Closure results for regular languages

The class of regular languages (i.e. languages recognized by some DFA) is closed under the following operations:

- complement
- union
- intersection
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:
1. **Given.** Name the variables for sets, machines assumed to exist.
2. **Goal.** State what you want to show.
3. **Construction.** Use objects previously defined + new tools working towards goal. **Explain how to construct new machine.**
4. **Strategy.** Explain why construction recognizes desired language.
5. **Recap.** Your construction shows closure under operation.
The regular operations  Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

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

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

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

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

- "Guess" some point in input when we should switch modes
- "Guess" which criteria to meet

Accept if either (or both) accepts
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 state
  - $\epsilon$ 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. \(\delta : Q \times \Sigma \to P(Q)\) 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 inputs of \(\delta\)
- C. Line 3, the outputs of \(\delta\)
- D. Line 5, that \(F\) is a set
- E. I don't know.
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 state
  - $\epsilon$ 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** $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$
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 \(y_i \in \Sigma_\varepsilon\) 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?
- Is 0101 accepted?
- Is 110 accepted?
- Is the empty string accepted?
- Can you describe the language of this machine?
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.
Simulating NFA with DFA

Not a closure proof, but same structure.

Proof:
1. **Given.** Name the variables for sets, machines assumed to exist.
2. **Goal.** State what you want to show.
3. **Construction.** Use objects previously defined + new tools working towards goal. **Explain how to construct new machine.**
4. **Strategy.** Explain why construction recognizes desired language.
5. **Recap.** Your construction shows any NFA has an equivalent DFA.
Simulating NFA with DFA

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

Proof:
1. **Given.** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$, a NFA
2. **Goal.** There is some DFA, $M$, with $L(M) = A$
3. **Construction.**

4. **Strategy.**
5. **Conclusion.**
From NFA to DFA

What is the tree of computation paths on input 0100?
Subset construction

1. **Given.** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$, a NFA
2. **Goal.** There is some DFA, $M$, with $L(M) = A$
3. **Construction.** Define $M = (Q', \Sigma, \delta', q_0', F')$ with
   - $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
   - $q_0' = \{ q_0 \}$
   - $F' = \{ X \mid X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
   - $\delta' (\quad) =$
Subset construction

1. **Given.** A, a language recognized by \( N = (Q, \Sigma, \delta, q_0, F) \), a NFA
2. **Goal.** There is some DFA, \( M \), with \( L(M) = A \)
3. **Construction.** Define \( M = (Q', \Sigma, \delta', q_0', F') \) with
   - \( Q' = \) the power set of \( Q = \{ X \mid X \text{ is a subset of } Q \} \)
   - \( q_0' = \{ q_0 \} \)
   - \( F' = \{ X \mid X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \} \)
   - \( \delta' (X, a) = \) What are the arguments of \( \delta' \)?

   A. \( \delta'(q, a) \) where \( q \) in \( Q \) and \( a \) in \( \Sigma \)
   B. \( \delta'({q}, a) \) where \( q \) in \( Q \) and \( a \) in \( \Sigma \epsilon \)
   C. \( \delta'(X, a) \) where \( X \) is a subset of \( Q \) and \( a \) in \( \Sigma \)
   D. \( \delta'(X, a) \) where \( X \) is a subset of \( Q \) and \( a \) in \( \Sigma \epsilon \)
   E. I don't know
Subset construction

1. **Given.** A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$, a NFA
2. **Goal.** There is some DFA, $M$, with $L(M) = A$
3. **Construction.** Define $M = (Q', \Sigma, \delta', q_0', F')$ with
   - $Q' = \text{the power set of } Q = \{ X \mid X \text{ is a subset of } Q \}$
   - $q_0' = \{ q_0 \}$
   - $F' = \{ X \mid X \text{ is a subset of } Q \text{ and } X \cap F \text{ is nonempty} \}$
   - $\delta' ( X, a ) = \{ q \in Q \mid q \text{ is in } \delta( r, a ) \text{ for some } r \in X \}$
Subset construction example

What 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
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:
1. Given. A, a language recognized by $N = (Q, \Sigma, \delta, q_0, F)$, a NFA
2. Goal. There is some DFA, $M$, with $L(M) = A$
3. Construction. Done.
5. 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 if and only if
- it is recognized by a NFA.

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

Let \( A \) be a regular language, so recognized by some DFA \( M \).

Build a **NFA** that recognizes the result of operation on \( A \).

Conclude this result is also a regular language.
The regular operations  Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

- \( A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \)
- \( A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \)
- \( A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \)

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

"Guess" some point in input when we should switch modes

Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ build

$$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$$ with $\delta$...
\[ \delta(q, x) = \] if \( q \) is in \( Q_1 \), \( x \) is in \( \Sigma \)

[Blank]

[Blank]

[Blank] if \( q \) is in \( Q_2 \), \( x \) is in \( \Sigma \)

[Blank] if [Blank]

Concatenation
Concatenation

\[ \delta(q, x) = \delta_1(q, x) \quad \text{if } q \text{ is in } Q_1, x \text{ is in } \Sigma \]
\[ \delta_2(q, x) \quad \text{if } q \text{ is in } Q_2, x \text{ is in } \Sigma \]
\[ \{q_2\} \quad \text{if } q \text{ is in } F_1, x = \varepsilon \]
\[ \emptyset \quad \text{otherwise} \]

Correctness proof in the book (page 61)
Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, build

$$N = ( Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup \{q_0\} )$$

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

*Construction in the book (page 63)*
Regular languages

To prove that a set of strings over the alphabet $\Sigma$ is regular,

- Build a **DFA** whose language is this set.
- Build an **NFA** whose language is this set.
- Use the **closure properties** of the class of regular languages to construct this set from others known to be regular.

Union
Intersection
Complementation
Concatenation
Reversal
Kleene star