DFA to Regular Expressions

Proposition: If L is regular then there is a regular expression r such that $L = L(r)$.

Proof Idea: Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing L, with $Q = \{q_1, q_2, \ldots q_n\}$, and $F = \{q_{f_1}, \ldots q_{f_k}\}$

- Construct regular expression r_{1f_i} such that $L(r_{1f_i}) = \{w : \mid \delta(q_1, w) = q_{f_i}\}$, i.e., r_{1f_i} describes the set of strings on which M ends up in state q_{f_i} when started in the initial state q_1.

- Then

$$\begin{aligned} L = L(M) \quad &= L(r_{1f_1}) \cup L(r_{1f_2}) \cup \cdots \cup L(r_{1f_k}) \\ &= L(r_{1f_1} + r_{1f_2} + \cdots + r_{1f_k}) \end{aligned}$$

Thus, the desired regular expression is $r_{1f_1} + r_{1f_2} + \cdots + r_{1f_k}$

Constructing r_{1f_i}

Idea 1 For every i, j, build regular expressions r_{ij} describing strings taking M from state q_i to state q_j, where q_i need not be the initial state and q_j need not be a final state.

Idea 2 Build the expression r_{ij} inductively.

- Start with expressions that describe paths from q_i to q_j that do not pass through *any* intermediate states; i.e., these are single nodes or single edges.
- Inductively build expressions that describe paths that pass through progressively a larger set of states.

Definitions and Notation

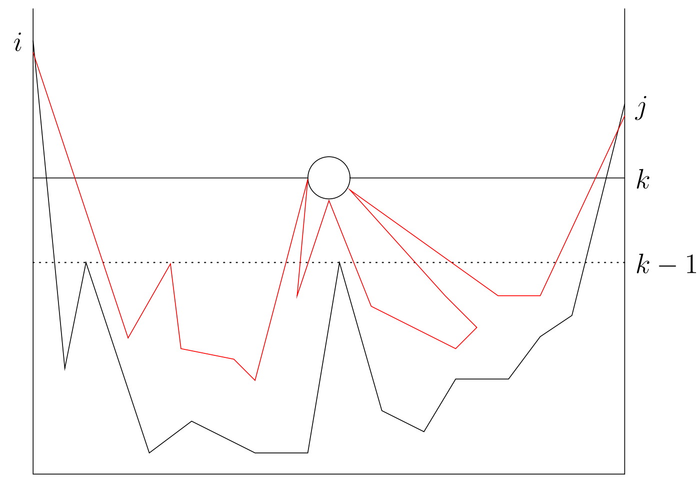
Define R_{ij}^k to be the *set* (not regular expression) of strings leading from q_i to q_j such that any intermediate state is $\leq k$

- Note, the superscript k refers only to the intermediate states; so i and j could be greater than k.
- R_{ij}^0 set of strings that go from q_i to q_j without passing through any intermediate states; in other words they are ϵ or single edges.
- R_{ij}^n is set of all strings going from q_i to q_j

Constructing set R_{ij}^k: Base Case

$$R_{ij}^0 = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{if } i = j \end{cases}$$

Constructing set R_{ij}^{k}: Inductive Step



Assume we have R_{ij}^{k-1}

$$R_{ij}^{k} = R_{ij}^{k-1} \cup R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1}$$

Constructing the Regular Expression

Task: Construct expression r_{ij}^{k} such that $L(r_{ij}^{k}) = R_{ij}^{k}$.

Base Case

$$r_{ij}^{0} = \begin{cases} \emptyset & \text{if } R_{ij}^{0} = \emptyset \\ a_1 + a_2 + \cdots a_m & \text{if } R_{ij}^{0} = \{a_1, a_2, \ldots a_m\} \\ \epsilon + a_1 + \cdots a_m & \text{if } R_{ij}^{0} = \{\epsilon, a_1, \ldots a_m\} \end{cases}$$

Constructing the Regular Expression: Inductive step

Assume inductively, r_{ij}^{k-1} is the regular expression for R_{ij}^{k-1}

$$
\begin{aligned}
R_{ij}^{k} &= R_{ij}^{k-1} \cup R_{ik}^{k-1}(R_{kk}^{k-1})^{*} R_{kj}^{k-1} \\
&= L(r_{ij}^{k-1}) \cup L(r_{ik}^{k-1})(L(r_{kk}^{k-1}))^{*} L(r_{kj}^{k-1}) \\
&= L(r_{ij}^{k-1} + r_{ik}^{k-1}(r_{kk}^{k-1})^{*} r_{kj}^{k-1})
\end{aligned}
$$

$r_{ij}^{k-1} + r_{ik}^{k-1}(r_{kk}^{k-1})^{*} r_{kj}^{k-1}$ is the Regular Expression for R_{ij}^{k}.

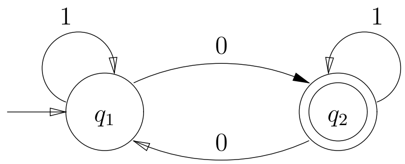
Completing the Proof

Proposition: If L is regular then there is a regular expression r such that $L = L(r)$.

