Formal Language Theory

Week 5 Notes

Lecture 7: Feb 24

There is non-determinism in how the string \( uv \) might be broken.

\[ A \cdot B = \{ x \cdot y : x \in A \land y \in B \} \]

\[ A \cdot \beta = \{ w : w \text{ can be broken as } w = x \cdot y \text{ such that } x \in A \land y \in B \} \]

\[ A^2 = A \cdot A \]

\[ A^* = A^0 \cup A^1 \cup A^2 \cup \ldots = \bigcup_{i=0}^{\infty} A^i \]

Rule: For any language \( A \), even \( A = \emptyset \), \( A^0 = \emptyset \). Like \( a^0 = 1 \).

Rule: For all languages \( A \), \( A \cdot \emptyset = \emptyset \) and \( A \cdot \emptyset^* = A \) (chom)

Distributive Law: \( A \cdot (B \cup C) = A \cdot B \cup A \cdot C \) and \( (A \cdot B) \cdot C = A \cdot (B \cdot C) \)

Main difference: \( \emptyset + \emptyset = \emptyset \) \( \emptyset \cdot A = \emptyset \) and \( \emptyset \cdot A = A \)

For all languages \( A, B, \) and \( C \):

\[ A \cdot (B \cup C) = A \cdot B \cup A \cdot C \]

\[ (A \cdot B) \cdot C = A \cdot (B \cdot C) \]

\[ \alpha \cdot (\beta + \gamma) = \alpha \beta + \alpha \gamma \quad (\alpha + \beta) \cdot \gamma = \alpha \cdot \gamma + \beta \cdot \gamma \]

Example of a totally new kind of rule: \( (\epsilon + r)^* = r^* (\epsilon + v^* \cup A)^* = A^* \)

Example:

\[ L_{11} = \{ x \in \{0, 1\}^* : \#0(x) \text{ is even} \} \]

\[ L_{12} = \{ x \in \{0, 1\}^* : \#0(x) \text{ is odd} \} \]

\[ L_{11} = (1 + 01)^* \]

\[ L_{12} = 01^* (011)^* \]

Rule: For any state \( \rho \), \( L_{\rho p} = r^* \) for some regular \( r \).

The Abstract 2-State GNFA

\[ L_{11} = (r + tu^*v)^* \]

\[ L_{12} = L_{11}tu^* = (r + tu^*v)^*tu^* \]

\[ L_{21} = (u + v^*r)^* \]

\[ L_{22} = (u + v^*t)^* \]
**Defn:** A generalized NFA (GNFA) is a 5-tuple \(N = (Q, \Sigma, S, s, F)\) which is like an NFA except \(S \subseteq Q \times \text{Regexp}(\Sigma) \times Q\). (i.e., \(\delta\) is a regexp.

**Defn:** A computation path from a state \(p\) to a state \(q\) is a sequence \((p, x_1, q_1, x_2, q_2, \ldots, q_{m-1}, x_m, q_m = q)\) such that for all \(i, 1 \leq i \leq m, (q_{i-1}, x_i, q_i)\) is an instruction in \(S\).

If \(N\) can process a string \(w \in \Sigma^*\) from \(p\) to \(q\) if \(w\) can be broken into \(m\) substrings \(W = u_1 \cdot u_2 \cdot u_3 \cdots u_m\) such that for each \(i, 1 \leq i \leq m, u_i \in L(x_i)\) \(\text{and } u_i\) matches \(x_i\).

Finally, \(L(p, q) = \{w : N\text{ can process }w \text{ from }p\text{ to }q\} \quad \text{and} \quad u_i \in L(x_i)\) \(\text{and } u_i\) matches \(x_i\).

**Example:** In this new form:

\[
\begin{align*}
0+1 & \xrightarrow{0} 1 \\
& \xrightarrow{0+1} 2 \\
& \xrightarrow{0+1} 3
\end{align*}
\]

To witness this defn, if \(I\) my path is \((1, 1, 2, (0+1)^2, 3)\) then \(I\) fail.

\(W = u_1 \cdot u_2\) with \(u_1 = 1\) and \(u_2 = 101 \in L((0+1)^2)\).

