CSE 105
THEORY OF COMPUTATION

Spring 2016

http://cseweb.ucsd.edu/classes/sp16/cse105-ab/
Today's learning goals

- Define the regular operations on languages
- Prove closure properties of the class of regular languages
- Trace nondeterministic finite automata to determine whether a string is accepted
- Distinguish between an NFA and a DFA
- Explain why nondeterminism can help
- Design NFA to recognize a given language
Complementation

**Claim**: If \( A \) is a regular language over \( \{0,1\}^* \), then so is \( \overline{A} \) aka "the class of regular languages is closed under complementation"

**Proof**: Let \( A \) be a regular language. Then there is a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) such that \( L(M) = A \). We want to build a DFA whose language is \( \overline{A} \). Consider

\[
M' = (Q, \Sigma, \delta, q_0, \overline{F})
\]

Claim of Correctness \( L(M') = \overline{A} \)

Proof of claim… HW2.
Why closure proofs?

• Stretch the power of the model

• General technique of proving a new language is regular

• Puzzle!
Theorem: The class of regular languages is closed under the union operation.

Proof:

What are we proving here?

A. For any set A, if A is regular then so is A U A.
B. For any sets A and B, if A U B is regular, then so is A.
C. For two DFAs M1 and M2, M1 U M2 is regular.
D. None of the above.
E. I don't know.
Union

**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 $A_1 \cup A_2$. 

Sipser Theorem 1.25 p. 45
Goal: build a machine that recognizes $A_1 \cup A_2$.

Strategy: use machines that recognize each of $A_1$, $A_2$. **HOW?**
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 $M = (Q1 \times Q2, \Sigma, \delta, ?, ?)$

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.
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 $M = (Q_1 \times Q_2, \Sigma, \delta, ?, ?)$
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 $A_1 U A_2$ 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 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 \( M = (Q_1 \times Q_2, \Sigma, \delta, ?, ?, A) \).

The set of accepting states for \( M \) is

- A. \( F_1 \times F_2 \)
- B. \{ (r,s) | r \text{ is in } F_1 \text{ and } s \text{ is in } F_2 \} \)
- C. \{ (r,s) | r \text{ is in } F_1 \text{ or } s \text{ is in } F_2 \} \)
- D. \( F_1 \cup F_2 \) **(NOT RIGHT TYPE)**
- 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 $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1,q_2), \{(r,s) \in Q_1 \times Q_2 | r \in F_1 \text{ or } s \in F_2\})$ 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?

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

1. $$F = F_1 \times F_2 = \{ (r, s) \mid r \in F_1 \text{ and } s \in F_2 \}$$

2. $$A \cap B = \bar{A} \cup \bar{B} \Rightarrow A \cap B = (\bar{A} \cup \bar{B})$$
Payoff

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

Is this a regular set?
Payoff

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

Is this a regular set?

A = \{ w \mid w \text{ contains } aba \text{ as a substring} \}
B = \{ w \mid w \text{ contains } baab \text{ as a substring} \}

\overline{A} \cap \overline{B} = \overline{A \cup B}
Sample closure proofs

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

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

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

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

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

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

- "Guess" some stage of input at which switch modes
- "Guess" one of finite list of criteria to meet

Accept if either (or both) accepts
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 \text{ 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. \(\delta : Q \times \Sigma_e \rightarrow \mathcal{P}(Q)\) 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 \(\Sigma\)
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? ✗
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 \(y_i \in \Sigma_\varepsilon\) 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. \(r_m \in F\).
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** $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$
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.
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.
Next steps

- Defining NFA to recognize specific languages.
- Showing that NFA and DFA are equally expressive
- Using NFA to prove closure of class of regular languages under (the rest of the) regular operations
For next time

Reading quiz 2 due tomorrow
  • On TritonEd

Homework 2 due Friday
  • Closure proofs – see Discussion slides
  • NFA