§ 1.3 - Regular Expression \( \rightarrow \) Regular Languages

\( \text{(give the algorithm)} \)

\( = \text{State Theorem: } \) Regular Language \( \rightarrow \) Regular Expr.

\( \text{(give an example but skip the algorithm)} \)

§ 1.4 - Instead of "pumping lemma"

we will use the "Myhill-Nerode Theorem":

\( \text{avoid} \)

Also \( \rightarrow \) to prove some languages are not regular

\( \text{to find minimum # states in a DFA} \)

for some languages

For Thursday, we'll have student selected breakout rooms; you'll need Zoom \( \geq 5.3.0 \)
Regular Expression: If $\Sigma$ is an alphabet, a regular expression over $\Sigma$:

1. If $T \in \Sigma$, "$T$" is a regular expression
2. "$\varepsilon$" is a regular expression
3. $\emptyset$ (empty set)

If $R_1, R_2$ regular expressions, then

4. $(R_1 \cup R_2)$
5. $(R_1 \cdot R_2)$
6. $(R_1^*)$

are also regular expressions

[Sig] gives a bunch of example

Say $\Sigma = \{a, b\}$, then

$\Sigma^* aa \Sigma^* = (a ub)^* aa (a ub)^*$

$= (a ub)^* \cdot a \cdot a \cdot (a ub)^*$

could be $((a ub)^* \cdot a) \cdot (a \cdot (a ub)^*)$
describes \( \{a, b\}^* \circ \{a\} \circ \{a\} \circ \{a, b\}^* \) notation in §1.1

or \( \Sigma^* a \Sigma^* \) in shorthand

1. \( \Sigma^* a \Sigma^* = \{ \omega \in \Sigma^* \mid \omega \) has aa as a substring\}

2. \((\Sigma^* \cup a)(bb) = \{ \Sigma^* bb \} \cup \{abb\} = \Sigma^* bb

3. \((abbb) \circ (aa) = \{ aaaa, bbaaa \}

or \((abbb) aa

Is ab \in (a\cup b)^*

Is ab \in \) language that \((a\cup b)^*\) describes

\((a\cup b)^* \) as set \( \{a, b\}^* = \) \{all strings over \{a, b\}\}

4. \( a^* \cup b^* \) as a set \( \{a^*\} \cup \{b^*\}

\( = \{\epsilon, a, aa, a^3, c^4\} \cup \{c, b, b^2, b^3, \ldots\} \)

\( = \{w \mid w \) consists only of a's \}

5. \((a^5)^* \circ (a^7)^* = \{ a^{5m+7n} \mid m, n = 0, 1, 2, \ldots\} \)

build NFA
Thm: Any regular expression describes a regular language.

pf: ① $\{\epsilon\}$ ② $\{\exists\}$ ③ $\{\} = \emptyset$ are regular languages; if $R_1, R_2$ are regular, then so are $R_1 \cup R_2$, $R_1 \cap R_2$, and $R_1^*$.

NFA's help us

In more detail  $\{\epsilon\} \xrightarrow{\delta} \emptyset \xrightarrow{\delta} \emptyset \xrightarrow{\delta} \{\exists\}

$\subseteq \{\epsilon\}$

= things in $\subseteq$ not in $\{\epsilon\}$

We'll just assume this w/o proof:

Thm: Any regular language is described by a regular expression.

Example: $L = \{0, 3, 6, 9, 12, \ldots\} < \{0, 1, \ldots, 9\}^*$
In Section §1.3, there is an algorithm

\[
\text{DFA} \rightarrow \text{regular expression}
\]

The regular expression for \( \{0, 3, 6, 9, 12, 15, \ldots \} \)

is quite long...
§ 1.4 Non-regular languages.

Replace "pumping lemma" with Myhill-Nerode theorem.

Idea:

DFA

\[ \begin{array}{c}
\text{q}_0 \\
\text{q}_1 \\
\text{q}_2 \\
\text{q}_3 \\
\end{array} \]

\[ \begin{array}{cccc}
a & b & \rightarrow & b \\
\downarrow & \downarrow & \uparrow & \uparrow \\
\text{q}_0 & \text{q}_1 & \text{q}_2 & \text{q}_3 \\
\end{array} \]

\[ L \]

Fundamental Observation:

\( \forall a \), get to \( q_1 \):

\( ab \) \( \ni q_1 \)

\( aab \) \( \ni q_1 \)

\[ \{ w \in \{a, b\}^* \mid aw \in L \} = \{ w \in \{a, b\}^* \mid abw \in L \} \]

\( \text{i.e.} \)

\( aw \in L \Rightarrow abw \in L \)
Given $\Sigma$ alphabet, $L \subseteq \Sigma^*$ write for $u \in \Sigma^*$

Accepting Futures $L(u) = \{ w \mid u w \in L \}$

E.g. $L = \{ \omega \in \{a,b\}^* \mid \omega \text{ has exactly } 2 \text{ a's} \}$