Good path: \((1, 0+1, 1, 1, 2, (0+1)^2, 3)\) matched with \(W = 1 \cdot 1 \cdot 01\) \(u_1 = 1 \quad u_2 = 1 \quad u_3 = 01\)

**Theorem:** Given any DFA or NFA or GNFA \(N\), we can compute a regexp \(\lambda\) such that \(L(N) = L(\lambda)\).

**Proof:** We have already proved this for \(K \leq 2\), where \(N = (Q, \Sigma, S, s, F)\) and \(K = 1\).

Given \(N\) with \(K \geq 2\), if \(N\) has more than one accepting state \(q \neq s\), then add a new accepting state \(f\) and \(E\)-ars from all \(q \in F\) to \(f\).
Hence we may suppose \( N \) is in well-structured form. \((K \text{ became } K+1)\)

Number \( S = 1 \), and the final states \( 2, 3, 4, 5 \), or just \( 1 \) if \( F = \emptyset \)?

Thus states \( 3, \ldots, K \) are non-accepting. We will eliminate them by introducing a new state \( R \),

\[ q = 0 \times K \]

which we write \( N \) give

\[ T_{pq} = T_{pq} \]

\[ q = 0 \times K \]

\[ T_{pr} = W \]

To eliminate the highest-numbered state \( q \), we bypass all arcs \((p, q, g)\) into \( q \).

To bypass \((p, q, g)\), for all outgoing arcs \((q, v, r)\), we update the direct path \( T_{pr} \) by

\[ T_{pr} = T_{pr} + t^{uv} \]

Then we can delete \((p, q, g)\).

for \( \text{int} j = \#; j \geq 3; j-- \) \{ \}

// eliminate state \( \# \) i.e. state \( \#$

for \( \text{int} i = 1; i < j-1; i++ \) \{

\[ T_{ij}, h += \text{old} \]

(\text{that is, new} \ T_{ij}, h = \text{old} \ T_{ij}, h + \text{old} \ T_{ij}, h \)

\[ T_{ij}, h += \text{old} \]

(\text{old} \ T_{ij}, h = \text{old} \ T_{ij}, h

\[ T_{ij}, h += \]
Example (text w/o adding new start state, still renumber by new acc state §)

Elim state 4: Inference: Just \((1, b, 4)\)

Outgoing: \((4, b, 1)\): Update \((4, 1) (1, -1, 1)\)

\[ T(1, 4) = \emptyset, \quad T(4, 1) = \emptyset \]
\[ T(1, 4)^* = \emptyset, \quad T(4, 1)^* = \emptyset \]

New \(T(1, 4) = \text{old } T(1, 1) \cup T(1, 4) T(4, 4)^* T(4, 1)\).

You can also initialize \(\emptyset \cup 0 \cdot 3 \cdot b = bb\)

\(T(1, 1) \in \emptyset\), likewise \(\emptyset \cup b \cdot 3 \cdot b = 3 \cup bb\)

\(T(1, 2) \in \emptyset\) for all \(2\).

\[ \triangle \text{ Both answers will work because the ultimate expression will have } T(1, 4)^* \text{ and } (bb)^* = (b + bb)^* \]

\[ \text{Q-pass } (1, b, 4) \text{ to } 3 \]

New \(T(1, 3) = \text{old } T(1, 1) \cup T(1, 4) T(4, 4)^* (T(4, 3) \cup b \cdot 3 \cdot a = a + ba\)

New \(T(1, 2) = \text{old } T(1, 1) \cup T(1, 4) T(4, 4)^* T(4, 2)\).

We cannot replace \(\emptyset \cup \) by \(\emptyset\) because dest state \(2 \neq \) origin \(1\).

Elim 3:

Inference: On 14 \((1, 3)\)

Outgoing: \((3, 1)\): Update \(T(1, 1)\) again

\(T(1, 3) T(3, 3)^* T(3, 1)\).

\(T(1, 1) T(1, 3)^* T(3, 1)\).

\(L_{11} = T_{11} = (b + (atba) b)^* (b + (atba) b)^*\)

\(L_{11} = T_{11} = (bb + (atba) b a)^* (b + (atba) b)^*\)

\(L(M) = L_{11} \cdot T_{12} = (bb + (atba) b a)^* (b + (atba) b)\)
Theorem: For any language \( L \subseteq \Sigma^* \) (any \( \Sigma \)):

\[
\begin{align*}
&\text{(a) There is a DFA } M \text{ s.t. } L = L(M) \\
&\text{(b) There is an NFA } N \text{ s.t. } L = L(N) \\
&\text{(c) There is a regular expression } r \text{ s.t. } L = L(r)
\end{align*}
\]

\( L \) is a regular language if any holds.

When is a language \( L \) (non-) regular?

Related: Why is a regular \( L \) regular?

