Regular Expressions

Short hands

\[ L((a|b)c) = L(a^*b^*c^*) \]

\[ L(((a|b)c)(a|b)) = \{a, b, c\} \]

Procedural Left Associativity

\[ (a|b)c \]

\[ a|b \cdot c^+ \]

\[ (a|b)(b|(c^*)) \]

\[ L(\emptyset) = \emptyset \]

\[ L(\epsilon) = \{\epsilon\} \]

\[ L(a|b|c) = L(a) \cup L(b) \cup L(c) \]

Regular expressions over \( \Sigma \)

\[ \emptyset \text{ is an RE.} \]

\[ \epsilon \text{ is an RE.} \]

\[ a \text{ is an RE.} \]

\[ a^* \text{ is an RE.} \]

is \( R_1 \) \& \( R_2 \) are REs, then so are

\[ (R_1 \cup R_2) \]

\[ (R_1 \cdot R_2) \]

\[ (R_1^*) \]

The language denoted by \( R, \Sigma \epsilon \)

is:

\[ L(a) = \emptyset \]

\[ L(\epsilon) = \{\epsilon\} \]

\[ L(a|b|c) = L(a) \cup L(b) \cup L(c) \]

"Words ending with "TXT""

\[ \epsilon^*TXT \]

\[ (a|b|c)^*(a|b|c)^*(a|b|c)^* \]

\[ 2|(ld)^* \]

\[ (\Sigma\epsilon)^* \]

\[ (\Sigma^+\epsilon^+\Sigma^+) \]

\[ 0^*10^* \]

\[ a^* \]

\[ (\Sigma\epsilon)(\Sigma\epsilon) \]

\[ \Sigma \]

\[ 0^*(10^*10^*)^* \]

\[ 0^* \]

\[ (0^*10^*10^*)^* \]

\[ 0^*10^*10^* \]

\[ (d^*d^*u|d|d^*) \]

\[ (\Sigma\epsilon)(\Sigma\epsilon) \]

\[ d^+ \]
**Theorem:**
A regular expression $R \in \mathcal{RE}$ and an NFA $M_a \in \mathcal{L}(R) \subseteq \mathcal{LCM}$.

**Proof:**
By induction on $R$, the # of $\lor, \cdot, *$ operators in $R$.

Base cases (Kao):
- Then $R$ is $\emptyset$, $\epsilon$, or $a$ for $a \in \Sigma$.
- Explicitly give simple NFAs recognizing $\emptyset$, $\epsilon$, and $a$ for each $a \in \Sigma$ (details omitted).

Induction Step ($R$ has $k > 0$ operators):
- I.H.: Assume that for all regular expressions $R'$ with $k$ operators, there is an NFA $M_{R'}$ recognizing $\mathcal{L}(R')$.

If $R$ has $k > 0$ operators, let:
- $R = (R_1 \lor R_2)$ or $(R_1 \cdot R_2)$ or $(R_1)^*$
- When $R_1 \cdot R_2$ is any $a \in \Sigma$ and $R_1 \lor R_2$ have $k_1$ operators.
- By I.H., $\exists M_{R_1}, M_{R_2}$ NFA recognizing $\mathcal{L}(R_1), \mathcal{L}(R_2)$.
- Modify $M_{R_2}$ with previous proofs of closure under $\cdot$ to get $M_a \in \mathcal{L}(R_a)$. (Details omitted.)

**Converse?**
For every $D/NFA \exists rgy expr defining the same language.
Every regular language can be described by a regular expression.

**Theorem**

If $L$ is accepted by a GNFA, then $L$ is regular.

**Proof Sketch:**

Replace edge labeled "r" by NFA equivalent to $r$ based on previous theorem.
If $L$ is regular, then $L = L(R)$ for some regular expression $R$.

Proof will take FA for $L$, & reduce it to a (G)NFA for same $L$ with progressively fewer states until $R$ becomes obvious.
In a nutshell, delete state k from G, but enlarge language on each edge to compensate, so that potential contribution of k is added to each edge in G'.

Path in G: 1 2 3 k 4 k k 2 1 ...
Path in G': 1 2 3 4 2 1 ...

Strings in L(edge reg exp)

Claim 2

\[
L(\gamma ij) = \{ w | G \text{ can move from } i \text{ to } j \text{ reading } w \text{ and passing through no intermediate states except possibly } k \}.
\]

Equivalently:

\[
L(\gamma ij) = \{ w | G \text{ can move from } i \text{ to } j \text{ reading } w \text{ along a simple path } \gamma \}
\]

\[
\equiv \{ w \gamma_{i:k} \gamma_{k:j}^* | w \}
\]

Relating edges of G' to paths in G:

A path in G: any sequence of states
A simple path in G: any sequence of 3 states at 1st 2 last are not, and all intermediate ones (if any) are.

The Point:
(a) every path in G can be decomposed into simple paths
(b) every edge in G', say i→j, corresponds to the edge in simple path in G with these end points
Claim 41: A NFA is equivalent to a regular expression.

Proof: NFA $\rightarrow$ GNFA $\rightarrow$ 2-state GNFA $\rightarrow$ RE

Summary:
- $L$ is regular $\Rightarrow$ $L = L(M)$ for some DFA $M$
- $L = L(C(N))$ for some NFA $N$
- $L = L(C(G))$ for some GNFA $G$
- $L = L(CR)$ for a regular expression $R$