Automaton and Regular Expressions

Deterministic Finite Automata (DFAs), Non-deterministic Finite Automata (NFAs) and REs have the same expressive power i.e. allow precisely same patterns/sets to be specified.

For every DFA there is an equivalent RE

For every NFA there is an equivalent DFA

For every RE there is an equivalent NFA
There is a (beautiful!) procedure from converting a regular expression to an NFA.
There is a (beautiful!) procedure from converting a regular expression to an NFA.

Associate each regular expression with an NFA with the following properties:
- There is exactly one accepting state.
- There are no transitions out of the accepting state.
- There are no transitions into the starting state.

These restrictions are stronger than necessary, but make the construction easier.
Base Cases

Automaton for $\varepsilon$

Automaton for single character $a$
Construction for $R_1R_2$
Construction for $R_1R_2$
Construction for $R_1 | R_2$
Construction for $R_1 | R_2$
Construction for $R_1 | R_2$
Construction for $R_1|R_2$
Construction for $R_1|R_2$
Construction for $R_1^*$
Construction for $R_1^*$
Construction for $R_1^*$
Construction for $R_1*$
Construction for $R_1*$
Overall Result

- Any regular expression of length $n$ can be converted into an NFA with $O(n)$ states.
- Can determine whether a string of length $m$ matches a regular expression of length $n$ in time $O(mn^2)$. 
Reminder: FAs in Scanners

- Want DFA for speed (no backtracking or cloning).
- But conversion from regular expressions to NFA is easier.
- Use a procedure for converting an NFA to an equivalent DFA.
Why DFA?

- In the worst-case, an NFA with \( n \) states takes time \( O(mn^2) \) to match a string of length \( m \).
- DFAs, on the other hand, take only \( O(m) \).
DFA Example

```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
```

- **States**: A, B, C, D
- **Start State**: A
- **Transitions**:
  - A -> B (0)
  - A -> C (1)
  - B -> A (1)
  - B -> D (0)
  - C -> B (1)
  - C -> C (0)
  - D -> B (0)
  - D -> D (1)
From NFA to DFA

- NFAs can be in many states at once, while DFAs can only be in a single state at a time.
- **Key idea:** Make the DFA simulate the NFA.
- Have the states of the DFA correspond to the sets of states of the NFA.
- Transitions between states of DFA correspond to transitions between sets of states in the NFA.
From NFA to DFA

- **Subset construction**: construct a DFA from the NFA, where each DFA state represents a set of NFA states.

- **Key idea**: the state of the DFA after reading some input is the set of all NFA states that could have reached after reading the same input.
Algorithm: example of a fixed-point computation.

- Find $\varepsilon$-closure (all states reachable via 0 or more $\varepsilon$-transitions) of start state. Create a DFA state corresponding to this set. Add it to the unvisited list.
**Algorithm**: example of a fixed-point computation.

- Find $\varepsilon$-closure (all states reachable via 0 or more $\varepsilon$-transitions) of start state. Create a DFA state corresponding to this set. Add it to the unvisited list.
- While there exist unvisited DFA states, select one (call it $d$):
  - For each symbol $s$ in the alphabet, determine the NFA states reachable by any NFA state in the set corresponding to $d$.
From NFA to DFA

- **Algorithm**: example of a fixed-point computation.
  - Find $\varepsilon$-closure (all states reachable via 0 or more $\varepsilon$-transitions) of start state. Create a DFA state corresponding to this set. Add it to the unvisited list.
  - While there exist unvisited DFA states, select one (call it $d$):
    - For each symbol $s$ in the alphabet, determine the NFA states reachable by any NFA state in the set corresponding to $d$.
    - Determine the $\varepsilon$-closure of these states. Create a transition from $d$ on symbol $s$ to a state corresponding to this closure set.
From NFA to DFA

**Algorithm:** example of a fixed-point computation.

- Find $\varepsilon$-closure (all states reachable via 0 or more $\varepsilon$-transitions) of start state. Create a DFA state corresponding to this set. Add it to the unvisited list.
- While there exist unvisited DFA states, select one (call it $d$):
  - For each symbol $s$ in the alphabet, determine the NFA states reachable by any NFA state in the set corresponding to $d$.
  - Determine the $\varepsilon$-closure of these states. Create a transition from $d$ on symbol $s$ to a state corresponding to this closure set.
  - If the state corresponding to this set is new, add it to the list.
NFA to DFA Example

![NFA to DFA Example Diagram](image)

The diagram represents a transition between states in an NFA and DFA. The states and transitions are as follows:

