Lexical Analysis:
DFA Minimization
Automating Scanner Construction

PREVIOUSLY
RE → NFA (Thompson’s construction)
• Build an NFA for each term
• Combine them with ε-moves
NFA → DFA (subset construction)
• Build the simulation

TODAY
DFA → Minimal DFA
• Hopcroft’s algorithm
DFA Minimization

Details of the algorithm

- Group states into maximal size sets, *optimistically*
- Iteratively subdivide those sets, as needed
- States that remain grouped together are equivalent
DFA Minimization

Remember DFA = (Q, Σ, δ, q₀, F)

Initial partition, P₀, has two sets: {D₉} and {D-D₉}

Splitting a set s (“partitioning a set by a”)

• Assume qᵢ, & qⱼ ∈ s and δ(qᵢ, a) = qₓ and δ(qⱼ, a) = qᵧ
• If qₓ and qᵧ are not in the same set, then s must be split
  → qᵢ has transition on a, qⱼ does not ⇒ a splits s
• One state in the final DFA cannot have two transitions on a (otherwise we have an NFA!)
DFA Minimization (the algorithm)

\[
P \leftarrow \{ D_F, \{D-D_F\}\}
\]

while (P is still changing)

\[
T \leftarrow \emptyset
\]

for each set \( p \in P \)

\[
T \leftarrow T \cup \text{Split}(p)
\]

\[
P \leftarrow T
\]

\[
\text{Split}(S)
\]

for each \( \alpha \in \Sigma \)

if \( \alpha \) splits \( S \) into \( s_1 \) and \( s_2 \)

then return \( \{s_1, s_2\} \)

return \( S \)

This is a another
fixed-point algorithm!
Key Idea: Splitting $S$ around $\alpha$

Original set $S$

$S$ has transitions on $\alpha$ to $R$, $Q$, & $T$

The algorithm partitions $S$ around $\alpha$
Key Idea: Splitting $S$ around $\alpha$

Original set $S$

$S_2$ is everything in $S - S_1$

Could we split $S_2$ further?
Yes, will do this in another iteration!
DFA Minimization

What about $a (b \mid c)^*$?

First, the subset construction:

<table>
<thead>
<tr>
<th>NFA states</th>
<th>$\varepsilon$-closure($\Delta(s,\varepsilon)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$q_0$, $q_1, q_2, q_3, q_4, q_6, q_9$</td>
</tr>
<tr>
<td></td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$q_1, q_2, q_3, q_4, q_6, q_9$</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$q_5, q_7, q_8$</td>
</tr>
<tr>
<td></td>
<td>$q_3, q_4, q_6$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$q_7, q_8, q_9$</td>
</tr>
<tr>
<td></td>
<td>$q_3, q_4, q_6$</td>
</tr>
</tbody>
</table>

**Final states**
Apply DFA Minimization algorithm

\[
P \leftarrow \{ D_F, \{D-D_F}\}\\
\text{while (} P \text{ is still changing)}\\
\quad T \leftarrow \emptyset\\
\quad \text{for each set } p \in P\\
\quad \quad T \leftarrow T \cup \text{Split}(p)\\
\quad P \leftarrow T
\]

\text{Split}(S)\\
\quad \text{for each } \alpha \in \Sigma\\
\quad \quad \text{if } \alpha \text{ splits } S \text{ into } s_1 \text{ and } s_2\\
\quad \quad \quad \text{then return } \{s_1, s_2\}\\
\quad \text{return } S
DFA Minimization

Then, apply the minimization algorithm

<table>
<thead>
<tr>
<th>Current Partition</th>
<th>Split on</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 )</td>
<td>a</td>
</tr>
<tr>
<td>{ ( s_1 ), ( s_2 ), ( s_3 ) } ( { s_0 } )</td>
<td>none</td>
</tr>
</tbody>
</table>

In a previous lecture, we observed that a human would design a simpler automaton than Thompson’s construction & the subset construction did.

Minimizing that DFA produces the one that a human would design!
Abbreviated Register Specification

Start with a regular expression

\[ r0 \mid r1 \mid r2 \mid r3 \mid r4 \mid r5 \mid r6 \mid r7 \mid r8 \mid r9 \]

The Cycle of Constructions
Abbreviated Register Specification

Thompson’s construction produces

To make it fit, we’ve eliminated the ε-transition between “r” and “0...9”.

The Cycle of Constructions
Abbreviated Register Specification

The subset construction builds

This is a DFA, but it has a lot of states ...

The Cycle of Constructions
Abbreviated Register Specification

The DFA minimization algorithm builds

This looks like what a skilled compiler writer would do!

The Cycle of Constructions