Quiz 3 distribution

- mean = 3.18,  mode = 4.5,  stddev = 1.25
Epsilon-Closure

- def. 0 (informal, Sipser): 
  \( E(q) = \{ p \mid p \text{ is reachable from } q \text{ by 0 or more } \varepsilon \text{ edges} \} \)

- def. 1 (default): \( E(q) \) is the \textbf{smallest} set such that:
  - \( q \in E(q) \)
  - if \( p \in E(q) \), then \( \delta(p, \varepsilon) \subseteq E(q) \)

- def. 2 (inductive): \( E(q) = \bigcup_i E_i(q) \)
  - \( E_0(q) = \{q\} \)
  - \( E_{i+1}(q) = \bigcup_{p \in E_i(q)} \delta(p, \varepsilon) \)

- from closure of state to closure of set:
  - \( E(R) = \bigcup_{q \in R} E(q) \)

\[
E(q_0) = \{q_0, q_1, q_2, q_3, q_5\} \\
E(q_2) = \{q_2, q_3, q_5\}
\]
Computation of NFA w/ epsilon

• string \( w \) is accepted iff. \( \delta^*(q_0, w) \cap F \neq \emptyset \)
• have to refine \( \delta^* \) using epsilon-closure \( E(\ldots) \)

without epsilon
\[
\begin{align*}
\delta &: Q \times \Sigma \mapsto P(Q) \\
\delta^* &: Q \times \Sigma^* \mapsto P(Q)
\end{align*}
\]
\[
\begin{align*}
\delta^*(q, \epsilon) &= \{q\} \\
\delta^*(q, xa) &= \bigcup_{p \in \delta^*(q, x)} \delta(p, a)
\end{align*}
\]
alternatively
\[
\begin{align*}
\delta^*(q, ax) &= \bigcup_{p \in \delta(q, a)} \delta^*(p, x)
\end{align*}
\]

with epsilon
\[
\begin{align*}
\delta &: Q \times (\Sigma \cup \{\epsilon\}) \mapsto P(Q) \\
\delta^* &: Q \times \Sigma^* \mapsto P(Q)
\end{align*}
\]
\[
\begin{align*}
\delta^*(q, \epsilon) &= E(\{q\}) \\
\delta^*(q, xa) &= E\left(\bigcup_{p \in \delta^*(q, x)} \delta(p, a)\right)
\end{align*}
\]
alternatively
\[
\begin{align*}
\delta^*(q, ax) &= \bigcup_{p \in E(\delta(q, a))} \delta^*(p, x)
\end{align*}
\]
Example: NFA=>DFA w/ epsilon

Transition Function?

For each state, we need two edges:

\[ \delta'(1, a) = \emptyset \]

Since no edges labeled "a" leave \( 1 \)

\[ \delta'(1, b) = 2 \]

From \( 1 \) we can get to \( 2 \), but no further with \( \epsilon \)-edges

\[ \delta'(2, a) = 23 \]

\[ \delta'(2, b) = 3 \]

Can get to \( 1 \) but can get to \( 3 \)

\[ \delta'(3, a) = 13 \]

\[ \delta'(3, b) = \emptyset \]

Since no edges labeled "b" leave \( 3 \)

States in the DFA?

\[ Q' = \emptyset \downarrow 3 = P(Q) \]

What states in the DFA?

Start State?

\[ q_0' = E(\emptyset 13) = \emptyset 1, 3 \]

\[ E(1) = 13 \]

Accept states? Any that contain 1

\[ F' = 1 12 13 123 \]
Example: NFA=>DFA w/ epsilon

Note: 1 and 12 are unreachable. They can be removed.
**Theorem**

The class of regular languages is "closed" under union.

"Closure" of a language.

