CSE 105
THEORY OF COMPUTATION

Spring 2018

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

Today's learning goals

Sipser Section 1.1

- Construct finite automata using algorithms from closure arguments
- Determine if a finite automaton is nondeterministic
- Trace the computation(s) of a nondeterministic finite automaton
- Describe the language of an NFA

General proof structure/strategy

Theorem: **For any L over Σ, if L is regular then** [the result of some operation on L] **is also regular.**

Proof:

Given name variables for sets, machines assumed to exist.

WTS state goal and outline plan.

Construction using objects previously defined + new tools working towards goal. Give formal definition and explain.

Correctness prove that construction works.

Conclusion recap what you've proved.

The regular operations

Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

$$A \cup B = \{x | x \in A \text{ or } x \in B\}$$ ✔

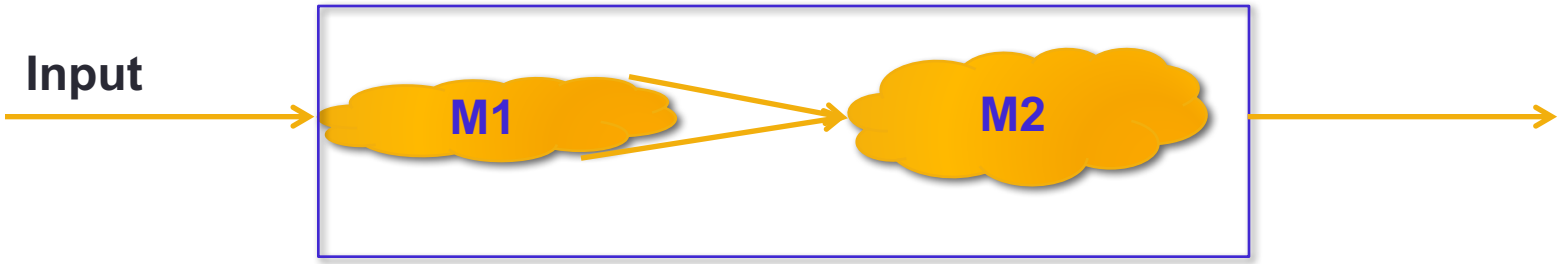
$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

$$A^* = \{x_1 x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$$

How can we prove that the concatenation of two regular languages is a regular language?

Concatenation

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

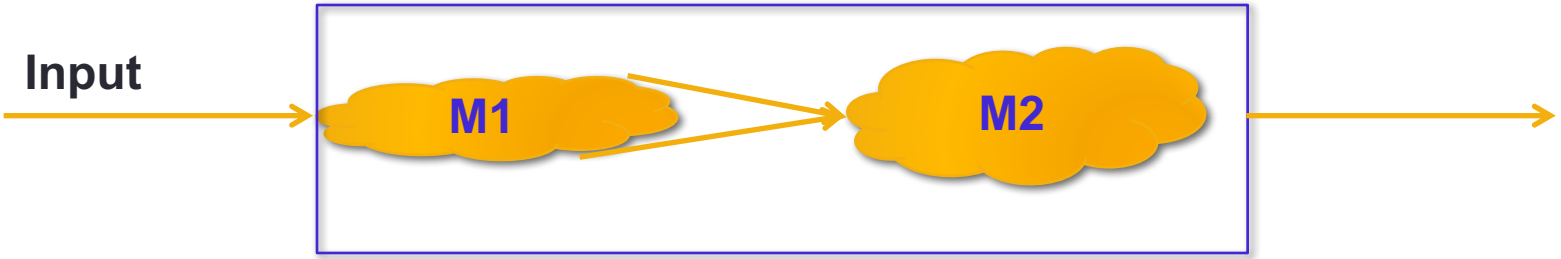


How would this work for A = L(a* U b*), B = $\{b^{2m+1} \mid m \geq 0\}$?

Solution: nondeterminism

- Allow transitions that don't "consume" any input
- Instead of exactly one outgoing transition from each state labelled by each symbol
  - Can have 0 transitions
  - Can have > 1 transitions

Concatenation



Allow to move between M1 and M2 "spontaneously" when reach an accepting state of M1.

Corresponds to guess that string we've read so far is in L(M1) and rest of the input will be in L(M2).

