COL352 Lecture 8

Contents

1 Recap

Definitions from last class. See below.

2 Definitions

Definition 1

Let $L \in \Sigma^*$ be any language. We define the language L^* as follows:

$$L^* = L_0 \cup L_1 \cup \cdots = \bigcup_{n=0}^{\infty} L^n$$

where $L^0 = \{\epsilon\}, L_1 = L, L^n = L \cdot L \cdots L$ where there are n instances of L.

Definition 2

Let Σ be a finite alphabet. A regular expression over Σ is any expression that is in one of the following forms:

1. $\emptyset$.
2. ϵ.
3. a, where $a \in \Sigma$.
4. $(R_1 \cup R_2)$ where R_1, R_2 are regular expressions over Σ.
5. $(R_1 R_2)$ or $(R_1 \cdot R_2)$ where R_1, R_2 are regular expressions over Σ.
6. (R^*) where R is a regular expression over Σ.

Definition 3

The language of a regular expression R, denoted by $\mathcal{L}(R)$ is defined as follows:

1. $\mathcal{L}(\emptyset) = \emptyset$.
2. $\mathcal{L}(\epsilon) = \{\epsilon\}$.
3. $\mathcal{L}(a) = \{a\}$.
4. $\mathcal{L}((R_1 \cup R_2)) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$.
5. $\mathcal{L}((R_1 R_2)) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$.
6. $\mathcal{L}\left((R^*)\right) = \left(\mathcal{L}(R)\right)^*$

3 Content

Theorem 1

Suppose $L \subseteq \Sigma^*$ is the language of some regular expression over Σ. Then L is a regular language.

Note 1

Sketch of proof by structural induction

Obvious for cases 1, 2, 3 in the definition of regular expressions. For 4, 5, assume true for R_1, R_2, prove the claim for $R_1 \cup R_2$ and $R_1 \cdot R_2$. For 6, assume true for R, prove for $(R)^*$.

Sketch of proof by normal induction

Associate a size with every regular expression. Base case: Prove the claim for regular expressions of size 0. Inductive case: Assume claim is true for all regular expressions of size $< n$, and prove the claim for regular expressions of size $= n$.

Proof. Define *size* of a regular expression as follows: $size(\emptyset) = size(\epsilon) = size(a) = 0 \forall a \in \Sigma$, and $size((R_1 \cup R_2)) = size((R_1 \cdot R_2)) = size(R_1) + size(R_2) + 1$, and $size((R^*)) = size(R) + 1$. Then induct on *size*. ☐

Question 1

Is the converse true?

Answer. Yes, the converse is true.

Theorem 2

Let $L \subseteq \Sigma^*$ be a regular language. Then there exists a regular expression R over Σ such that $L = \mathcal{L}(R)$.

Note 2

Sketch of proof:

1. Suppose $D = (Q, \Sigma, \delta, q_0, A)$ recognizes L_0. Let $m = |Q|$. Wlog, assume $Q = \{1, \ldots, m\}$.
2. For now, ignore q_0, A, and focus on δ.
3. For $i, j \in \{1, \ldots, m\}$ and $k \in \{0, \ldots, m\}$, define the language $L_{i,j,k} \subseteq \Sigma^*$ as follows:

   $$L_{ijk} = \{x \mid \text{ run of } D \text{ on } x \text{ starts, ends at } i, j \text{ respectively, and each intermediate state is} \leq k\}$$

   It is allowed to have $i, j > k$. Note that this is a regular language for all i, j, k (for states $> k$, make all outgoing edges end on that state, set starting state to i, and accepting states $= \{j\}$).
4. Then we shall design a regular expression R_{ijk} for each L_{ijk}.
5. From $\{R_{ijk} \mid i, j \in \{1, \ldots, m\} \wedge k \in \{0, \ldots, m\}\}$, construct a regular expression for L.
   We can show that $L = \bigcup_{a \in A} L_{q_0 am}$ (exercise).
   Then we have $R = \left(\bigcup_{a \in A} R_{q_0 am}\right), L = \mathcal{L}(R)$.

