CSE 105
THEORY OF COMPUTATION

Spring 2018

http://cseweb.ucsd.edu/classes/sp18/cse105-ab/

Today's learning goals

Sipser Section 1.3, 1.4

- Convert between regular expressions and automata
- Choose between multiple models to prove that a language is regular
- Describe the limits of the class of regular languages

DFA equiv NFA

Theorem: For each language L,

L is recognizable by some DFA

iff

L is recognizable by some NFA

Details

Suppose M = (Q, Σ, δ,q0, F) is a DFA. An **equivalent NFA** is
N = (Q, Σ, δ',q0, F) where

$$\delta'((q,x)) = \begin{cases} \{\delta((q,x))\} & \text{if } q \in Q,\ x \in \Sigma \\ \emptyset & \text{if } q \in Q,\ x = \varepsilon \end{cases}$$

Conversely, suppose N = (Q,Σ,δ,q0,F) is a NFA. An **equivalent DFA** is
M = (Q', Σ, δ',q0', F') with Q' = the power set of Q = { X | X is a subset of Q }
q0' = { q0 } U δ((q0, ε)) … i.e. states accessible from q0 via spontaneous moves
F' = {X | X and F are not disjoint}
δ' ((X, x)) = { q in Q | q is in δ((r,x)) for some r in X or is accessible via spontaneous moves}

DFA equiv NFA equiv RegExp

Theorem: For each language L,

L is recognizable by some DFA

iff

L is recognizable by some NFA

iff

L is describable by some regular expression

From RegExp to DFA

Structural induction!

1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$

Base cases

4. $R = (R_1 \cup R_2)$
5. $R = (R_1 \circ R_2)$
6. (R_1^*)

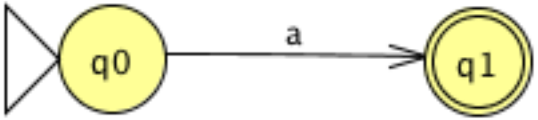
Inductive steps

- Build DFAs (or NFAs) corresponding to base cases in inductive definitions of regular expressions.
- Describe constructions for DFAs corresponding to each of the inductive steps: union, concatenation, Kleene star.

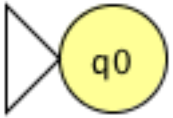
Structural induction

Thm 1.45, 1.47, 1.49 pp 59-62

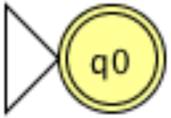
- Base cases



#1



#2



#3

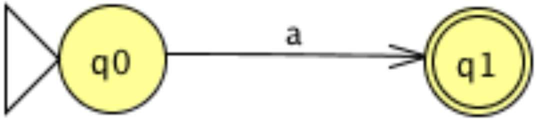
Which of these recognizes L(ø) ?

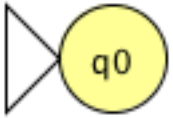
A. NFA #1
B. NFA #2
C. NFA #3
D. More than one of the above
E. None of the above

