CSE 105
THEORY OF COMPUTATION

Fall 2016

http://cseweb.ucsd.edu/classes/fa16/cse105-abc/
Today's learning goals

- Design NFA to recognize a given language
- Compare properties of regular and NFA-recognizable languages
- Convert an NFA (with or without epsilon transitions) to a DFA recognizing the same language
Theorem: The class of regular languages over fixed alphabet $\Sigma$ is closed under the union operation.

Proof: Let $A_1, A_2$ be any two regular languages over $\Sigma$. WTS that $A_1 \cup A_2$ is regular.

Goal: build a machine that recognizes $A_1 \cup A_2$. 
Goal: build a machine that recognizes $A_1 \cup A_2$.

Strategy: use machines that recognize each of $A_1$, $A_2$.

Accept if either (or both) accepts

** HOW? **
Theorem: The class of regular languages over fixed alphabet $\Sigma$ is closed under the union operation.

Proof: Let $A_1$, $A_2$ be any two regular languages over $\Sigma$. Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(M_2) = A_2$ and \textbf{WTS} that $A_1 \cup A_2$ is regular.

Define $M = (?, \Sigma, \delta, ?, ?)$
Theorem: The class of regular languages over fixed alphabet $\Sigma$ is closed under the union operation.

Proof: Let $A_1, A_2$ be any two regular languages over $\Sigma$. Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(M_2) = A_2$ and WTS that $A_1 \cup A_2$ is regular.

Define $M = (Q_1 \times Q_2, \Sigma, \delta, ?, ?)$ where all possible pairs of states $(q, q')$ where $q \in Q_1$ and $q' \in Q_2$.
Theorem: The class of regular languages over fixed alphabet $\Sigma$ is closed under the union operation.

Proof: Let $A_1, A_2$ be any two regular languages over $\Sigma$. Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(M_2) = A_2$ and WTS that $A_1 \cup A_2$ is regular. Define $M = (Q_1 \times Q_2, \Sigma, \delta, ?, ?)$.

What should be the initial state of $M$?

A. $q_0$
B. $q_1$
C. $q_2$
D. $(q_1, q_2)$
E. I don't know.
The class of regular languages over fixed alphabet \( \Sigma \) is closed under the union operation.

Proof: Let \( A_1, A_2 \) be any two regular languages over \( \Sigma \). Given \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) such that \( L(M_1) = A_1 \) and \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) such that \( L(M_2) = A_2 \) and WTS that \( A_1 \cup A_2 \) is regular.

Define \( M = (Q_1 \times Q_2, \Sigma, \delta, ?, ?) \),

When \( r \) is a state in \( M_1 \), \( s \) is a state in \( M_2 \), and \( x \) is in \( \Sigma \), then \( \delta( (r,s), x ) = \)

A. \( (r,s) \)
B. \( ( \delta(r,x), \delta(s,x) ) \)
C. \( ( \delta_1(r,x), s ) \)
D. \( ( \delta_1(r,x), \delta_2(s,x) ) \)
E. I don't know.
Theorem: The class of regular languages over fixed alphabet $\Sigma$ is closed under the union operation.

Proof: Let $A_1, A_2$ be any two regular languages.

Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(M_2) = A_2$ and

WTS that $A_1 \cup A_2$ is regular.

Define $M = (Q_1 \times Q_2, \Sigma, \delta, q_1 \times q_2, F_1 \times F_2)$

The set of accepting states for $M$ is

A. $F_1 \times F_2$
B. $\{ (r,s) \mid r \text{ is in } F_1 \text{ and } s \text{ is in } F_2 \}$
C. $\{ (r,s) \mid r \text{ is in } F_1 \text{ or } s \text{ is in } F_2 \}$
D. $F_1 \cup F_2$
E. I don't know.
Proof: Let $A_1, A_2$ be any two regular languages over $\Sigma$. Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ such that $L(M_2) = A_2$. 

WTS that $A_1 \cup A_2$ is regular. Define

$M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), \{(r, s) \in Q_1 \times Q_2 \mid r \in F_1 \text{ or } s \in F_2\})$

with \( \delta((r, s), x) = (\delta_1(r, x), \delta_2(s, x)) \) for each $(r, s)$ in $Q_1 \times Q_2$ and $x$ in $\Sigma$.