Plan

1. Use nondeterminism to build machines that will recognize the outputs of these set operations.

2. Show that this nondeterminism is not essential: nondeterministic machines can be converted to deterministic ones.

Differences between NFA and DFA

- **DFA**: unique computation path for each input
- **NFA**: allow several (or zero) alternative computations on *same input*
  - δ(q,x) may specify **more than one** possible next states
  - ε transitions allow the machine to **transition between states spontaneously**, without consuming any input symbols

Formal definition of NFA

Def 1.37 p. 53

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called the states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

Which piece of the definition of NFA means there might be **more than one** possible next state from a given state, when reading symbol x from the alphabet?

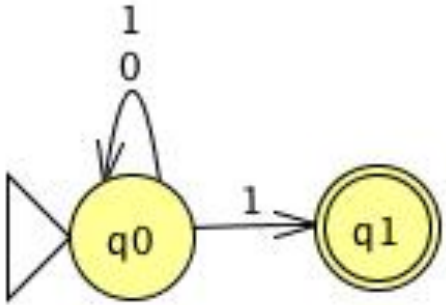
A. Line 2, the size of Σ
B. Line 3, the domain of δ
C. Line 3, the codomain of δ
D. Line 5, that F is a set
E. I don't know.

Acceptance in an NFA

An NFA $(Q, \Sigma, \delta, q_0, F)$ accepts a string w in Σ^* iff we can write $w = y_1 y_2 \cdots y_m$ where each $y_i \in \Sigma_\varepsilon$ and **there is** a sequence of states $r_0, \ldots, r_m \in Q$ such that

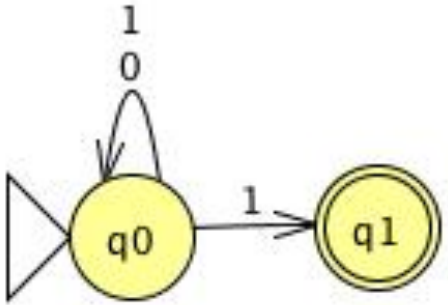
1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for each $i = 0, \ldots, m-1$
3. $r_m \in F$.

Tracing NFA execution



- Is 0 accepted?
- Is 1 accepted?
- Find a length 3 string that is accepted.
- Find a length 3 string that is rejected.
- Is the empty string accepted?

Tracing NFA execution



What's an example of a NFA, N, whose start state is not accepting but where ε is in L(N)?

A. Impossible!
B. Modify picture above by adding double circle at q0.
C. Modify picture above by removing self loops at q0.
D. Modify picture above by adding spontaneous move from q0 to q1.
E. I don't know.

…structure i was using to display the output was a TreeCtrl, so I figured some kind of weird tree traversal system would work and i made it into **this implicit state machine** that used a stack to traverse up and down into different levels. In short, it was an absolute monster of a structure (which i am immensely proud of for continuing to work as new requirements were added every couple of hours :p). Needless to say, all that code is now gone. Tuesday I came in on a mission to reimplement the tree without any interruption in the continual updating of my project. This was made possible by mentor deciding to just disappear into thin air for the day. I designed am **NFA(thanks be Shacham)** that could cover every case and also allow the timeline to be completely extensible. Note, I needed an NFA because there are a few input strings that require up to 5 transitions (the tree gets really deep) and the states couldn't really be just jumped to because of the way the TreeCtrl works. After lunch, I implemented this NFA (as it turns out, there was a slight amount of copy and paste) and had it working and fully functional within an hour. On Wednesday, I arranged a short meeting with my mentor to show it off and discuss some requirements for a side project I was to begin working on. When my mentor showed up to the meeting, he brought with him a Qualstar for me! Qualstars are internal awards given to employees who exceed expectations and provide excellent work (or something like that) and I got one for completing my first project so quickly and saving engineer time.