Proof: Let q_1 be the initial state, and $\{q_{f_1}, q_{f_2}, \ldots q_{f_k}\}$ the final states of M (which recognizes L), then the desired regular expression is

$$r^n_{1f_1} + r^n_{1f_2} + \cdots r^n_{1f_k}$$

Example



$$
\begin{aligned}
r^0_{11} &= 1+\epsilon & \qquad r^0_{22} &= 1+\epsilon \\
r^0_{12} &= 0 & r^0_{21} &= 0 \\
r^1_{12} &= r^0_{12} + r^0_{11}(r^0_{11})^* r^0_{12} & r^1_{22} &= r^0_{22} + r^0_{21}(r^0_{11})^* r^0_{12} \\
&= 0 + (1+\epsilon)^+ 0 & &= (1+\epsilon) + 0(1+\epsilon)^* 0
\end{aligned}
$$

$$
\begin{aligned}
r^2_{12} &= r^1_{12} + r^1_{12}(r^1_{22})^* r^1_{22} \\
&= (0+(1+\epsilon)^+0) + (0+(1+\epsilon)^+0)((1+\epsilon)+0(1+\epsilon)^*0)^+ \\
&= (0+(1+\epsilon)^+0)(1+\epsilon+0(1+\epsilon)^*0)^* \\
&= (1+\epsilon)^*0(1+\epsilon+01^*0)^* \\
&= 1^*0(1+01^*0)^* \\
L(M) &= L(r^2_{12})
\end{aligned}
$$

Analysis of the Translation

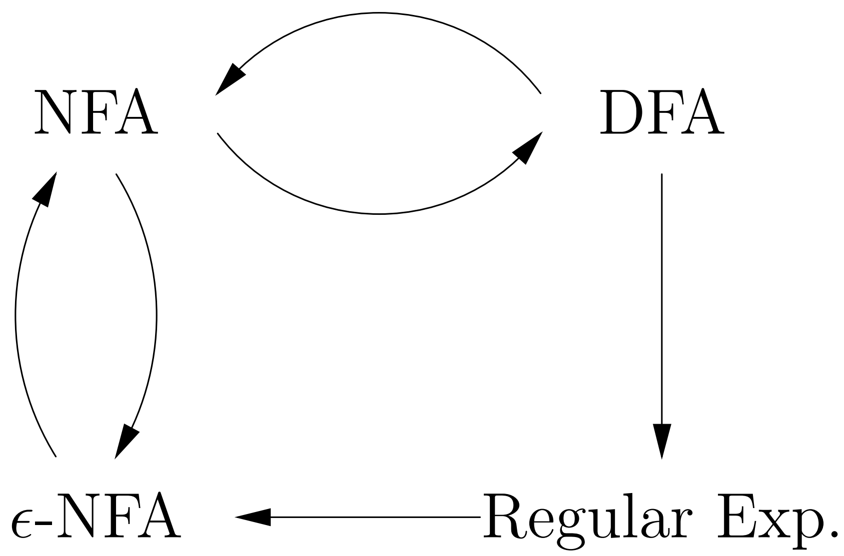
Size of the constructed regular expression

- Number of regular expressions $= O(n^3)$
- At each step the regular expression may blowup by a factor of 4
- Each regular expression r_{ij}^n can be of size $O(4^n)$

The above method works for both NFA and DFA

For converting DFA there is slightly more efficient method (see textbook)

Thus far . . .



Regular Expression Identities

Associativity and Commutativity

$$L + M = M + L$$

$$(L + M) + N = L + (M + N)$$

$$(LM)N = L(MN)$$

Note: $LM \neq ML$

Distributivity

$$L(M + N) = LM + LN$$

$$(M + N)L = ML + NL$$

More Identities

Identities and Anhilators

$$\emptyset + L = L + \emptyset = L$$
$$\epsilon L = L\epsilon = L$$
$$\emptyset L = L\emptyset = \emptyset$$

Idempotent Law

$$L + L = L$$

Closure Laws

$$(L^*)^* = L^*$$
$$(\emptyset)^* = \epsilon$$
$$\epsilon^* = \epsilon$$
$$L^+ = LL^* = L^*L$$
$$L^* = L^+ + \epsilon$$

Testing Regular Expression Identities

To test if an equation $E = F$ holds [Example: $(L+M)^* = (L^*M^*)^*$]

1. Convert E and F into concrete expressions C and D, by replacing each language variable in E and F by a concrete symbol
   [Replacing L by a and M by b, we get $C = (a+b)^*$ and $D = (a^*b^*)^*$]

2. $L(C) = L(D)$ iff the equation $E = F$ holds
   [$L((a+b)^*) = L((a^*b^*)^*)$ and so $(L+M)^* = (L^*M^*)^*$ holds]

Correctness of algorithm follows from Theorems 3.13 and 3.14 in the book

However, caution!!

The algorithm only applies to equations of regular expressions and not to all equations

- Consider the equation $L \cap M \cap N = L \cap M$, where $\cap$ means set intersection
- The equation is clearly false
- Concretizing we get $\{a\} \cap \{b\} \cap \{c\}$ and $\{a\} \cap \{b\}$, which are clearly equal!!