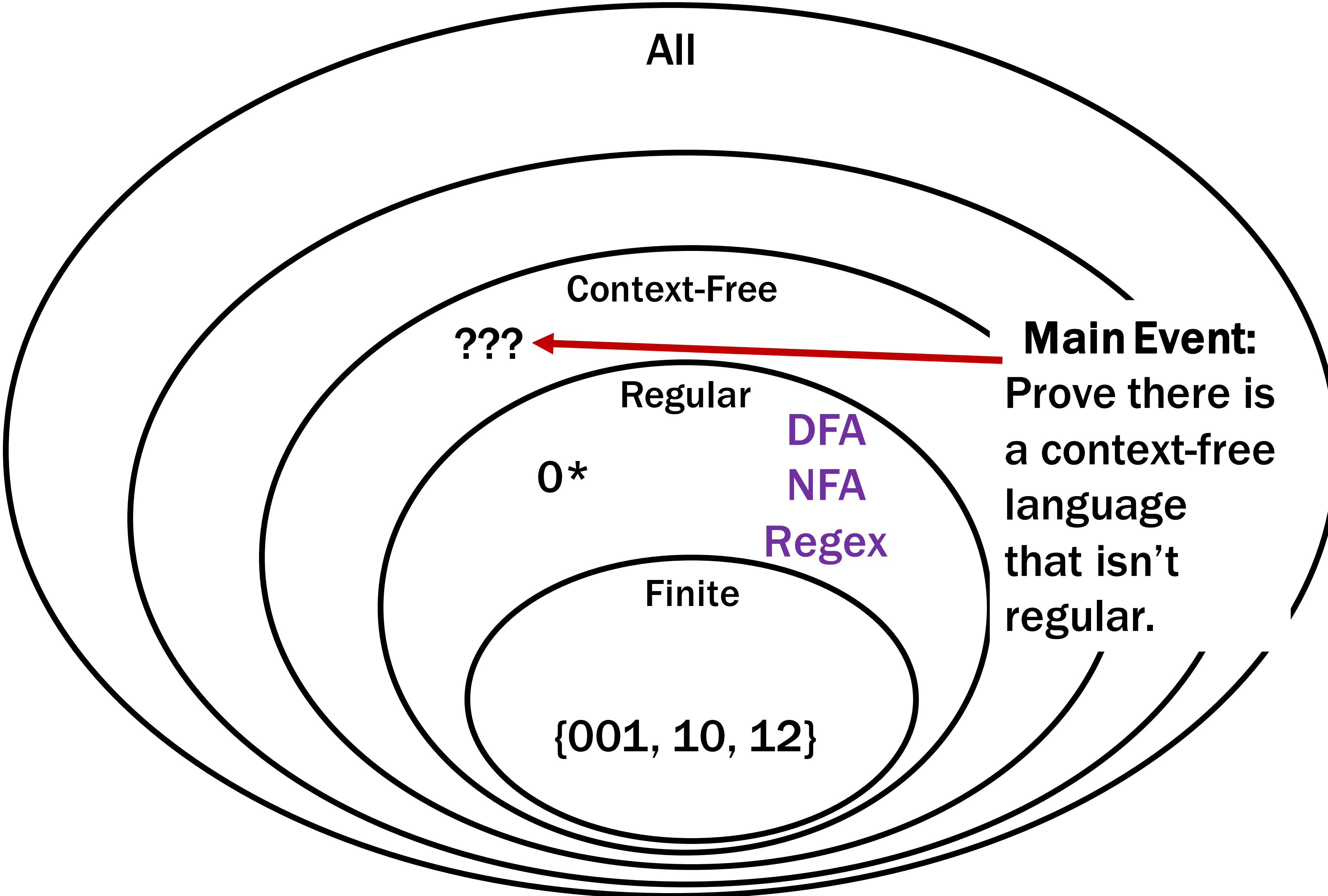
The story so far...

REs $\subseteq$ CFGs

$\equiv$

DFAs $\equiv$ NFAs

Languages and Representations!



