**DFA’s/Regular Languages**

DFA – Deterministic Finite Automata

**Regular Operators**

*Let A and B be two languages over the same alphabet*

1. The Union of A, B = \( A \cup B = \{ W : W \in A \text{ or } W \in B \} \)
2. The Concatenation of A, B = \( AB = \{ W W' : W \in A \text{ and } W' \in B \} \)
3. The Star if A aka Kleene Star or Kleene Closure of A
   \( A^* = \{ u_1 u_2 \ldots u_k : k \geq 0 \text{ and } u_i \in \Sigma \text{ for all } i \} \)

**Closure**

A collection of objects is closed under some operation. We can apply the operation to numbers of our collection and get another member

EX: \( \mathbb{N} = \{ 1, 2, 3, \ldots \} \)  
+ , \times = Closed  
\( \div, - \) = Not Closed

EX: M(Pg.31)  
\( A = \{ 0, 01 \} \)  \( B = \{ 1, 10 \} \)

\( A \cup B = \{ 0, 01, 1, 10 \} \)
\( AB = \{ 01, 010, 011, 0110 \} \)
\( A^* = \{ \text{E} , 0 , 01 , 00 , 001 , 010 , 0101 , 000 \ldots \} \) (Side Note: E means empty string)

Note: \( \Sigma = \{ 0, 1 \} \)  \( \Sigma^* = \) Every possible Binary String  (Side Note: \( \Sigma \) is the alphabet)

Define \( A^0 = \{ \text{E} \} \)

For \( k \geq 1 \)  \( A^k = AA^{k-1} \)

\[ A^* = \bigcup_{k=0}^{\infty} A^k \]
Three empty objects
φ - The empty language, a set with no strings
{E} - The language that only accepts the empty string
\( L(B) = \{E\} \quad |L(B)| = 1 \)

E is an Empty String, a sequence of length 0

Ex: \( φ^* = \{E\}^* = \{E\} \)

For any other language A, \( A^* \) is infinite (With finite strings)

**Theorem (M35)/(S45)**: The set of regular languages is closed under the union operation.
If A, B are regular languages, then \( A \cup B \) is a regular language

**Proof by Construction**

M accepts W \( \iff \) \( M_1 \) accepts W or \( M_2 \) accepts W

\( M = \{Q, \Sigma, S, q, F\} \), \( M_1 = \{Q_1, \Sigma_1, S_1, q_1, F_1\} \), \( M_2 = \{Q_2, \Sigma_2, S_2, q_2, F_2\} \)

We need to simulate both \( M_1 \) and \( M_2 \) simultaneously.
Construct \( M = \{Q, \Sigma, S, q, F\} \)

\[
Q = Q_1 \times Q_2 = \{(r_1, r_2) : r_1 \in Q_1 \land r_2 \in Q_2 \}
\]

\(|Q| = |Q_1| \times |Q_2|\)
If M is in state \((r_1, r_2)\) it means if \(M_1\) would have read the input string up to this point, it would be in \(r_1\). \(M_2\) would be in \(r_2\).

\[\Sigma = \Sigma_1 = \Sigma_2 \quad \ast \text{we can assure alphabets are equal}\]

Start state \((q_1, q_2) = q\)

- Set of accept states
  \[F = \{(r_1, r_2) : r_1 \in F_1 \text{ or } r_2 \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)\]

- The transition function

  \[\delta : Q \times \Sigma \rightarrow Q\]
  \[\delta ((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \quad \forall r_1 \in Q_1 \text{ and } r_2 \in Q_2 \text{ and } a \in \Sigma\]

“This concludes the construction of the finite automaton M that recognizes the union of \(A_1\) and \(A_2\). This construction is fairly simple, and thus its correctness is evident from the strategy described in the proof idea. More complicated constructions require additional discussion to prove correctness.” (Introduction to the Theory of Computation, Sipser, pg.47)

- Concatenation, star

Concatenation:

(When do you know to make jump?)

Ex: \(L(A) = \{0001, 010\} – \text{accepted strings}\)

010000100010001010 – star: don’t know how long the string will be (hard to identify when to stop)
NFA Homework Help

Determinism vs Non-Determinism

- Deterministic Finite Automate (DFA) – if in state \( q_i \) and we read a, then \( Q_i = \delta(q_i, a) \)
- In the non-deterministic machine, several choice may exist, and be chosen for the next stage
  - NFA – nondeterministic finite automata

Nondeterminism is a generalization of determinism
  - Every DFA is an NFA

![Diagram showing NFA](image)

(Just go all of the ways)

Ex: (S 48) (M 35)

N1

\[ q_1 \xrightarrow{1} q_2 \xrightarrow{O, E} q_3 \xrightarrow{1} q_4 \]

- E – you can go from \( q_2 \) to \( q_3 \) without ever reading anything

Differences

1) Multiple options
2) Switch states w/o reading \( \Sigma \)
3) No options for symbols
*as long as one is an accept state, it accepts

---

**Extra Notes/Examples**

**Ex 1.24 (S p. 45)**

\[ \Sigma = \text{standard 26 letters (a-z)} \]

A = \{good, bad\}

B = \{boy, girl\}

A \cup B = \{good, bad, boy, girl\}

A concatenation B = \{goodboy, goodgirl, badboy, badgirl\}

A* = \{ E, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, \ldots \}