Lecture 11+12

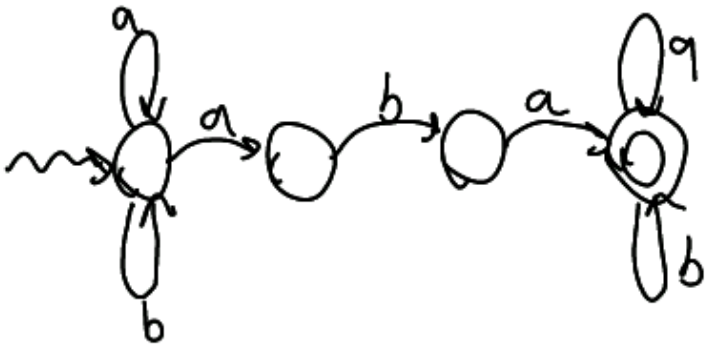
Regular Expressions

CS 241: Foundations of Sequential Programs
Fall 2014

Troy Vasiga et al
University of Waterloo

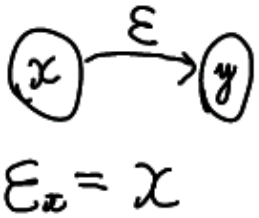
Review

- An example comparing DFAs and NFAs

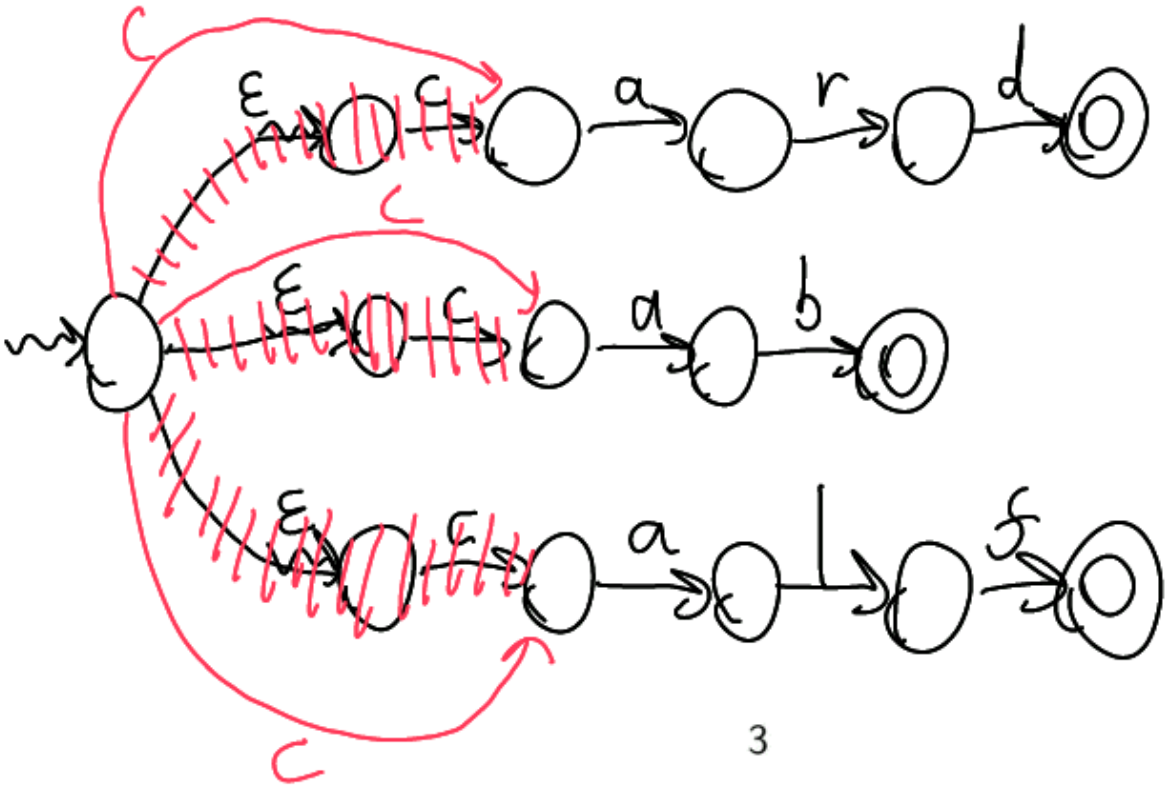


- Can convert between an NFA and DFA using the subset construction
  - define each set of states that can be occupied at the same step
  - one state in the DFA for each unique set of states in the NFA