The language of “Binary Palindromes” is Context-Free

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

Is it regular?

Is the language of “Binary Palindromes” Regular ?

Intuition (NOT A PROOF!):

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

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.

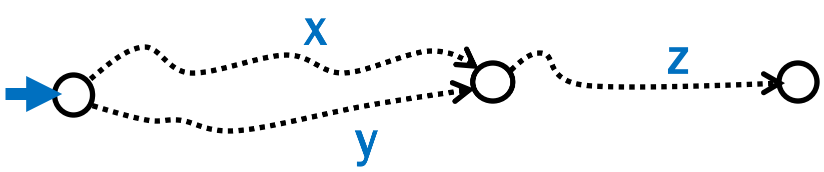
Proof idea: any machine that does not remember the entire first half will be wrong for some inputs

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!

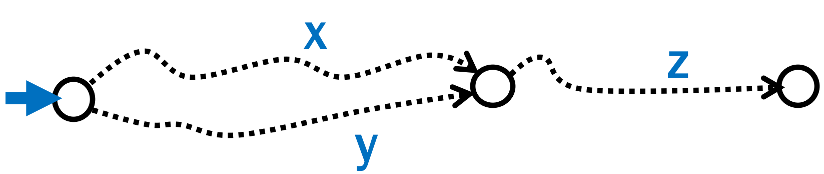


B = {binary palindromes} can't be recognized by any DFA

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

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

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

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

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

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

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

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

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.***

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

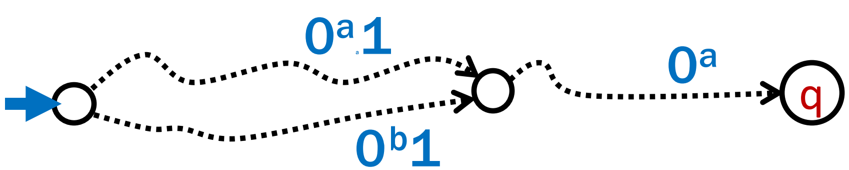
Proof. Suppose for contradiction that some DFA, M, recognizes B.

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.

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 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

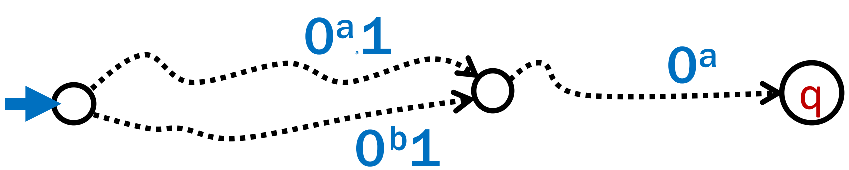
Proof. Suppose for contradiction that some DFA, M, recognizes B.

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.

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, 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 "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 t (depends on s_a and s_b) 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."

Lecture 27 Activity

You will be assigned to breakout rooms. Please:

- Introduce yourself
- Choose someone to share their screen, showing this PDF
- Fill in the gaps of the proof that the language $A = \{0^n 1^n : n \geq 0\}$ is not regular.

1. Suppose for contradiction that some DFA, M, recognizes A.
2. Let S = {???}. Since S is infinite and M has finitely many states, there must be two *distinct* strings, ??? and ??? that end in the same state in M.
3. Consider appending t=??? to both strings.
4. Note that ???t ∈ A, **but** ???t ∉ A since ????. But they both end up in the same state of M, call it **q**. Since ???t ∈ A, state **q** must be an accept state but then M would incorrectly accept ???t ∉ A so M does not recognize A.
5. Since M was arbitrary, no DFA recognizes A.

Fill out the poll everywhere for Activity Credit!
Go to pollev.com/philipmg and login with your UW identity

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 ≥ 0}. Since S is infinite and M has finitely many states, there must be two strings, $(^a$ and $(^b$ for some a ≠ 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 ≠ 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 <u>"accept" completion</u> 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.

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.

Corollary: Our minimization algorithm was correct.

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

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.