It suffices to do step 4, since the rest has been done above already.
Recall the Floyd-Warshall algorithm for finding shortest walks between every pair of vertices. Define

$$D[i, j, k] = \text{length of the shortest } i \to j \text{ walk which passes through } \{1 \ldots k\}.$$

Then we have the following recurrence:

$$D[i, j, k] = \min(D[i, j, k-1], D[i, k, k-1] + D[k, j, k-1])$$

Here we break the analysis into two cases: the shortest walk either passes through k or doesn't pass through k.

function FLOYD-WARSHALL($G[m \times m]$) $\triangleright$ $G[i,j]$ contains the weight of an edge (i,j) if it exists, and if it doesn't, it is ∞ except when $i = j$, where $G[i,i] = 0$
    $D := G$
    for $k = 1 \dots m$ **do**
        for $i = 1 \dots m$ **do**
            for $j = 1 \dots m$ **do**
                $D[i,j,k] := \min(D[i,j,k-1], D[i,k,k-1] + D[k,j,k-1])$
            end for
        end for
    end for
end function

We can construct R_{ijk} from $R_{ij(k-1)}$, $R_{ik(k-1)}$, $R_{kj(k-1)}$, $R_{kk(k-1)}$ as follows, in a similar fashion.

Proof.
Let $D = (Q, \Sigma, \delta, q_0, A)$ be a DFA recognizing L. We shall use this DFA throughout the proof.

We start off with some definitions.

For $i, j \in \{1, \dots, m\}$ and $k \in \{0, \dots, m\}$, define the language $L_{i,j,k} \subseteq \Sigma^*$ as follows:

$L_{ijk} = \{x \mid$ run of $(Q, \Sigma, \delta, i, \{j\})$ on x starts, ends at i, j respectively, and each intermediate state is $\leq k\}$

Note that it is allowed to have $i, j > k$.

Claim 2.1

L_{ijk} is a regular language.

Proof. Consider the DFA $D_{ijk} = (Q \cup \{s_e\}, \Sigma, \delta', i, \{j\})$, where δ' is defined as:

$$\delta'(s,a) = \begin{cases} \delta(s,a) & \text{if } s = i \vee s \leq k \\ s_e & \text{otherwise} \end{cases}$$

We shall call this DFA <u>the</u> DFA corresponding to L_{ijk}.

Now we show the following two subclaims:

Claim 2.2

$L_{ijk} \subseteq \mathcal{L}(D_{ijk})$

Proof. Consider any string s in L_{ijk}. Then the result directly follows, since all transitions on the states involved apart from the last state in the run agree with the transitions on the run on D_{ijk}, because of the definition of δ'. $\square$

Claim 2.3

$\mathcal{L}(D_{ijk}) \subseteq L_{ijk}$

Proof. Consider any string accepted by D_{ijk}. Suppose it has an intermediate state $> k$. Then the next state will be s_e, and since $\delta(s_e, a) = s_e$ for all a, the last state of the run will be s_e, which is not an accepting state of D_{ijk}. This implies that the assumption is false, and thus the string is in L_{ijk}, whence we are done. $\square$

Using these two claims, it follows that $L_{ijk} = \mathcal{L}(D_{ijk})$, and thus L is regular by definition of a regular language. □

Note 3

In the case $k = m$, s_e is an isolated state and δ' is semantically the extension of δ on the extra state s_e.

Consider the following algorithm that takes as input a DFA D and returns a regular expression.