ϵ-NFAs



- allows transition between states on "no input"
- can be used as "glue" for joining machines together
- example: $L = \{$ card, cab, calf $\}$
- it is not surprising that ϵ-NFAs can be converted to NFAs



Regular Expressions

↳ specify a regular language

Defined recursively: a regular expression (RE) is

- $\emptyset$, or $L = \{\}$
- ϵ, or $L = \{\epsilon\}$ — finite languages ← base cases
- a, where $a \in \Sigma$ ← $L = \{a\}$
- $E_1 E_2$ where E_1 and E_2 are REs ← concatenations
- $E_1 | E_2$ where E_1 and E_2 are REs ← union / alternating "or"
- E^* where E is a REs ← repetition

zero or more copies

recursive cases

RE examples

- L = {cab, car, card}

cab | car | card

or: ca(b|r|rd)
ca(b|r(d|ε))

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

$(aa)^*$ ε, aa, aaaa, ...

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

$b^*(ab^*ab^*)^*$

More RE examples

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

$aa|bb$

$(a|b)^*(aa|bb)(a|b)^*$

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

Ex

(Unix) Shorthands and Extensions

- $[a-z] \leftarrow (a|b|c|\ldots|z)$
- E+ $\leftarrow EE^*$
  One or more copies
- others may occur in your future readings

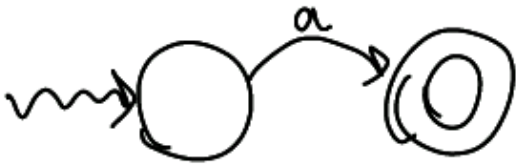
RE to ϵ-NFA

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

- $\emptyset$, or

- ϵ, or

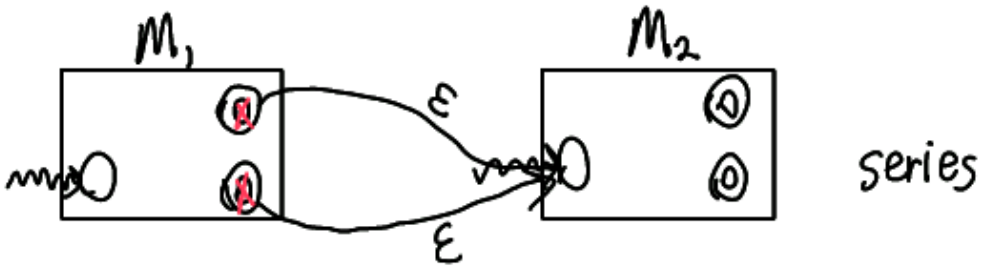
- a, where $a \in \Sigma$



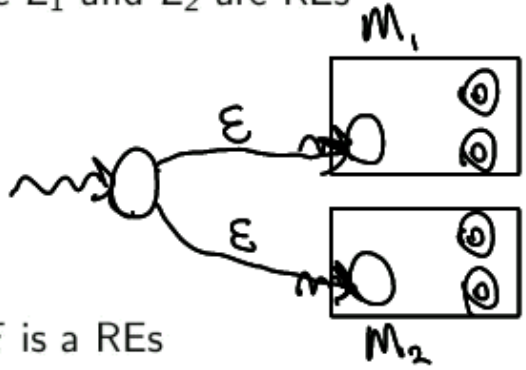
(continued on next slide)

RE to ϵ-NFA (continued)

