1. Color code explained

Definition Block

Theorem Block

Proof Block

Proof that cannot be completed without more info so it is incomplete

2. Automata and Language

2.1. Finite State Automate

a FA is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function
4. $q_0 \in Q$ is the initial state
5. $F \subseteq Q$ is the set of accept states

2.2. Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ and let $w = w_0, w_1 ... w_n$ a string of $[\Sigma]$ M accepts w if $\exists r_0, r_1, ..., r_n \in Q$ where:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, w_{i+1})$ for $i = 0...n - 1$
3. $r_n \in F$

2.3. regular language

A language is regular if some finite automata recognizes it

2.3.1. regular operations

Let A and B be regular languages, there a three operations **union**, **concatenation** and **star** as follows :

- **Union:** $A \cup B = \{x \mid x \in A \vee x \in B\}$
- **Concatenation:** $A \circ B = \{xy \mid x \in A \wedge y \in B\}$
- **Star:** $A^{\star} = \{x_1, x_2, ..., x_n \mid n \geq 0 \wedge x_i \in A\}$

2.3.1.1. Theorem closedness under union

A and B are regular languages then $A \cup B$

2.3.1.1.1. Proof

Let M_1 a state machine recognizes A_1
Let M_2 a state machine recognizes A_2

we can construct M a state machine recognizes $A_1 \cup A_2$ defined as follows

1. $M.Q = M_1.Q \times M_2.Q$
2. $M.\Sigma = M_1.\Sigma \cup M_2.\Sigma$
3. $M.\delta((r_1, r_2), a) = (M_1.\delta(r_1, a), M_2.\delta(r_2, a))$
4. $M.q_0 = (M_1.q_0, M_2.q_0)$
5. $M.F = \{(f_1, f_2) \mid f_1 \in M_1.F \vee f_2 \in M_2.F\}$

2.3.1.2. Theorem colosedness under concatenation

A and B are regular languages then $A \circ B$ is a regular languages

2.3.1.2.1. incomplete Proof

Let M_1 a state machine recognizes A_1
Let M_2 a state machine recognizes A_2

we can construct M a state machine recognizes $A_1 \circ A_2$ defined as follows

Cannot be done without non determinism

2.4. Non deterministic finite state Automata NFA

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

1. Q is a finite set called states
2. Σ is a finite set called the alphabet here sigma includes ε the empty input
3. $\delta : Q \times \Sigma \longrightarrow P(Q)$ is the transition function
4. $q_0 \in Q$ is the initial state
5. $F \subseteq Q$ is the set of accept states

2.4.1. Theorem equivalence of NFA and DFA

DFA and NFA are equivalent meaning forall languages recognized by DFA there is an NFA that recognizes and the reciproc is valid

2.4.1.1. Proof: DFA ⇒ NFA

forall DFA there is an NFA:
Let M_1 a DFA recognizes A
we can construct an NFA M that recognizes A

1. $M.Q = M_1.Q$
2. $M.\Sigma = M_1.\Sigma$
3. $M.\delta(q, a) = \{M_1.\delta(q, a)\}$
4. $M.q_0 = M_1.q_0$
5. $M.F = M_1.F$

2.4.1.2. Proof: NFA $\Rightarrow$ DFA

forall NFA there is a DFA:
Let M_1 a NFA recognizes A
we can construct an NFA M that recognizes A

1. $M.Q = P(M_1.Q)$
2. $M.\Sigma = M_1.\Sigma$
3. $M.\delta(R, a) = \bigcup_{r \in R} M_1.\delta(r, a) \cup \bigcup_{r \in R} M_1.\delta(M_1.\delta(r, a), \varepsilon)$ the second union here is to account for state that have and ε/empty transition
4. $M.q_0 = \{M_1.q_0\}$
5. $M.F = \{f \mid f \in M.Q \text{ where } \exists x \in f \text{ such that } x \in M_1.F\}$ in human it means the final state of M must containt at least one final state of M_1

2.4.1.3. Proof: NFA $\Longleftrightarrow$ DFA

using **NFA** $\Rightarrow$ **DFA** and **DFA** $\Rightarrow$ **NFA** we get **NFA** $\Longleftrightarrow$ **DFA**

2.4.1.4. Theorem Closure under union using NFA

A and B are regular languages then $A \cup B$ is a regular language

2.4.1.4.1. Proof

Let N_1 recognize A_1 , and N_2 recognize A_2 we can construct N such that it can recognize $A_1 \cup A_2$ as follows:

1. $N.Q = \{q_0\} \cup N_1.Q \cup N_2.Q$
2. $N.\Sigma = N_1.\Sigma \cup N_2.\Sigma$
3. $N.\delta$ is defined as follows
   - $N.\delta(q_0, \varepsilon) = \{N_1.q_0, N_2.q_0\}$
   - $N.\delta(N_1.q, a) = N_1.\delta(N_1.q, a)$
   - $N.\delta(N_2.q, a) = N_2.\delta(N_2.q, a)$
   - $N.\delta(\_, \_) = \emptyset$
4. q_0 is the starting state of
5. $N.F = N_1.F \cup N_2.F$

2.4.1.5. Theorem Closure under concatenation using NFA

A and B are regular languages then $A \circ B$ is a regular language

Let N_1 recognize A_1 , and N_2 recognize A_2 we can construct N such that it can recognize $A_1 \circ A_2$ as follows:

1. $N.Q = N_1.Q \cup N_2.Q$
2. $N.\Sigma = N_1.\Sigma \cup N_2.\Sigma$
3. $N.\delta$ is defined as follows
   - $N.\delta(q, \varepsilon) = N_1.\delta(q, \varepsilon) \cup \{N_2.q_0\}$ if $q \in N_1.F$
   - $N.\delta(N_1.q, a) = N_1.\delta(N_1.q, a)$
   - $N.\delta(N_2.q, a) = N_2.\delta(N_2.q, a)$
4. $N.q_0 = N_1.q_0$
5. $N.F = N_2.F$

2.4.1.6. Theorem Closure under star using NFA

A is a regular language then $A^\star$ is a regular language

Let N_1 recognize A, we can construct N such that it can recognize $A^\star$ as follows:

1. $N.Q = \{N.q_0\} \cup N_1.Q$
2. $N.\Sigma = N_1.\Sigma$
3. $N.\delta$ is defined as follows
   - $N.\delta(N.q_0, \varepsilon) = \{N_1.q_0\}$
   - $N.\delta(N.q_0, \_) = \emptyset$
   - $N.\delta\left(N.q_f, \varepsilon\right) = \{N_1.q_0\} \cup N_1.\delta\left(N.q_f, \varepsilon\right)$ where $N.q_f \in N.F$
   - $N.\delta(q, a) = N_1.\delta(q, a)$
4. $N.q_0$ is a new state
5. $N.F = N.q_0 \cup N_1.F$

2.5. Regular Expression

we Say R is a regular expression of an alphabet Σ if R is:

1. $a \in \Sigma$
2. ε
3. $\emptyset$
4. $R_1 \cup R_2$
5. $R_1 \circ R_2$
6. $R_1^\star$

where R_1, R_2 are regular expressions