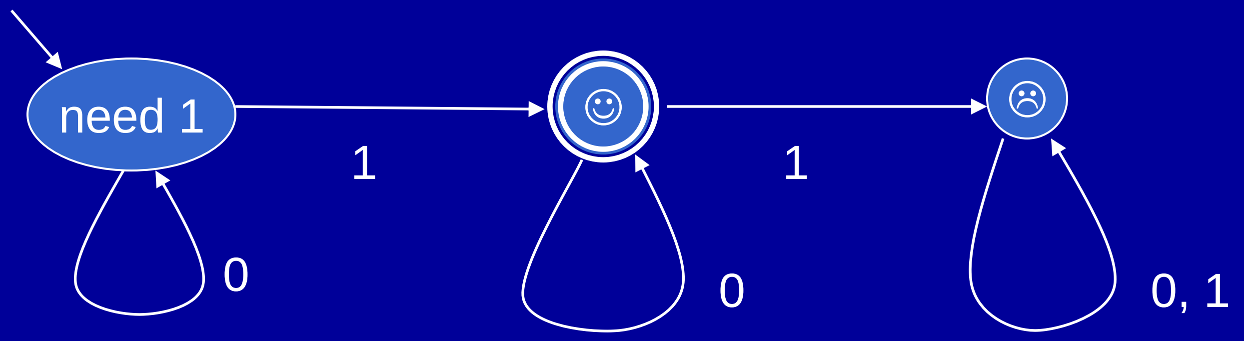
CS 461 – Sept. 12

- Simplifying FA's.
  - Let's explore the meaning of "state". After all if we need to simplify, this means we have too many states.
  - Myhill-Nerode theorem
  - Handout on simplifying

What is a state?

Example: 0*10*



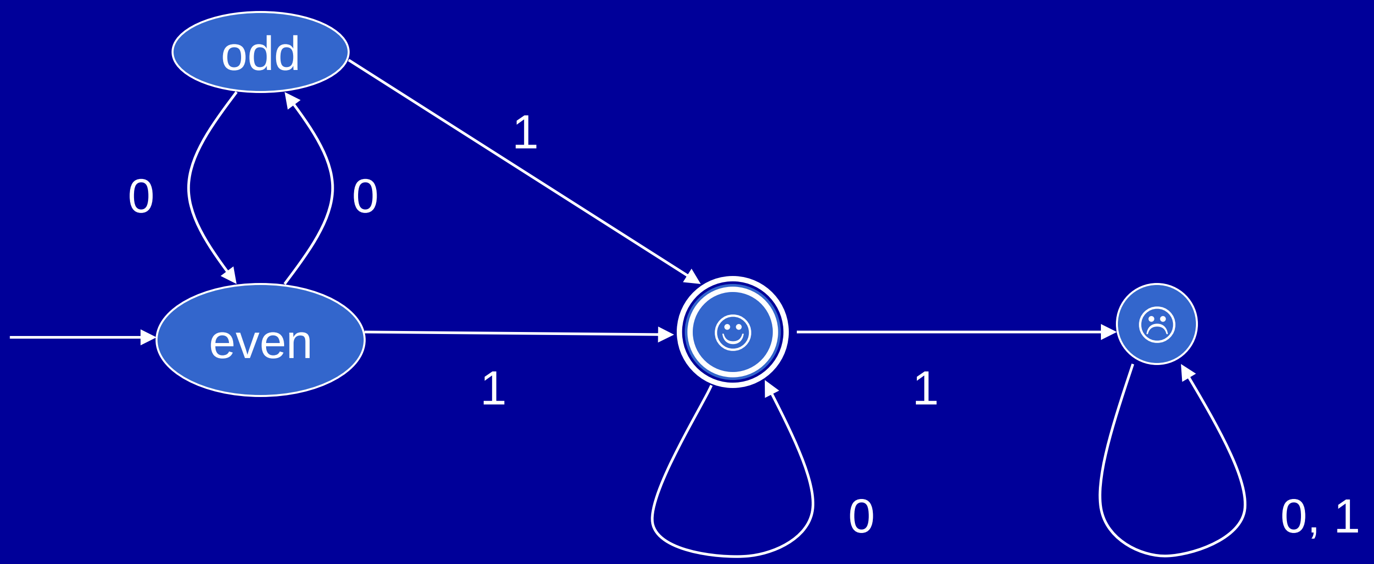
state = 0* state = 0*10* state = 0*10*1 (0 + 1)*

- No matter what next input symbol is, all words in same state act alike.

  x, y same state → ∀z, xz same state as yz

- A state is a collection of words that react to input in the same way.