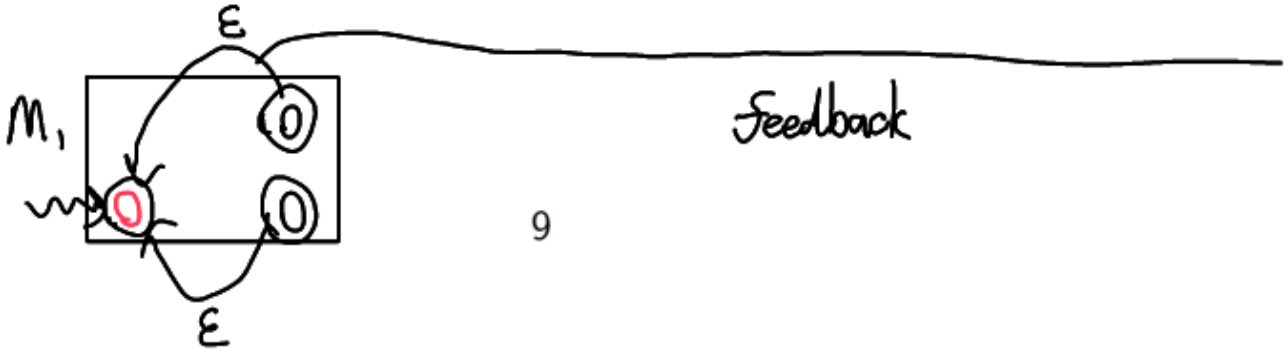
- E_1E_2 where E_1 and E_2 are REs



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



- E^* where E is a REs

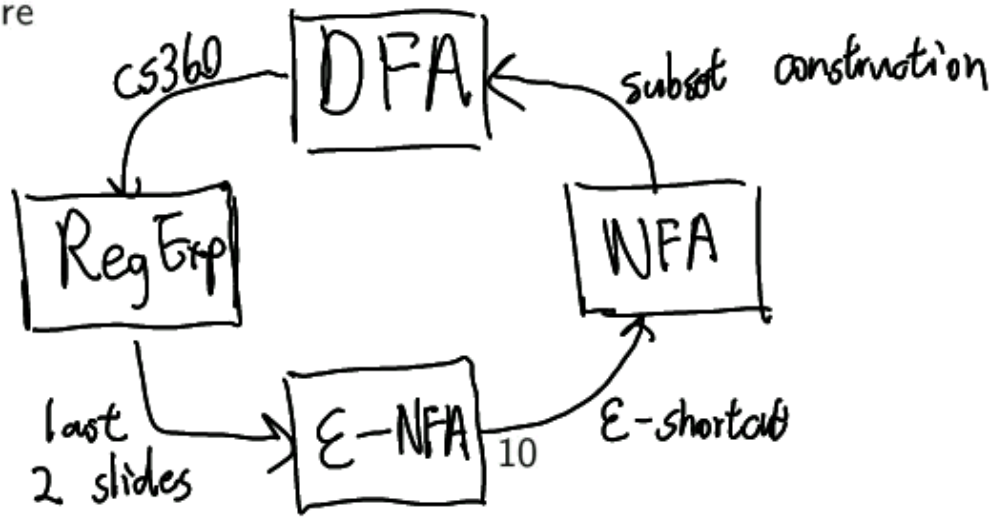


Circle of Life

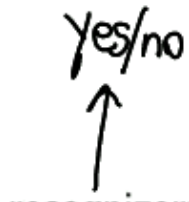
Definition of Regular Language:

specified by a regular expression
recognized by a DFA
NFA
ε-NFA

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

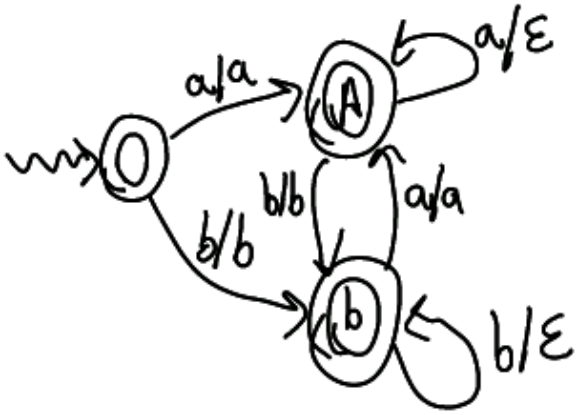
Transducers

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

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

aaabbaaaa ⇒ aba

baabbb ⇒ bab



Transducers

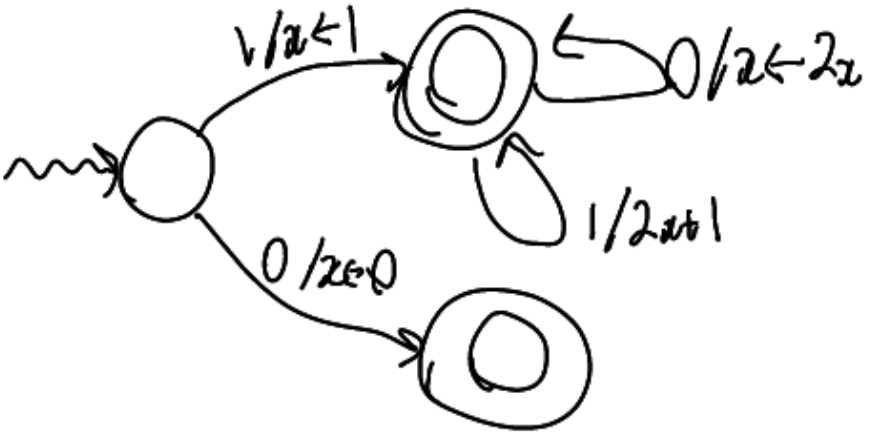
Example 2: Binary integers from $\Sigma = \{0, 1\}$.