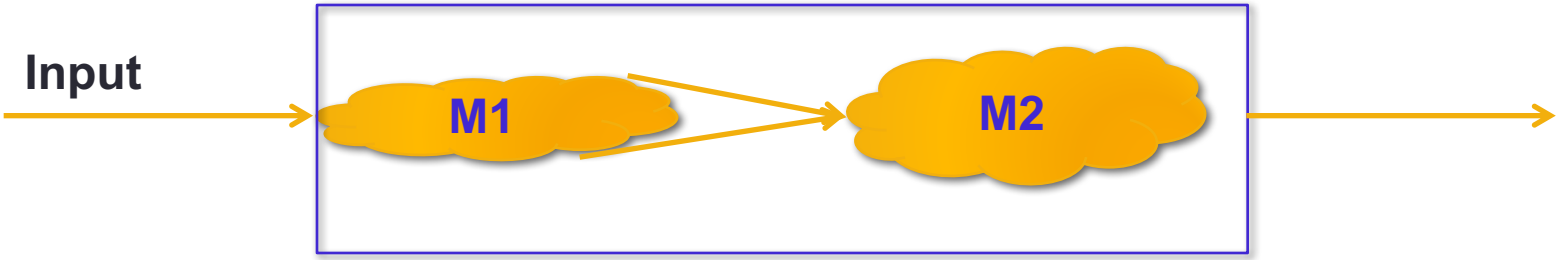
I FINALLY USED SOMETHING I LEARNED AT UCSD WHICH I NEVER THOUGHT I WOULD USE!!!!! i used CSE 105 to design my NFA and knew that an NFA would be a good way to solve this problem only because of that class. Also, that Qualstar i got kinda feels like a real world A+, except it actually means something :p

Chris Miranda (CSE 197)

Concatenation, formally

pp. 60-61

- "Guess" some stage of input at which switch modes



Given M1 = (Q1,Σ,δ1,q1,F1), M2 = (Q2,Σ,δ2,q2,F2) build

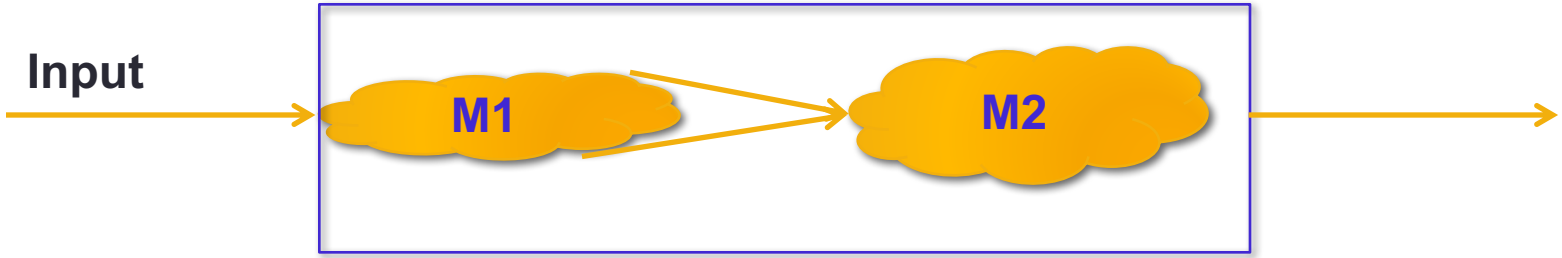
N = (…, Σ, δ, q1, F2) with δ…

More differences between NFA and DFA

- **DFA**: unique computation path for each input
- **NFA**: allow several (or zero) alternative computations on *same input*
  - δ(q,x) may specify **more than one** possible next states
  - ε transitions allow the machine to **transition between states spontaneously**, without consuming any input symbols
- Types of components of formal definition
  - **DFA** $\delta : Q \times \Sigma \to Q$
  - **NFA** $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$