Claim that $L(M) = A_1 \cup A_2$. Proof…
Intersection

• How would you prove that the class of regular languages is closed under intersection?
• Can you think of more than one proof strategy?

\[ A \cap B = \{ x \mid x \text{ in } A \text{ and } x \text{ in } B \} \]

\[ = (\overline{A} \cup \overline{B}) \]

*Proof:* Let \( A, B \) be regular languages. WTS \( A \cap B \) reg.

By Def. \( A \cap B = (\overline{A} \cup \overline{B}) \) and since class of reg. langs. is closed under \( \cup \), \( \overline{\cap} \), this set is regular.
Payoff

\{ w \mid w \text{ contains neither the substrings aba nor baab}\}

Is this a regular set?
Payoff

\[ \{ w \mid \text{w contains neither the substrings \text{aba} nor \text{baab}} \} \]

Is this a regular set?

\[ A = \{ w \mid \text{w contains \text{aba} as a substring} \} \]

\[ B = \{ w \mid \text{w contains \text{baab} as a substring} \} \]

\[ \overline{A} \cap \overline{B} = \overline{A \cup B} \]

M using complement. M₂

M using product construction applied to M₁, M₂
Sample closure proofs

- The class of regular languages over \{0,1\} is closed under the FlipBits operation, where
  \[
  \text{FlipBits}(L) = \{ w | w \text{ is obtained from some } w' \text{ in } L \text{ by flipping each 0 in } w \text{ to 1, and each 1 to 0}\}
  \]

- The class of regular languages of \{a,b,z\} is closed under the DeleteWordsWithZ operation, where
  \[
  \text{DeleteWordsWithZ}(L) = \{ w | w \text{ is in } L \text{ and } w \text{ doesn't contain } z\}
  \]
General proof structure/strategy

**Theorem:** For any $L$ over $\Sigma$, 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

For A, B languages over same alphabet, define:

\[ A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \]

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

\[ A^* = \{ x_1x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \]

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

- "Guess" some stage of input at which switch modes
- "Guess" one of finite list of criteria to meet
Example: choose between options

\{ w \in \{0,1\}^* \mid w \text{ has at least two } 0\text{s or at least two } 1\text{s} \}
Example: switch modes

\{ w \in \{0,1\}^* \mid w \text{ ends with } 010 \}
Differences between NFA and DFA

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

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

\[ \delta(q, x) = \{ q', q'', \ldots \} \]

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 \(\Sigma\)
- B. Line 3, the domain of \(\delta\)
- C. Line 3, the codomain of \(\delta\)
- D. Line 5, that \(F\) is a set
- E. I don't know.
Tracing NFA execution

- Is 0 accepted? ✗
- Is 1 accepted? ✓
- Is 0101 accepted? ✓
- Is 110 accepted? ✗
- Is the empty string accepted? ✗

\[ \{ w \mid w \in \{0,1\}^* \} \]
Acceptance in an NFA

An NFA \((Q, \Sigma, \delta, q_0, F)\) accepts a string \(w\) in \(\Sigma^*\) iff we can write \(w = y_1y_2 \cdots y_m\) where each \(y_i \in \Sigma \epsilon\) 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. \(r_m \in F\).
More differences between NFA and DFA

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

Types of components of formal definition

- **DFA** $\delta : Q \times \Sigma \rightarrow Q$
- **NFA** $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$
Similarities between DFA and NFA

- If $L$ is a language recognized by a DFA, is there some NFA that recognizes it?

A. Yes  
B. No  
C. Depends on $L$  
D. I don't know.

Proof: Let $L$ be recognized by DFA $M = (Q, \Sigma, \delta, q_0, F)$. We show there's NFA $N$ s.t. $L(N) = L$.

Put $N = (Q, \Sigma, \tilde{\delta}, q_0, F)$ where $\tilde{\delta}(q, x) = \{ \delta(q, x) \}$ if $x = \varepsilon$ and $\tilde{\delta}(q, x) = \emptyset$ if $x \neq \varepsilon$.
Similarities between DFA and NFA

• If L is a language recognized by an NFA, is there some DFA that recognizes it (aka is it regular)?

A. Yes
B. No
C. Depends on L
D. 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 based 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. Needless to say, my mentor was not impressed with my NFA and gave me some more work to do...

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)
For next time

Homework 2 due tonight

Regrade requests for HW1 must be submitted by noon tomorrow.

**Budget for invalid regrade requests**