Today's learning goals

- NFA definition, computations, etc.
- Design NFA to recognize a given language
- Compare properties of regular and NFA-recognizable languages
- Relation between DFAs and NFAs
Differences between NFA and DFA

• **DFA**: unique computation path for each input

• **NFA**: allow several (or zero) alternative computations on *same input*
  
  • $\delta(q, x)$ may specify *more than one* possible next states
  
  • $\epsilon$ transitions allow the machine to *transition between states spontaneously*, without consuming any input symbols
  
  • computation can *get stuck* at some state, if there's a missing arrow
A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where

1. \(Q\) is a finite set called the states
2. \(\Sigma\) is a finite set called the alphabet
3. \(\delta : Q \times \Sigma \rightarrow P(Q)\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F \subseteq Q\) is the set of accept states.

Which piece of the definition of NFA means there might be **more than one** possible next state from a given state, when reading symbol \(x\) from the alphabet?

A. Line 2, the size of \(\Sigma\)
B. Line 3, the domain of \(\delta\)
C. Line 3, the codomain of \(\delta\)
D. Line 5, that \(F\) is a set
E. I don't know.
Acceptance in an NFA

An NFA \((Q, \Sigma, \delta, q_0, F)\) accepts a string \(w\) in \(\Sigma^*\) iff we can write \(w = y_1 y_2 \cdots y_m\) where each \(y_i \in \Sigma_e\) and there is a sequence of states \(r_0, \ldots, r_m \in Q\) such that

1. \(r_0 = q_0\)

2. \(r_{i+1} \in \delta(r_i, y_{i+1})\) for each \(i = 0, \ldots, m - 1\)

3. \(r_m \in F\).
Tracing NFA execution

What's the length of a shortest string accepted by this NFA?

A. 0
B. 1
C. 2
D. 3
E. I don't know
Tracing NFA execution

The language recognized by this NFA is …
Designing NFA

Design an NFA which recognizes the language

\{ \text{w0 in } \{0,1\}^* \mid |w| \text{ is a multiple of 3}\}
Designing NFA

The language recognized by this NFA is …

The transition function of this NFA can be formally written as …
structure i was using to display the output was a TreeCtrl, so I figured some kind of weird tree traversal system would work and i made it into **this implicit state machine** that used a stack to traverse up and down into different levels. In short, it was an absolute monster of a structure (which i am immensely proud of for continuing to work as new requirements were added every couple of hours :p). Needless to say, all that code is now gone. Tuesday I came in on a mission to reimplement the tree without any interruption in the continual updating of my project. This was made possible by mentor deciding to just disappear into thin air for the day. I designed am **NFA(thanks based Shacham)** that could cover every case and also allow the timeline to be completely extensible. Note, I needed an NFA because there are a few input strings that require up to 5 transitions (the tree gets really deep) and the states couldn't really be just jumped to because of the way the TreeCtrl works. After lunch, I implemented this NFA (as it turns out, there was a slight amount of copy and paste) and had it working and fully functional within an hour. On Wednesday, I arranged a short meeting with my mentor to show it off and discuss some requirements for a side project I was to begin working on. When my mentor showed up to the meeting, he brought with him a Qualstar for me! Qualstars are internal awards given to employees who exceed expectations and provide excellent work (or something like that) and I got one for completing my first project so quickly and saving engineer time. Needless to say, my mentor was not impressed with my NFA and gave me some more work to do…

I FINALLY USED SOMETHING I LEARNED AT UCSD WHICH I NEVER THOUGHT I WOULD USE!!!!! i used CSE 105 to design my NFA and knew that an NFA would be a good way to solve this problem only because of that class. Also, that Qualstar i got kinda feels like a real world A+, except it actually means something :p

Chris Miranda (CSE 197)
Similarities between DFA and NFA

- If L is a language recognized by a DFA, is there some NFA that recognizes it?

A. Yes  
B. No  
C. Depends on L  
D. I don't know.
Similarities between DFA and NFA

- If L is a language recognized by an NFA, is there some DFA that recognizes it (aka is it regular)?

  A. Yes
  B. No
  C. Depends on L
  D. I don't know.
Simulating NFA with DFA

Not quite a closure proof, but ...

Proof:

**Given** name variables for sets, machines assumed to exist.

**WTS** state goal and outline plan.

**Construction** using objects previously defined + new tools working towards goal. Give formal definition and explain.

**Correctness** prove that construction works.

**Conclusion** recap what you've proved.
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

Proof:

**Given** A, a language recognized by \( N = (Q, \Sigma, \delta, q_0, F) \) a NFA

**WTS** there is some DFA \( M \) with \( L(M) = A \)

**Construction**

**Correctness**

**Conclusion**
From NFA to DFA

What is the tree of computation paths?
For next time

Try to convert NFA from previous slide to an equivalent DFA on your own.

Homework 2 due tonight

Regrade requests for HW1 must be submitted by noon tomorrow. **Budget for invalid regrade requests**