CSE 105
THEORY OF COMPUTATION

Fall 2021

http://cseweb.ucsd.edu/classes/fa21/cse105-a/
Today's learning goals  Sipser Ch 1.1, 1.2

• More on closure properties of regular languages
  – Complement, union, intersection … (last time)
  – Concatenation, Star, Reverse: how?
• Non-determinism
  – Define Non-deterministic Finite Automata (NFA)
  – Using NFAs to prove closure properties
The regular operations  Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

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

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

\[ A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \]

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 0s or at least two 1s}\}
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
  - computation can *get stuck* at some state, if there's a missing arrow
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. \(\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)\) 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 \(\Sigma\)
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?
Tracing NFA execution

The language recognized by this NFA is ...
Acceptance in an NFA

An NFA \((Q, \Sigma, \delta, q_0, F)\) accepts a string \(w\) in \(\Sigma^*\) iff we can write \(w = y_1 y_2 \cdots y_m\) where each \(y_i \in \Sigma_e\) 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
Designing NFA

Design an NFA which recognizes the language

\{ w0 in \{0,1\}^* | |w| is a multiple of 3 \}
Designing NFA

The language recognized by this NFA is …

The transition function of this NFA can be formally written as …
Simulating NFA with DFA

Not quite a closure proof, but …

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.
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

Proof:

Given $A$, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

WTS there is some DFA $M$ with $L(M) = A$

Construction

Correctness

Conclusion
From NFA to DFA

What is the tree of computation paths?
From NFA to DFA

- Transform the following NFA into an equivalent DFA
- Idea: Use subsets of \{q_0,q_1,q_2,q_3\} as states
Subset construction

**Given** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

**WTS** there is some DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \}$  \(\text{ (assuming no } \epsilon\text{-transitions)}\)
- $F' = \{ X \mid X \text{ is a subset of } Q \text{ and } (X \cap F) \text{ is nonempty} \}$
- $\delta'(\quad) =$
Subset construction

**Given** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

**WTS** there is some DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with

- $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
- $q_0' = \{ q_0 \}$ (assuming no $\varepsilon$-transitions)
- $F' = \{ X \mid X \text{ is a subset of } Q \text{ and } (X \cap F) \text{ is nonempty} \}$
- $\delta' (X, x) = \{ q \in Q \mid q \text{ is in } \delta(r, x) \text{ for some } r \in X \}$
Subset construction example

How big is $Q'$?

A. 2
B. 4
C. 5
D. 16
E. I don't know
What is the initial state $q_0'$?

A. $q_0$
B. $q_3$
C. $\{q_0,q_1,q_2,q_3\}$
D. $\{q_0\}$
E. I don't know
Subset construction example

NFA

DFA
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

Proof:

Given A, a language recognized by N = (Q,Σ,δ,q0,F) a NFA

WTS there is some DFA M with L(M) = A

Construction Define M = (Q', Σ, δ',q0', F') with Q' = P(Q),...

q0'={q0}, δ' ( X, x ) = { q in Q | q is in δ(r,x) for some r in X }  

Correctness ??

Conclusion
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

**Proof:**

**Given** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$ a NFA

**WTS** there is some DFA $M$ with $L(M) = A$

**Construction** Define $M = (Q', \Sigma, \delta', q_0', F')$ with …

**Correctness** ??

**Conclusion**

Details, with epsilon transitions: Sipser 55-56
Application

A language $A$ over $\Sigma$ is **regular** if and only if

- it is recognized by a DFA
- it is recognized by a NFA

To prove that the class of regular languages is closed under operation …. :

Let $A$ be a regular language, so recognized by DFA $M$. Build a **NFA** that recognizes the result of …. on $A$. Conclude this result is also a regular language.
For next time

Wednesday (tomorrow!):

• Homework 2 due
• Homework 3 out (includes Thursday lecture)