Review: DFAs and NFAs

Dan, remind us what we’re doing.

If you’re reading this at home, just read the last lectures’ slides, I guess.
Review: Converting $\epsilon$-NFAs to NFAs

Follow these 4 easy steps.

1. take $\epsilon$ shortcuts
2. pull back final states
3. remove $\epsilon$ transitions
4. remove dead states
Defined recursively: a regular expression (RE) is

- $\emptyset$, or
- $\epsilon$, or
- $a$, where $a \in \Sigma$
- $E_1E_2$ where $E_1$ and $E_2$ are REs
- $E_1|E_2$ where $E_1$ and $E_2$ are REs
- $E^*$ where $E$ is a RE
- $(E)$ where $E$ is a RE
RE examples

- \( L = \{ \text{cab, car, card} \} \)

- \( \Sigma = \{a\}, L = \{w: \text{w contains even \#s of a's} \} \)

- \( \Sigma = \{a, b\}, L = \{w: \text{w contains even \#s of a's} \} \)
More RE examples

- $\Sigma = \{a, b\}, L = \{w: \text{contains either aa or bb}\}$

- $\Sigma = \{a, b\}, L = \{w: \text{contains no occurrence of aa or bb}\}$
RE to $\epsilon$-NFA

Convert piece by piece. Recall:
a regular expression (RE) is

- $\emptyset$, or

- $\epsilon$, or

- $a$, where $a \in \Sigma$

(continued on next slide)
RE to $\epsilon$-NFA (continued)

- $E_1E_2$ where $E_1$ and $E_2$ are REs

- $E_1|E_2$ where $E_1$ and $E_2$ are REs

- $E^*$ where $E$ is a REs
Circle of Life

- Arden's Lemma*
- Regex
- Thompson's construction
- \( \varepsilon \)-NFA
- DFA
- NFA
- subset construction
- \( \varepsilon \)-closure
Practical Applications of DFAs

- Most real-world examples do not care about recognizers (DNA match may be the exception)
- Mostly, DFAs are used for:
  - transforming/transducing input
  - searching in text
  - scanning/translating
Transducers

A transducer is a DFA with output. That is, transitions look like input/output.

Example 1: Remove stutters from $\Sigma = \{a, b\}$. 
Final word about transducers

- Mealy Machine
- Moore Machine
- "translating" does not "give meaning"
DFA for MIPS

- It is easy to write a DFA to recognize an individual token type
- It is not difficult to combine these into one DFA that recognizes a word as some token type
The scanning problem:
Input: some string $w$ and a language $L$

Output: $w_1w_2\cdots w_n = w$ where $w_i \in L$ for all $i$

There may be more than one possible answer: 0x1234abcd

We solve this problem by...