- **State $q_1$**
  - Transition on 0, 1 to $q_2$
- **State $q_2$**
  - Transition on 0, 1, $\epsilon$ to $q_3$
- **State $q_3$**
  - Transition on 0, 1 to $q_4$
- **State $q_4$**
  - Final state

The diagram illustrates the conversion process from an NFA to a DFA.
NFA to DFA Example

Step 1

<table>
<thead>
<tr>
<th>state</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_1, q_2, q_3$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3, q_4$</td>
<td>$q_3, q_4$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_4$</td>
<td>$q_4$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
NFA to DFA Example

Step 2
Initially $S = \{ \{ q_1 \} \}$
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
Step 3

- \( \{q_1\} \) on 0 changes the state to \( \{q_1\} \)
- \( \{q_1\} \) on 1 changes the state to \( \{q_1, q_2, q_3\} \). *New!*
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. **New!**
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. **New!**
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. **New!**
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
- \{q_1, q_3, q_4\} on 0 changes the state to \{q_1, q_4\}. New!
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
- \{q_1, q_3, q_4\} on 0 changes the state to \{q_1, q_4\}. New!
- \{q_1, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
- \{q_1, q_3, q_4\} on 0 changes the state to \{q_1, q_4\}. New!
- \{q_1, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}.
- \{q_1, q_2, q_3, q_4\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
- \{q_1, q_3, q_4\} on 0 changes the state to \{q_1, q_4\}. New!
- \{q_1, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}.
- \{q_1, q_2, q_3, q_4\} on 0 changes the state to \{q_1, q_3, q_4\}.
- \{q_1, q_2, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}.
NFA to DFA Example

Step 3

- $\{q_1\}$ on 0 changes the state to $\{q_1\}$
- $\{q_1\}$ on 1 changes the state to $\{q_1, q_2, q_3\}$. New!
- $\{q_1, q_2, q_3\}$ on 0 changes the state to $\{q_1, q_3, q_4\}$. New!
- $\{q_1, q_2, q_3\}$ on 1 changes the state to $\{q_1, q_2, q_3, q_4\}$. New!
- $\{q_1, q_3, q_4\}$ on 0 changes the state to $\{q_1, q_4\}$. New!
- $\{q_1, q_3, q_4\}$ on 1 changes the state to $\{q_1, q_2, q_3, q_4\}$.
- $\{q_1, q_2, q_3, q_4\}$ on 0 changes the state to $\{q_1, q_3, q_4\}$.
- $\{q_1, q_2, q_3, q_4\}$ on 1 changes the state to $\{q_1, q_2, q_3, q_4\}$.
- $\{q_1, q_4\}$ on 0 changes the state to $\{q_1\}$. New!
NFA to DFA Example

Step 3

- \{q_1\} on 0 changes the state to \{q_1\}
- \{q_1\} on 1 changes the state to \{q_1, q_2, q_3\}. New!
- \{q_1, q_2, q_3\} on 0 changes the state to \{q_1, q_3, q_4\}. New!
- \{q_1, q_2, q_3\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}. New!
- \{q_1, q_3, q_4\} on 0 changes the state to \{q_1, q_4\}. New!
- \{q_1, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}.
- \{q_1, q_2, q_3, q_4\} on 0 changes the state to \{q_1, q_3, q_4\}.
- \{q_1, q_2, q_3, q_4\} on 1 changes the state to \{q_1, q_2, q_3, q_4\}.
- \{q_1, q_4\} on 0 changes the state to \{q_1\}.
- \{q_1, q_4\} on 1 changes the state to \{q_1, q_2, q_3\}.
### NFA to DFA Example

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>{q_1}</td>
<td>{q_1}</td>
<td>{q_1, q_2, q_3}</td>
</tr>
<tr>
<td>{q_1, q_2, q_3}</td>
<td>{q_1, q_3, q_4}</td>
<td>{q_1, q_2, q_3, q_4}</td>
</tr>
<tr>
<td>{q_1, q_3, q_4}</td>
<td>{q_1, q_4}</td>
<td>{q_1, q_2, q_3, q_4}</td>
</tr>
<tr>
<td>{q_1, q_4}</td>
<td>{q_1}</td>
<td>{q_1, q_2, q_3}</td>
</tr>
<tr>
<td>{q_1, q_2, q_3, q_4}</td>
<td>{q_1, q_3, q_4}</td>
<td>{q_1, q_2, q_3, q_4}</td>
</tr>
</tbody>
</table>

A state in the DFA is an accepting state if at least one of the NFA states it contains is accepting.