Regular Operations on Languages

A, B languages

\( A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \) \hspace{1cm} \text{(union)}

\( A \circ B = \{xy \mid x \in A \text{ and } y \in B \} \) \hspace{1cm} \text{(concatentation)}

\( A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \) \hspace{1cm} \text{(star)}

**Th:** If \( A_1 \) and \( A_2 \) are regular, then \( A_1 \cup A_2 \) is regular.

**Idea:** \( A_1 \) has \( M_1 \) that accepts \( A_1 \), \( A_2 \) has \( M_2 \), accepts \( A_2 \).
We need \( M \) that accepts \( A_1 \cup A_2 \).

\( M \) is constructed from \( M_1 \) and \( M_2 \) by simulating both;
if either would accept, \( M \) should accept.

Can’t do them one after the other; need to do simultaneously.
New state will be a pair of old states.

---

**Proof:** Regular languages are closed under union

Let \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \)
Let \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \)

Construct \( M = (Q, \Sigma, \delta, q_0, F) \) to recognize
\( A_1 \cup A_2 \) as follows.

\( Q = \{(r_1, r_2) \mid r_1 \text{ in } Q_1 \text{ and } r_2 \text{ in } Q_2\} = Q_1 \times Q_2 \)
\hspace{1cm} \text{(Cartesian product)}

\( \delta((r_1,r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \)

\( q_0 = (q_1, q_2) \)

\( F = \{(r_1,r_2) \mid r_1 \text{ in } F_1 \text{ or } r_2 \text{ in } F_2\} \)
Union FA of FA’s

- **Q1** = \{ q0, q1, q2 \} = \{ 0,1 \}
  - States: q0, q1, q2
  - Alphabet: \{ 0,1 \}
  - Transition diagrams:
    - From q0: 0 \to q0, 1 \to q1
    - From q1: 1 \to q2
    - From q2: 0 \to q2

- **L(M1)** = \{ w \mid \text{the sum of w’s digits is a multiple of 3} \}

- **Q2** = \{ s0, s1 \} = \{ 0,1 \}
  - States: s0, s1
  - Alphabet: \{ 0,1 \}
  - Transition diagrams:
    - From s0: 0 \to s0, 1 \to s1
    - From s1: 1 \to s1

- **L(M2)** = \{ w \mid \text{the sum of w’s digits is a multiple of 2} \}

What about other Regular Operations?

- **A, B languages**
  - **A \cap B** (intersection)
  - **A \circ B** (concatenation)

- **DFA D1**
  - States: q0, q1
  - Alphabet: \{ 0,1 \}
  - Transition diagrams:
    - From q0: 0 \to q0, 1 \to q1
    - From q1: 1 \to q2

- **DFA D2**
  - States: s0, s1
  - Alphabet: \{ 0,1 \}
  - Transition diagrams:
    - From s0: 0 \to s0, 1 \to s1
    - From s1: 1 \to s1

- **A^*** (star)
  - May not be DFA!!
**Nondeterministic Finite Automata**

Nondeterminism allows several possible next states (no one state is determined)

<table>
<thead>
<tr>
<th>Deterministic</th>
<th>S.D.</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Computation**

**NFA that accepts multiples of 2 or 3**

\[ Q = \{ q_0, q_1, q_2, s_0, s_1, \text{new} \} \quad \Sigma = \{ 0,1 \} \]

\[ L(M) = \{ w \mid \text{sum of w's digits is a multiple of 2 or 3} \} \]
Formal Definition of Nondeterministic FA

A NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where,

1. $Q$ is a finite set of states
2. $\Sigma$ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \rightarrow P(Q)$ (power set of $Q$, include $\varepsilon$)
4. $q_0 \in Q$, the start state
5. $F \subseteq Q$, set of accept (final) states

We say that $M$ accepts $w$

if there is a sequence of states $r_0, r_1, \ldots, r_m$ in $Q$

and $w = y_1 y_2 \ldots y_m, y_i \in \Sigma \in$ such that

1. $r_0 = q_0$
2. $r_{i+1}$ element of $\delta(r_i, y_{i+1})$
3. $r_m$ is in $F$

---

NFA vs. DFA

NFA are seemingly more powerful (NOT!)
NFA allow choice of next state

When choice, NFA may be more compact

Handling $\varepsilon$

$q_1 \xrightarrow{\varepsilon} q_2$

If in $q_1$ reading input symbol $a$, and take $\varepsilon$ edge, still reading $a$, but in state $q_2$.

All $\varepsilon$ edges must be included in execution tree.

Qu: Is every DFA an NFA?
Example NFA

![Example NFA Diagram]

Execution on baabb

$L(M) = \text{ }$

More Example NFA's

![More Example NFA Diagram]

$L(M) = \text{Eq. DFA? no. of states?}$

Ex 1.15, p. 52
Equivalence of NFA and DFA

Def: Two NFA M1, M2 are equivalent if L(M1) = L(M2).

Th.: For each NFA, there is an equivalent DFA.

Proof idea:
Start with NFA N. Want DFA M with L(N) = L(M).
We will simulate NFA with DFA.
NFA has set of possible next states; DFA just has 1.
Solution: state of the DFA will be subset of states of NFA

Proof: Let N = (Q, Σ, δ, q0, F) be an NFA.
Case 1: N has no ε transitions.

M = (Q', Σ, δ', q0', F')

1. Q' = P(Q)
2. δ'(R, a) = \bigcup_{r \in R} δ(r, a)
3. q0' = {q0}
4. F' = \{ R \in Q' | R \cap F \neq \phi \}

Case 2: N has ε transitions.

Proof Idea: What needs to change? Have to account for all ε moves after read an input symbol.

Proof: For R a subset of Q, define
E(R) = \{ q | q can be reached from some state in R by following only ε edges \}

Change the construction in Case 1 as follows:
2. δ'(R, a) = \{ q \in Q | q \in E(δ(r, a)) for some r \in R \}
3. q0' = \{ E(q0) \}

M keeps track of exactly the subset of states N would be in, and accepts when N accepts.
### Equivalence of NFA and DFA

**Cor.:** A language is regular iff some NFA recognizes it.

**Example: NFA, construct DFA**

(Ex. 1.21 p. 57)

Get lots of states....can we do better?  
*States that can’t be reached from start  
states that have no incoming edges ..........More later!*