function GenerateRegularExpression($D = (Q, \Sigma, \delta, q_0, A)$)
  let $R[m \times m \times (m+1)]$ be a table initialized by $\emptyset$.
  for $i = 1 \ldots m$ **do**
    for $j = 1 \ldots m$ **do**
      if $i = j$ **then**
        $$R[i,i,0] := \left(\epsilon \cup \bigcup_{\delta(i,a)=i} a\right)$$ ▷ Use `foldl` to formalize this and fix parentheses
      else
        $$R[i,j,0] := \left(\bigcup_{\delta(i,a)=j} a\right)$$
      end if
    end for
  end for
  for $k = 1 \ldots m$ **do**
    for $i = 1 \ldots m$ **do**
      for $j = 1 \ldots m$ **do**
        $R[i,j,k] := (R[i,j,k-1] \cup (R[i,k,k-1] \cdot (R[k,k,k-1]^*) \cdot R[k,j,k-1]))$
      end for
    end for
  end for
  return $\left(\bigcup_{a \in A} R[q_0, a, m]\right)$
end function

Now we show the following claim:

Claim 2.4

$L_{ijk} = \mathcal{L}(R[i,j,k])$

Proof.
The proof shall proceed via induction on k.

1. Base case: $k = 0$. In this case, the run consists of at most 2 states. In the case when $i = j$, there can be runs with one state and two states, and they are precisely those which correspond to strings in $\mathcal{L}(R[i,i,0])$ by the definition of the language of a regular expression. The case for $i \neq j$ is similar, except that there need to be at least two states, and hence all possible strings in L_{ij0} have length 1, and correspond to precisely the strings in $\mathcal{L}(R[i,j,0])$.

2. Inductive step: Suppose $k > 0$.

   Fix i, j. We have $\mathcal{L}(R[i,j,k]) = \mathcal{L}(R[i,j,k-1]) \cup (\mathcal{L}(R[i,k,k-1]) \cdot \mathcal{L}(R[k,k,k-1])^* \cdot \mathcal{L}(R[k,j,k-1]))$, which, by the inductive hypothesis, gives us

   $$\mathcal{L}(R[i,j,k]) = L_{ij(k-1)} \cup (L_{ik(k-1)} \cdot L^*_{kk(k-1)} \cdot L_{kj(k-1)})$$

   Then we claim the following:

Claim 2.5

$L_{ijk} \subseteq \mathcal{L}(R[i,j,k])$

Proof. Consider any string $x = x[1]x[2]\dots x[n]$ in L_{ijk}. Then any state in the run of the corresponding DFA on x is at most k.

If the state k is never reached in the run, then the string is in $L_{ij(k-1)}$, which is a subset of $\mathcal{L}(R[i,j,k])$ due to the expression above.

Suppose the state k is reached. Suppose $e = ix[1]i_1x[2]\dots i_{n-1}x[n]j$ is the run corresponding to x, and let $j_1 < \dots < j_r$ be such that $i_w = k \iff w \in \{j_1,\dots,j_r\}$.

Then consider the runs

$$\begin{aligned} ix[1]i_1\dots x[j_1]i_{j_1}, \\ i_{j_1}x[j_1+1]\dots i_{j_2}, \\ \vdots \\ i_{j_{r-1}}x[j_{r-1}+1]\dots i_{j_r}, \\ i_{j_r}x[j_r+1]\dots j \end{aligned}$$

The first and the last correspond to runs of the DFAs corresponding to the languages $L_{ik(k-1)}$ and $L_{kj(k-1)}$, and all intermediate runs correspond to the runs of the DFAs corresponding to the language $L_{kk(k-1)}$.

Hence, the string x can be decomposed into $r+2$ possibly empty substrings $x_1 \cdot x_2 \cdots x_{r+2}$, such that $x_1 \in L_{ik(k-1)}, x_{r+2} \in L_{kj(k-1)}$ and $x_w \in L_{kk(k-1)}$, where $2 \le w \le r+1$.

Hence we have $x_2 \cdots x_w \in L^*_{kk(k-1)}$. So we have $x = x_1 \cdot (x_2 \dots x_{r+1}) \cdot x_{r+2} \in L_{ik(k-1)} \cdot L^*_{kk(k-1)} \cdot L_{kj(k-1)} \subseteq \mathcal{L}(R[i,j,k])$.