Concatenation, formally

pp. 60-61



$\delta((q, x)) =$

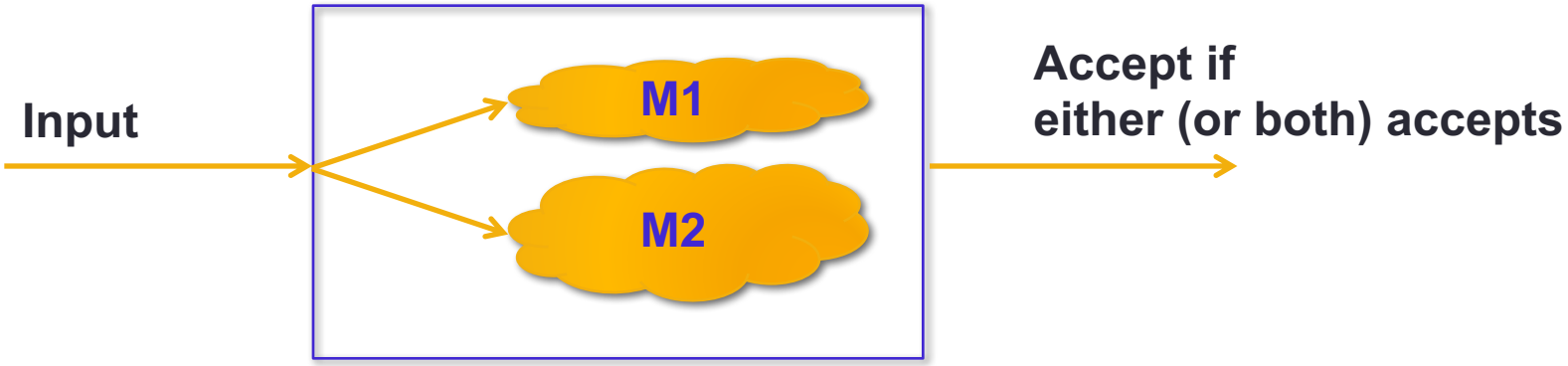
if q is in Q1, x is in Σ

if q is in Q2, x is in Σ

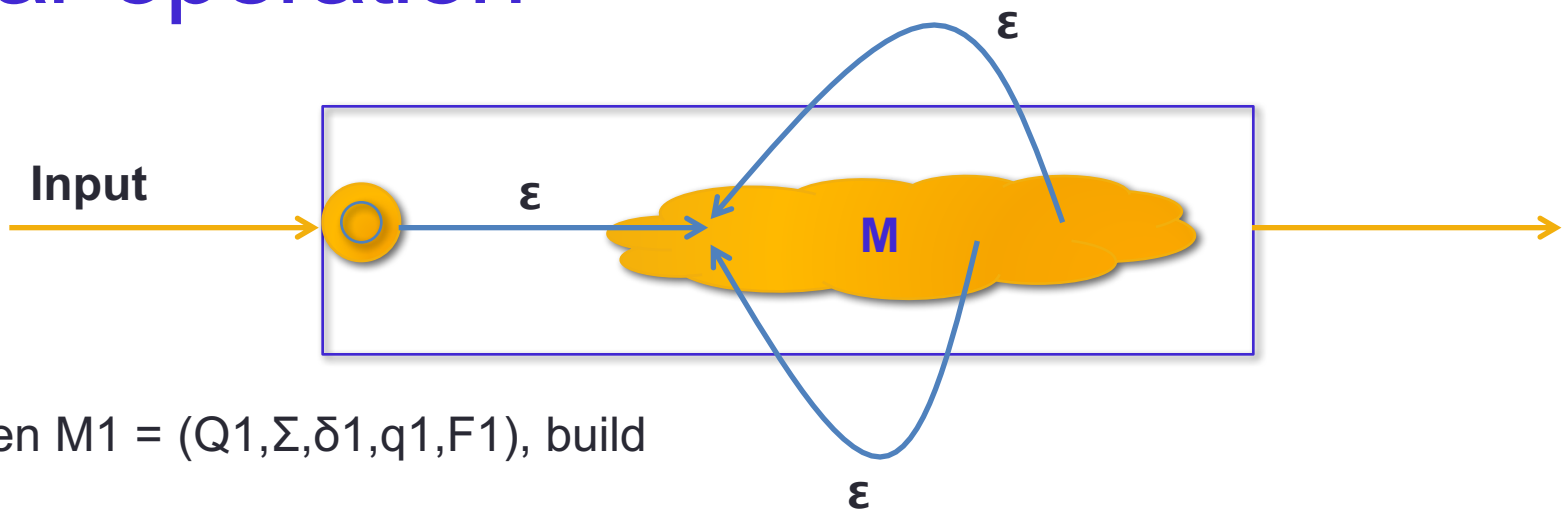
?

New – easier – construction for union

- "Guess" one of finite list of criteria to meet



Star operation



Given M1 = (Q1,Σ,δ1,q1,F1), build

N = (Q1 U {q0}, Σ, δ, q0, F1 U {q0})

and δ(q,x) = …

Construction in the book (page 63)

For next time

- Work on Group Homework 1 **due Saturday**

Pre class-reading for Friday: Page 55.