Accept $L(aaa) = \{ \omega \mid aaa \omega \in L \}$

$= \emptyset$

Accept $L(ababab) = \{ \omega \mid ababw \in L \}$

$= \{ \omega \mid \omega \text{ has exactly } b^* \text{ a's} \}$

$= \{ b, b^2, b^3, \ldots \}$

Accept $L(a) = \{ \omega \mid \omega \text{ has exactly } a \}$
\[
\text{Accfut}_L(\varepsilon) = \text{Accfut}_L(b) \\
= \text{Accfut}_L(b^2) = \{ w \text{ s.t. exactly 2 a's} \}
\]

So: \( \text{Accfut}_L(\varepsilon) = L \_ \text{2 a's} \quad (1) \)

\( \text{Accfut}_L(a) = L \_ \text{one a} \quad (2) \)

\( \text{Accfut}_L(aa) = L \_ 0 \text{ a's} = b \# \quad (3) \)

\( \text{Accfut}_L(aaa) = \emptyset \quad (4) \)

\( \Rightarrow \varepsilon, a, aa, aaa \text{ have to land in different states of any DFA for } L = L \_ 2 \text{ a's} \)

\[
\text{DFA:}
\]

\[
\begin{array}{ccccccc}
& b & & b & & a, b \\
\downarrow & & & & & \\
q_0 & \overset{a}{\rightarrow} & q_1 & \overset{a}{\rightarrow} & q_2 & \overset{a}{\rightarrow} & q_3 \\
& & & & & \uparrow & \uparrow & \uparrow
\end{array}
\]
size of \( \{ \text{AccFut}_L(w) \mid w \in \Sigma^* \} \)

= minimum number of states in DFA accepting \( L \).

Break: My watch 10:23 \( \sim \) 10:28

Myhill–Nerode Thm: For an alphabet \( \Sigma \),

\( L \subseteq \Sigma^* \)

size of \( \{ \text{AccFut}_L(w) \mid w \in \Sigma^* \} \):

- if finite, = min # states for a DFA for \( L \)
- if infinite, \( L \) is not regular
Proof: (1) \# \{ \text{AccFut}_L(w) \mid w \in \Sigma^* \} \leq k \text{ states for DFA}

(2) \# \{ \text{AccFut}_L(w) \mid w \in \Sigma^* \} \geq k \text{ states for DFA}

idea: states \leftrightarrow \uparrow

Initial state \leftrightarrow \text{AccFut}_L(\varepsilon)

Final state \leftrightarrow \text{AccFut}_L(w) \text{ that contain } \varepsilon
\[ L = \{ w \mid \omega \text{ contains } \geq 1 \text{ } a \text{ and } \omega \geq 2 \text{ } b's \} \]

\[ \text{AccFut}_L(\varepsilon) = L = \{ \omega \mid \exists \omega \in L \} \]

\[ \text{Start state} \]

\[ \text{AccFut}_L(\varepsilon) = L = \{ \omega \mid \omega \in L \} \]

\[ \text{AccFut}_L(a) = \{ \omega \mid \omega \text{ contains } \geq 1 \text{ } a \text{ and } \geq 2 \text{ } b's \} \]

\[ \text{AccFut}_L(ab) = \{ \omega \mid \omega \text{ contains } \geq 1 \text{ } a \text{ and } \geq 1 \text{ } b \} \]

\[ \text{AccFut}_L(abb) = \{ \omega \mid \omega \text{ contains } \geq 1 \text{ } a \text{ and } \geq 2 \text{ } b's \} \]

\[ \text{AccFut}_L(abb) = \{ \omega \mid \omega \text{ contains } \geq 0 \text{ } a's \text{ and } \geq 0 \text{ } b's \} \]

\[ \text{AccFut}_L(abb) = \{ \omega \mid \omega \text{ contains } \geq 0 \text{ } a's \text{ and } \geq 0 \text{ } b's \} \]

\[ \text{AccFut}_L(abb) = \{ \omega \mid \omega \text{ contains } \geq 0 \text{ } a's \text{ and } \geq 0 \text{ } b's \} \]

\[ \text{AccFut}_L(abb) = \Sigma^* = \text{AccFut}_L(abb) \]
Thm:\qquad L = \{ 0^n 1^n \mid n \in \mathbb{N} \} = \{ 01, 0011, 000111, \ldots \} \text{ is nonregular}.

Rough reason: if \( 0 \rightarrow 0 \rightarrow \ldots \) have to know see how many

Next time:

\[ \text{ActFut}_L(\emptyset) = \{ 01, 0011, 000111, \ldots \} \]
\[ \text{ActFut}_L(\emptyset) = \{ 1, 011, 00111, \ldots \} \]
\[ \text{ActFut}_L(\emptyset) = \{ 11, 01^3, 0^21^4, \ldots \} \]

Update to Zoom 5.3, something
Consider
\[ \{ \text{DFA}': (Q, \Sigma, \delta, q_0, F) \mid \text{for some } m \in \mathbb{N}, \ Q = \{1, 2, \ldots, m\} = [m] \} \]

(a)
\[ Q = \{1\}, \{1, 2\}, \{1, 2, 3\}, \ldots \]

a notion of "algorithm"