Since x was arbitrary, we have shown $L_{ijk} \subseteq \mathcal{L}(R[i,j,k])$, as needed. □

Claim 2.6

$L_{ijk} \supseteq \mathcal{L}(R[i,j,k])$

Proof. Consider any string x in $\mathcal{L}(R[i,j,k]) = L_{ij(k-1)} \cup (L_{ik(k-1)} \cdot L^*_{kk(k-1)} \cdot L_{kj(k-1)})$.

Either it is in $L_{ij(k-1)}$, in which case it is already in L_{ijk} since all intermediate states are $\le k-1 < k$, or it is in $L_{ik(k-1)} \cdot L^*_{kk(k-1)} \cdot L_{kj(k-1)}$.

In the second case, it is the result of concatenation of a string in $L_{ik(k-1)}$, some strings in $L_{kk(k-1)}$ and a final string in $L_{kj(k-1)}$. Suppose there are exactly r strings from $L_{kk(k-1)}$.

Consider the runs of each of these strings:

For the first string, let the run be $ix[1]s_{11}x[2]s_{12}\dots x[l_1]s_{1l_1} = k$.

For the d^{th} string $(2 \le d \le r+1)$, let the run be $kx[l_{d-1}+1]s_{d1}\dots x[l_d]s_{dl_d} = k$.

For the last string, let the run be $kx[l_{k+1}+1]s_{(r+2)1}\dots x[l_{k+2}]s_{(r+2)l_{r+2}} = j$.

Consider the following sequence:

$ix[1]s_{11}\dots x[l_1]kx[l_1+1]\dots x[l_2]k\dots kx[l_{k+1}+1]s_{(r+2)1}\dots x[l_{k+2}]j.$

It suffices to show that this is a (in fact *the*) valid run for the DFA corresponding to the language L_{ijk} (if it is a valid run, it must be accepting since it ends at j).

Consider any intermediate state in this run; it suffices to show that all such intermediate states are $\leq k$ (since all transitions are valid due to them being accepting runs, and all states are in the same state-space). Note that the states in this sequence are of the following kinds:

(a) Intermediate states in the runs of substrings mentioned above. In this case, it is trivially true that the states are $\leq k-1$.

(b) Start states of runs corresponding to the strings 2 through $k+2$. In this case, it's true since all such states are k.

(c) End states of runs corresponding to the strings 1 through $k+1$. In this state, it's true since all such states are k.

From here, we get the fact that x is accepted by the DFA corresponding to L_{ijk}, so $x \in L_{ijk}$. Since x was arbitrary, we get the fact that $\mathcal{L}(R[i,j,k]) \subseteq L_{ijk}$, as needed. □

From these two claims, it follows that $L_{ijk} = \mathcal{L}(R[i,j,k])$ for these particular values of i, j. Since i, j were arbitrary, this completes the inductive step.

Thus, we are done by induction on k. □

We shall now claim the following:

Claim 2.7

$$L = \bigcup_{a \in A} L_{q_0 a m}$$

Proof. This is fairly obvious; for showing that $L \subseteq \bigcup_{a \in A} L_{q_0 a m}$, consider any string in L, and consider the run corresponding to it. The run ends at an accepting state (say a), and x is in $L_{q_0 a m}$ by the definition of L_{ijk}, since all states are $\leq m$ and the start and end states are q_0, a respectively.
For showing the other direction, consider any string x in the RHS. Then there exists an $a \in A$ such that there is an accepting run of the DFA corresponding to $L_{q_0 a m}$ on x. It is a valid and accepting run of the DFA of L on x as well, since the run starts on q_0, ends at a (an accepting state), and all transitions are the same in L_{ijm} by the construction of the DFA corresponding to it. □

Now moving on to the main proof, note that

$$\begin{aligned} L &= \bigcup_{a \in A} L_{q_0 a m} \\ &= \bigcup_{a \in A} \mathcal{L}(R[q_0, a, m]) \\ &= \mathcal{L}\left(\left(\bigcup_{a \in A} R[q_0, a, m]\right)\right) \end{aligned}$$

From here, we get that $\left(\bigcup_{a \in A} R[q_0, a, m]\right)$ is a regular expression corresponding to L, whence we are done.

□