**Proof**

**Formally:**

Let $N_1 = (Q_1, \Sigma, S_1, q_1, F_1)$

$N_2 = (Q_2, \Sigma, S_2, q_2, F_2)$

Construct $N = (Q, \Sigma, S, q_0, F)$

$Q = Q_1 \cup Q_2 \cup \{ q_0 \}$

$q_0$ is new start state

$F = F_1 \cup F_2$

$\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \in Q_1 \\
\delta_2(q, a) & \text{if } q \in Q_2 \\
\{(q_0, q_2)\} & \text{if } q = q_0 \text{ and } a = \epsilon \\
\{\epsilon\} & \text{if } q = q_0 \text{ and } a \neq \epsilon 
\end{cases}$
The class of regular languages is closed under concatenation.

**Theorem**

**Formally**

Let $N_1 = (Q_1, \Sigma, S_1, q_1, F_1)$

$N_2 = (Q_2, \Sigma, S_2, q_2, F_2)$

Construct $N = (Q, \Sigma, S, q_0, F)$

$Q = Q_1 \cup Q_2$

$q_0 = q_1$, the start state of $N_1$

$F = F_2$, the final states of $N_2$

$S(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \in Q_1 \\
\delta_2(q, a) & \text{if } q \in Q_2 \\
\delta_1(q, a) \cup \delta_2(q, a) & \text{if } q \in F_1 \text{ and } a = \epsilon \\
\delta_1(q, a) & \text{if } q \in F_1 \text{ and } a \neq \epsilon
\end{cases}$
**THEOREM**

The class of regular languages is closed under "star."

**PROOF**

Same idea.

\[ A^* = \varepsilon, a, aa, aaa, b, bb, bbb, abbb, bba, bbabb. \]
DFA minimization (Linz 2.4)

• NFA => DFA => minimal DFA (b/c some nodes are equivalent or “mergeable”)

• first compute equivalence classes of nodes
  • initially, two classes: F and Q - F, and keep splitting:
  • if $\delta(p, a)$ and $\delta(q, a)$ fall into different classes, then $p$ and $q$ aren’t equivalent

• merge states in the same class, and form the minimal DFA

initial: $F=\{q_4\}, A=\{q_0, q_1, q_2, q_3\}$
step 1: $\delta(q_0, 1)$ in A
  but $\delta(q_1, 1) = \delta(q_2, 1) = \delta(q_3, 1)$ in F
so split: $F = \{q_4\}, A=\{q_0\}, B=\{q_1, q_2, q_3\}$
no further splitting possible

this partition is unique.
**Definition 1.52**

Say that $R$ is a **regular expression** if $R$ is

1. a for some $a$ in the alphabet $\Sigma$,
2. $\varepsilon$,
3. $\emptyset$,
   also notated: $R_1 | R_2$ or $R_1 + R_2$
4. $(R_1 \cup R_2)$, where $R_1$ and $R_2$ are regular expressions,
5. $(R_1 \circ R_2)$, where $R_1$ and $R_2$ are regular expressions, or
6. $(R_1^*)$, where $R_1$ is a regular expression.  also $R_1 \cdot R_2$

Let $R^*$ be shorthand for $RR^*$. $R^* \cup \varepsilon = R^*$

$R \cup \emptyset = R$.
Adding the empty language to any other language will not change it.

$R \circ \varepsilon = R$.
Joining the empty string to any string will not change it.

$R \cup \varepsilon$ may not equal $R$.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}.$

$R \circ \emptyset$ may not equal $R$.
For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset.$

$$(+ \cup - \cup \varepsilon) (D^* \cup D^* \cdot D^* \cup D^* \cdot D^*)$$

where $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is the alphabet of decimal digits. Examples of generated strings are: 72, 3.14159, +7., and -.01.

**Example 1.53**

In the following instances, we assume that the alphabet $\Sigma$ is $\{0,1\}$.