Let's reuse the original DFA

\[
M = \begin{align*}
\begin{array}{c}
\text{1} \\
\text{2} \\
\text{3} \\
\end{array}
\end{align*}
\]

\[
\begin{aligned}
\text{1} & \xrightarrow{a} \text{2} \\
\text{2} & \xrightarrow{b} \text{3} \\
\text{3} & \xrightarrow{a} \text{1}
\end{aligned}
\]

\( \alpha \text{ and } \beta \text{ both get processed from } \text{1} \text{ to } \text{2}. \)

Let \( L = L(M) \)

\( \alpha \text{ and } \beta \text{ get processed to the same state.} \)

\( \begin{align*}
\forall \alpha, \beta \in \Sigma^* : \alpha \beta \in L(M) & \iff \beta \alpha \in L(M)
\end{align*} \)

\( \forall \epsilon \in \Sigma^* : \epsilon \in L(M) \iff \epsilon \in L(M). \)

\( \forall \alpha \in \Sigma^* : \alpha \in L(M) \iff \epsilon \in L(M). \)

\( \forall x \in \Sigma^* : x \in L(M), \text{ or if } x = 0 \text{, they're both in } L(M). \)

If \( z = a \), then both cannot: \( \alpha a, \beta a \notin L(M) \).

Hence, \( M \) must have at least 2 states.

Suppose \( S \) is a set of strings such that for any distinct \( x, y \in S \): \( x \neq y \).

Then for any \( x, y \in S \) (\( x \neq y \)), any DFA \( M \) must process \( x \) to \( y \).

\( \forall \alpha, \beta \in \Sigma^* : L(x \beta) \neq L(y \beta). \)

\( \forall \epsilon \in \Sigma^* : L(\epsilon) \neq L(\epsilon). \)

\( \forall \alpha \in \Sigma^* : L(\epsilon) \neq L(\epsilon). \)

\( \forall x \in \Sigma^* : L(x) \neq L(\epsilon). \)

If \( S \) has \( K \) strings, then any DFA \( M \) must have at least \( K \) states.

If \( S \) has \( \infty \) strings, then any DFA \( M \) must have at least \( \infty \) states.

Hence, \( L \) has no finite DFA, so \( L \) is not regular.
Theorem: Suppose $L$ is a language and $S$ is a set of strings s.t. $S$ is infinite, and the Myhill-Nerode Thm holds. For all $x, y \in S$, $(x \neq y) \land (\exists z \in \Sigma^*) \land L(xz) \neq L(yz)$.

Then $L$ is not regular.

Example: $L = \{a^n b^n : n \geq 0\}$. To prove $L$ is not regular:

Take $S = \{a^n : n \geq 0\}$. Clearly $S$ is infinite.
Let any $x, y \in S$, $x \neq y$, be given. Then there are numbers $m, n \geq 0$, so $x = a^m$ and $y = a^n$.

Take $z = b^m$.

Then $xz = a^m b^m \in L$, but $yz = a^n b^m \notin L$ since $m \neq n$.

$i.e.$, $L(xz) \neq L(yz)$, and since $x, y \in S$ are arbitrary,$S$ is an infinite PC set for $L$, $\therefore L$ is not regular.

Extra: The other half of the theorem: If $L$ is not regular, then there is an infinite set $S$ such that $(\forall x, y \in S, x \neq y) \land (\exists z \in \Sigma^*) \land L(xz) \neq L(yz)$, i.e. such that $S$ is PC for $L$.

This means that in principle every non-regular language has some kind of Myhill-Nerode proof.

The contrapositive of this is: If every PC set $S$ for $L$ is finite, then $L$ is regular.

We can prove this - which completes both halves of the theorem - without going into quite as much detail as the text in problem 1.52. Consider the following unbounded "allocation" process:

For string $x = \epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$ in order, (having initialized $S = \emptyset$) {

If $(\exists y < x) \land (\forall z \in \Sigma^*) \land L(yz) = L(xz)$ then define $\text{State}(x) = \text{State}(y)$.

Else allocate $\text{State}(x)$ as a new state and put $S := S \cup \{x\}$. // Invariant: $S$ is PC for $L$.

If $x = 1^n$ then for all $w \in \Sigma^*$ of length $n-1$ and $c \in \Sigma$, define $\Delta(x, c) = \text{State}(w, c)$.

By hypothesis, $S$ stays finite, and so $(S, \Sigma, \Delta, \text{State}(c), F)$ is a DFA M s.t. $L(M) = L$, where $F = \{\text{State}(x) : x \in L\}$.\[\]