Lecture 9+10
Regular Expressions

CS 241: Foundations of Sequential Programs
Winter 2018

Troy Vasiga et al
University of Waterloo
$\epsilon$-NFAs

- allows transition between states on “no input”
- can be used as “glue” for joining machines together
- example: $L = \{ \text{card, cab, calf} \}$
Converting $\epsilon$-NFAs to NFAs

It should not be surprising that $\epsilon$-NFAs can be converted to NFAs

1. take $\epsilon$ shortcuts
2. pull back final states
3. remove $\epsilon$ transitions
4. remove dead states
Regular Expressions

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 REs
RE examples

- $L = \{\text{cab, car, card}\}$

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

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

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

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

- $E_1 E_2$ 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

Definition of Regular Language:

A picture
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”
▶ 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
▶ See the DFA for MIPS on the CS241 website
The scanning problem:

**Input:** some string $w$ and a language $L$

**Output:** $w_1 w_2 \cdots w_n = w$ where $w_i \in L$ for all $i$

There may be more than one possible answer: 0x1234abcd

We solve this problem by...
Simplified Maximal Munch

Input is a word $c_1 c_2 \cdots c_k$

\[ i = 0 \quad // \quad i \text{ is the index of the current character} \]

\[ \text{state} = \text{START} \]

loop

\[ \text{newstate} = \text{ERROR} \]

if ( \( i < k \) ):

\[ \text{newstate} = \delta(\text{state}, c_i) \]

if newstate == ERROR:

if state is not a final state:

report an error and exit

if state is not WHITESPACE:

output appropriate token

\[ \text{state} = \text{START} \]

if $i == k$:

exit

else:

\[ \text{state} = \text{newstate} \]

\[ i = i + 1 \]
Next Steps