Lexical Analysis: Recap

Token Specification

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ab)</td>
<td>({\text{Action 1}})</td>
<td></td>
</tr>
<tr>
<td>(aab)</td>
<td>({\text{Action 2}})</td>
<td></td>
</tr>
<tr>
<td>(a+)</td>
<td>({\text{Action 3}})</td>
<td></td>
</tr>
</tbody>
</table>

NFA

\[
\begin{array}{c}
q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \\
q_4 \xrightarrow{\epsilon} q_5 \xrightarrow{a} q_6 \xrightarrow{b} q_7 \\
q_8 \xrightarrow{a} q_9 \\
\end{array}
\]

DFA

Example:

Input: \(aab\)

- \(s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_2\)

\(\Sigma = \{a, b\}\)
Both DFAs accept: \(((a \mid b) b \ast a)^*\)

**DFA Minimization**

- Converts a DFA to another DFA that:
  - recognizes the same language
  - has a minimum number of states
- Increases time/space efficiency
• For every regular language $L$ there exists a unique minimal DFA that recognizes $L$
  - uniqueness up to renaming of states (isomorphism)
• Minimal DFA can be found mechanically
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to $q_3$

\[ \Sigma = \{a, b\} \]
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to $q_3$

$\Sigma = \{a, b\}$
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to $q_3$
- $q_2$, $q_6$ are both accepting sinks with self-loop for any character in $\Sigma$
- Any string reaches $q_2$ or $q_6$ is guaranteed to be accepted later
- $q_2$ and $q_6$ are equivalent states: we can unify them

$\Sigma = \{a, b\}$
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to \( q_3 \)
- \( q_2, q_6 \) are both accepting sinks with self-loop for any character in \( \Sigma \)
- Any string reaches \( q_2 \) or \( q_6 \) is guaranteed to be accepted later
- \( q_2 \) and \( q_6 \) are equivalent states: we can unify them

\[
\Sigma = \{a, b\}
\]
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to \( q_3 \)
- \( q_2, q_6 \) are both accepting sinks with self-loop for any character in \( \Sigma \)
- Any string reaches \( q_2 \) or \( q_6 \) is guaranteed to be accepted later
- \( q_2 \) and \( q_6 \) are equivalent states: we can unify them

- If DFA is in \( q_1 \) or \( q_5 \):
  - if next character is \( a \), it forever accepts in both states
  - if next character is \( b \), it forever rejects in both states

- \( q_1 \) and \( q_5 \) are equivalent states: we can unify them

\[ \Sigma = \{a, b\} \]
DFA Minimization: Example

- Remove unreachable states: there is no path from initial state to $q_3$
- $q_2$, $q_6$ are both accepting sinks with self-loop for any character in $\Sigma$
- Any string reaches $q_2$ or $q_6$ is guaranteed to be accepted later
- $q_2$ and $q_6$ are equivalent states: we can unify them
- If DFA is in $q_1$ or $q_5$:
  - if next character is $a$, it forever accepts in both states
  - if next character is $b$, it forever rejects in both states
- $q_1$ and $q_5$ are equivalent states: we can unify them

\[
\begin{align*}
q_0 & \longrightarrow a, b \\
q_1 & \overset{a}{\longrightarrow} q_1, \overset{b}{\longrightarrow} q_4 \\
q_4 & \overset{a, b}{\longrightarrow} q_7 \\
q_7 & \overset{a, b}{\longrightarrow} q_7 \\
\Sigma & = \{a, b\}
\end{align*}
\]
Equivalent States

Intuition

- Two states are equivalent if all subsequent behavior from those states is the same
- Equivalent states may be unified without affecting DFA’s behavior

Definition

- We say that states $p$ and $q$ are equivalent if for all $w$:
  $\hat{\delta}(p, w)$ is an accepting state iff $\hat{\delta}(q, w)$ is an accepting state
- $\hat{\delta}$ is the transition function extended for words
DFA Minimization: Procedure

- Write down all pairs of state as a table
- Every cell in table denotes if corresponding states are equivalent
- Table is initially unmarked
- We mark pair \((p_i, p_j)\) when we discover \(p_i\) and \(p_j\) are not equivalent

\[
\begin{array}{ccccccc}
  & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\
\end{array}
\]
1. Start by marking all cells \((q_i, q_j)\) where one of them is final and other is non-final.

2. Look for unmarked pairs \((q_i, q_j)\) such that for some \(c \in \Sigma\), the pair \((\delta(q_i, c), \delta(q_j, c))\) is marked. Then mark \((q_i, q_j)\).

3. Repeat step 2 until no such unmarked pairs remain.
Illustration of minimization algorithm

First mark accepting/non-accepting pairs
(q_1, q_3) is unmarked,

q_1 \xrightarrow{b} q_0,
q_3 \xrightarrow{b} q_1,

and (q_0, q_1) is marked,
so mark (q_1, q_3)
(q₁, q₃) is unmarked,

\[ q₁ \xrightarrow{b} q₀, \]

\[ q₃ \xrightarrow{b} q₁, \]

and (q₀, q₁) is marked,

so mark (q₁, q₃)
(q₂, q₃) is unmarked,
q₂ \xrightarrow{b} q₀,
q₃ \xrightarrow{b} q₁,
and (q₀, q₁) is marked,
so mark (q₂, q₃)
Illustration of minimization algorithm

(q₂, q₃) is unmarked,
q₂ \xrightarrow{b} q₀,
q₃ \xrightarrow{b} q₁,
and (q₀, q₁) is marked,
so mark (q₂, q₃)

```
<table>
<thead>
<tr>
<th></th>
<th>q₀</th>
<th>q₁</th>
<th>q₂</th>
<th>q₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₀</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>q₁</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>q₂</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>q₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Illustration of minimization algorithm

There is no way to mark the only unmarked pair \((q_1, q_2)\)
Obtain minimized DFA by collapsing \(q_1, q_2\) to a single state
Illustration of minimization algorithm
Convert the following DFA to a DFA with 3 states