Lecture 9+10

Regular Expressions

CS 241: Foundations of Sequential Programs
Winter 2018

Troy Vasiga et al
University of Waterloo

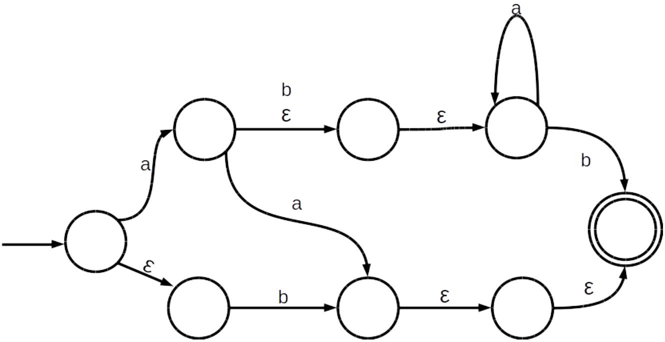
ϵ-NFAs

- allows transition between states on "no input"
- can be used as "glue" for joining machines together
- example: $L = \{$ card, cab, calf $\}$

Converting ϵ-NFAs to NFAs

It should not be surprising that ϵ-NFAs can be converted to NFAs

1. take ϵ shortcuts
2. pull back final states
3. remove ϵ transitions
4. remove dead states



Regular Expressions

Defined recursively: a regular expression (RE) is

- $\emptyset$, or
- ϵ, or
- a, where $a \in \Sigma$
- $E_1 E_2$ 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

RE examples

- L = {cab, car, card}

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

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

More RE examples

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

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

RE to ϵ-NFA

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

- $\emptyset$, or
- ϵ, or
- a, where $a \in \Sigma$

(continued on next slide)

RE to ϵ-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"

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
- See the DFA for MIPS on the CS241 website

Scanning

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 // i is the index of the current character
state = START
loop
   newstate = ERROR
   if ( i < k ):
      newstate = δ(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
      state = START
      if i == k:
         exit
   else:
      state = newstate
      i = i + 1
```

Next Steps