$0 \to 0$

$1010 \to 10$

$10000 \to 16$

Calculate decimal value into the variable x.



Final word about transducers

- Mealy Machine
- Moore Machine
- "translating" does not "give meaning"

Searching

Goal: Find the first occurrence of a string $p \in L$ inside a (larger) string T.
Solution (naive):

- aaba ← p
- T ← aacabbaababaabaa
-

Problems:

Knuth
Morris
Pratt

SE240

This turns out to be difficult, but does rely on DFAs (see CS240).

DFA for MIPS

lexer

actual DFA

- 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

"add $1, $2, $3"

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

<ADD> <REG> <COMMA> <REG> ...

There may be more than one possible answer: `0x1234abcd`

↳ HEXINT

↳ could be:

<INT><LABEL><INT><LABEL>
0 x 1234 abcd

We solve this problem by...

repeatedly runs the dfa

Simplified Maximal Munch

Input: $c_0 c_1 c_2 \ldots c_{k-1}$ (↑)

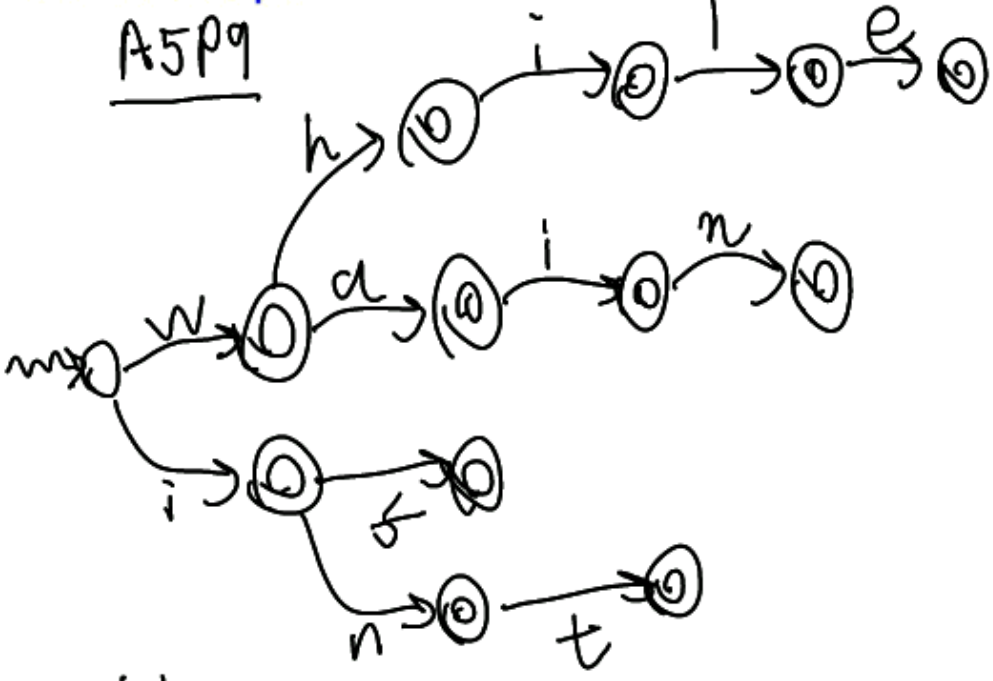
```
i = 0
state = START
loop
   newstate = ERROR ← assume worst case
   if ( i < k ):
      newstate = δ(state, c_i) ← If not at the end of the string, look up next state
   if newstate == ERROR:
      if state is not a final state:
         report an error and exit
      if state is not WHITESPACE:
         output appropriate token
      state = START ← restart again!
      if i == k:
         exit
   else:
      state = newstate
      i = i + 1
```

chop up whitespace

] move ahead in my input

Next Steps

A5P9



one token recognizer

↑ midterm + final