1. $0^*10^* = \{w | w$ contains a single 1$\}.$
2. $\Sigma^*1\Sigma^* = \{w | w$ has at least one 1$\}.$
3. $\Sigma^*001\Sigma^* = \{w | w$ contains the string 001 as a substring$\}.$
4. $1^*(01^*)^* = \{w |$ every 0 in $w$ is followed by at least one 1$\}.$
5. $(\Sigma\Sigma)^* = \{w | w$ is a string of even length$\}.$
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ the length of $w$ is a multiple of 3$\}.$
7. $01 \cup 10 = \{01, 10\}.$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w | w$ starts and ends with the same symbol$\}.$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*.$
The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or $\varepsilon$ before every string in $1^*$.

10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}.$

11. $1^*\emptyset = \emptyset$.
Concatenating the empty set to any set yields the empty set.

12. $\emptyset^* = \{\varepsilon\}.$
The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.
Examples

Parsing Practice

\[
\begin{align*}
aa \ bUc \ aa \ bUc \ aa &= \ ? \\
(aab) \ U \ (caab) \ U \ (ca) \\
&\Rightarrow aab | caab | caa \\
\d U \ ab^* \ cd^* &= \ ? \\
\d | \ ab^* \ cd^* &= \ ? \\
&\Rightarrow (d) \ U \ (a(b^*) \ c(d^*)) \\
&\Rightarrow d \ | \ (a(b^*) \ c(d^*))
\end{align*}
\]

Example Regular Expressions

Assume \( \Sigma = \{a, b, c, d\} \)

\[
\begin{align*}
a &\in \Sigma \\
abcc &\in \Sigma \\
ab &\cup \ cd &\Rightarrow ab/cd \\
&\in \Sigma \\
a(bUc) &d = a(b/c)d \\
&\in \Sigma \\
ab &c &\Rightarrow ac, abc, abbc, abbbbc, \ldots \\
a(bUe) &c = a(b/e)c = a[b/c]c \\
&\in \Sigma \\
\emptyset &\in \Sigma \\
a(bUc) &\emptyset \\
&\in \Sigma \\
\emptyset^* &\in \Sigma
\end{align*}
\]
Regular Expression == Regular Language

**Theorem**
A language is regular iff some regular expression describes it.

**Previous Definition**
A language is regular iff it is described by some finite state machine. (NFA = DFA).

**Regular Expression**
Each regular expression describes a language. Which language?

\[ L(R) = ? \]

- \( L(\alpha) = \{ \alpha \} \)
- \( L(R_1 \cup R_2) = L(R_1) \cup L(R_2) \)
- \( L(R_1 \circ R_2) = L(R_1) \circ L(R_2) \)
- \( L(R_1^*) = L(R_1)^* \)
- \( L(\emptyset) = \{ \emptyset \} \)
- \( L(R_1) = L(R_1) \)

Regular languages are closed under union, concatenation, and star.

\[ \Rightarrow \text{Regular Expressions describe regular languages.} \]

So we are saying that the class of languages recognized by DFA's, NFA's, and Reg. Expressions is the same! All have equivalent "power"!
Regular Expression == Regular Language

**Lemma 1:**

If a language is described by a regular expression, then it is regular.

Proof #1: Use the closure of $U$ and $\cdot$.

Proof #2: From a regular expression, build an NFA to recognize it.

**Lemma 2:**

If a language is regular, then it can be described by a regular expression.

Proof Approach:

- Start with a DFA that recognizes it.
- Build a GNFA (generalized nondeterministic finite state automaton).
- Reduce it (details to follow).
- This yields a regular expression.

Every large regular expression is made of smaller regular expressions.

Assume we can build the NFAs for smaller regular expressions.

Show how to build the NFA for larger regular expressions.
**RE => NFA**

**Definition of Regular Expressions**

- $R = a$
- $R = R_1 \cup R_2$
- $R = R_1 \circ R_2$
- $R = R_1^*$
- $R = \epsilon$
- $R = \emptyset$

- $R = a$
- $R = \epsilon$
- $R = \emptyset$

- $R = R_1 \cup R_2$
- $R = R_1 \circ R_2$
- $R = R_1^*$

Same construction used in proof of closure.