Equivalent states



- Whether we've seen even or odd number of 0's shouldn't matter. Only concern is ultimate outcome: will string be accepted or not?
- Words x and y <u>should be in same state</u> if ∀z, xz and yz have the same outcome.

In other words

- The 2 machines are equivalent.

	From state	Input 0	Input 1
→	Need 1	Need 1	Good
☺	Good	Good	Bad
	Bad	Bad	Bad

	From state	Input 0	Input 1
→	Even	Odd	Good
	Odd	Even	Good
☺	Good	Good	Bad
	Bad	Bad	Bad

Myhill-Nerode theorem

- Basis for simplification algorithm.
- Also gives condition for a set to be regular.
  - i.e. infinite # states not allowed.

<u>3 parts to theorem</u>:

1. For any language L, we have equivalence relation R: xRy if $\forall$z, xz and yz same outcome.
2. If L is regular, # of equivalences classes is finite.
3. If # equivalences classes finite, language is regular.

Proof (1)

For any language L, we have equivalence relation R: xRy if $\forall$z, xz and yz same outcome.

- To show a relation is an equivalence relation, must show it is reflexive, symmetric and transitive.
- Reflexive: xz and xz have same outcome. (i.e. both are accepted, or both are rejected.) ☺
- Symmetric. If xz has same outcome as yz, then yz has same outcome as xz.
- Transitive. If xz has same outcome as yz, and yz has same outcome as tz, then xz has same outcome as tz.

All 3 are obviously correct.

Proof (2)

If L is regular, # of equivalences classes is finite.

- Regular means L is recognized by some FA.
- Thus, # of states is finite.
- It turns out that (# equiv classes) <= (# states)

  Why? Because 2 states may be “equivalent.”

  More importantly: # classes can’t exceed # states. Proof:

  Assume # classes > # states. Then we have a state representing 2 classes. In other words, x and y in the same state but x <u>not</u> related to y. Meaning that ∃z where xz, yz don’t have same fate but travel thru the same states! This makes no sense, so we have a contradiction.

- Since (# equiv classes) <= (# states) and # states is finite, we have that # equiv classes is finite. ☺

Proof (3)

If # equivalences classes finite, language is regular.

- We prove regularity by describing how to construct an FA.

- Class containing ε would be our start state.
- For each class: consider 2 words x and y.
  - x0 and y0 have to be in the same class.
  - x1 and y1 have to be in the same class.
  - From this observation, we can draw appropriate transitions.
  - Since states are being derived from classes, the number of states is also <u>finite</u>.
- Accept states are classes containing words in L.

Example

- L = { all bit strings with exactly two 1's } has 4 equivalence classes

  [ε], [1], [11], [111]

- Let's consider the class [11]. This includes words such as 11 and 0101.
  - If we append a 0, we get 110 and 01010. These words also belong in [11].
  - If we append a 1, we get 111 and 01011. These words belong in the class [111].
- This is the same thought process we use when creating FA's transitions anyway.

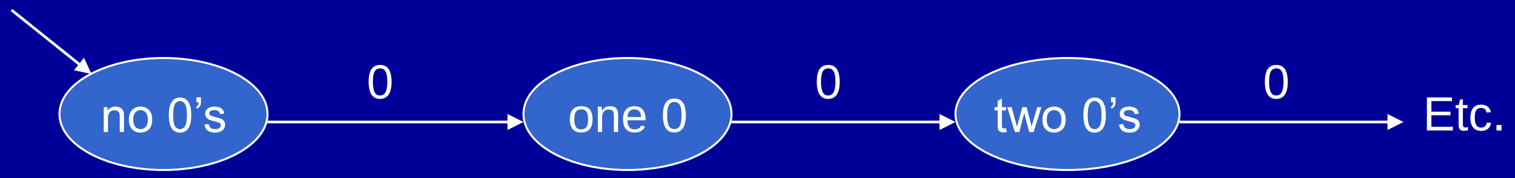
Learning from theorem

- There exist non-regular languages! It happens if # of equivalence classes is infinite.
  - Soon we'll discuss another method for determining non-regularity that's a little easier.

- Part 3 of proof tells us that there is an FA with the minimum number of states (states = equivalence classes).
  - See simplification algorithm handout.

Example

Is $\{ 0^n1^n \}$ regular? This is the language ε, 01, 0011, 000111, etc.



$c_1 = \{ \varepsilon \}$ $c_2 = \{ 0 \}$ $c_3 = \{ 00 \}$ $c_4 = \{ 000 \}$

- Should x=0 and y=00 be in the same class? No!
  Consider z = 1. Then xz = 01 and yz = 001. Different outcome!
- ∞ # classes → Can't draw FA.
- Equivalence classes are usually hard to conceive of, so we'll rely more on a different way to show languages not regular.