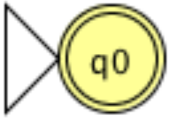
Structural induction

Thm 1.45, 1.47, 1.49 pp 59-62

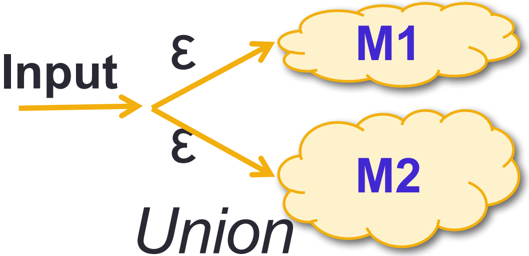
- Base cases





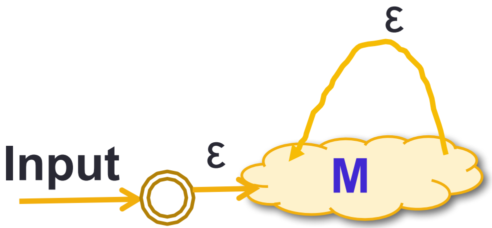


- Inductive steps



Union

Concatenation



Kleene star

Example

a*(ab)*

From DFA to RegExp

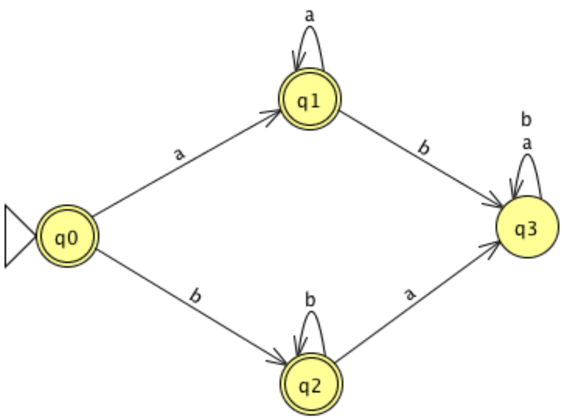
Sipser pp. 70-71

Trace possible paths from start state to accept state.

Intermediate machines (called Generalized NFA) can have ***regular expressions*** on transitions.

First

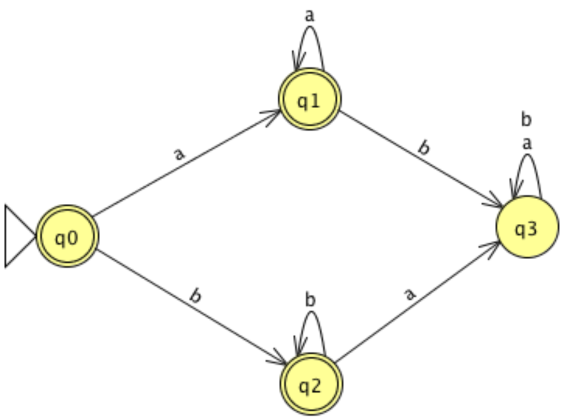
1. add new start state that has ε arrow ***to*** old start state
2. add new accept state that has ε arrows ***from*** old accept states (and ø arrows *from* nonaccept states)


a
q1
a
b
b
a
q0
q3
b
b
a
q2

From DFA to RegExp

Remove one state at a time.

- Restore automaton by modifying regular expresssions on transitions that went through removed state.



a
q1
a
b
b
a
q0
q3
b
b
a
q2

Regular languages

To prove that a set of strings is regular:

1. Build a **DFA** whose language is this set. OR
2. Build an **NFA** whose language is this set. OR
3. Build a **regular expression** describing this set. OR
4. Use the **closure** properties of the class of regular languages to construct this set from others known to be regular (complementation, union, intersection, concatenation, Kleene star, flipbits, reverse, etc.)

All roads lead to … regular sets?

Are there any languages over {0,1} that are **not regular**?

A. Yes: a language that is recognized by an NFA but not any DFA.

B. Yes: there is some finite language of strings over {0,1} that is not described by any regular expression.

C. No: all languages over {0,1} are regular because that's what it means to be a language.

D. No: for each set of strings over {0,1}, some DFA recognizes that set.

E. None of the above.

For next time

- Work on Group Homework 2 **due Saturday**

Pre class-reading for Wednesday: page 77.

Counting languages

How many languages over {0,1} are there?

A. Finitely many because {0,1} is finite.
B. Finitely many because strings are finite.
C. Countably infinitely many because {0,1}* is countably infinite.
D. Uncountably many because languages are in the power set of {0,1}*.
E. None of the above.

Counting regular languages over {0,1}

| { regular languages} | ≤ | { regular expressions } |

Each regular expression is a finite string over the alphabet

{ 0 , 1 , ε , ø , (,) , U , * }

The set of strings over an alphabet is countably infinite.

Conclude: countably infinitely many regular languages.

Where we stand

Fact 1: There exist nonregular languages.

Fact 2: If we know some languages are nonregular, we can conclude others must be too. (cf. Discussion)

But, we don't have any specific examples of nonregular languages.

Yet.