Theory of Computation

Condensed Notes

May 4, 2025

Chapter 1

Regular Language and Finite Automata

1.1 Compound FA (D1 x D2)

Algorithm to construct

If $D_1 = (Q, \Sigma, \delta_1, q_0, F_1)$ and $D_2 = (Q, \Sigma, \delta_2, q_0, F_2)$

Then :

- Number of states in any compound FA $(D_1 X D_2)$ = mn

  where:

  - m = no. of states of D_1
  - n = no. of states of D_2

- The set of states of the compound FA $(D_1 X D_2)$ is the cross product of individual set of states of D_1 and D_2.

- Initial state of $(D_1 X D_2)$ is (q_1, q_2)

  where:

  - q_1 = initial state of D_1
  - q_2 = initial state of D_2

- If the final state of $D_1 = f(D_1)$, Language of D_1 is $L(D_1)$

  the final state of $D_1 = f(D_1)$, Language of D_1 is $L(D_1)$

and

the final state of $(D_1XD_2) = f(D_1XD_2)$, Language of (D_1XD_2) is $L(D_1XD_2)$

Final state and language of (D_1XD_2) is as follows

- $f(D_1 \cup D_2) = \{$W : W has either $f(D_1)$ or $f(D_2)\}$
- $L(D_1 \cup D_2) = L(D_1) \cup L(D_2)$
- $f(D_1 \cap D_2) = \{$W : W has both $f(D_1)$ and $f(D_2)\}$
- $L(D_1 \cap D_2) = L(D_1) \cap L(D_2)$
- $(D_1 - D_2) = \{(q_{f1}, q)$: q_{f1} is the final state of D_1 and q is a nonfinal state of $D_2\}$
- $L(D_1 - D_2) = L(D_1) - L(D_2)$
- $(D_2 - D_1) = \{(q, q_{f2})$: q_{f2} is the final state of D_2 and q is a nonfinal state of $D_1\}$
- $L(D_2 - D_1) = L(D_2) - L(D_1)$

1.2 Interconversion of Finite Automations

1.2.1 NFA to DFA conversion

Algorithm to convert
if NFA $= (Q, \Sigma, \delta, q_0, F)$ and DFA $= (2^Q, \Sigma, \delta', q_0, F')$

1. **Initial state** of DFA is same as NFA.

2. start with the initial state, for any given string in Σ , construct new states of DFA which is a set of states of NFA.

   for example if in NFA $q_0 \xrightarrow{a} q_1$ and $q_0 \xrightarrow{a} q_2$ the for DFA $q_0 \xrightarrow{a} \{q_1, q_2\}$

   This $\{q_1, q_2\}$ is a new state.

3. For the new found state eg:$\{q_1, q_2\}$ (in this example) construct new transition as set of states coressponding to NFA.

   for example if in NFA $q_1 \xrightarrow{a} q_3$ and $q_2 \xrightarrow{a} q_4$ the for DFA $\{q_1, q_2\} \xrightarrow{a} \{q_3, q_4\}$

4. thus we continue to get new states untill we have constructed all the set or no new state is encountered.

5. **Final states** of DFA is all the states which contain any of the final states of NFA.

   if there are n states in NFA then there can be nomore than 2^n states in coressponding DFA.

1.2.2 ϵ-NFA to NFA conversion

Algorithm to convert
if ϵ-NFA $= (Q, \Sigma, \delta, q_0, F)$ and NFA $= (Q, \Sigma, \delta', q_0, F')$

1. **Initial state** of ϵ-NFA is same as NFA.

2. **transition function** of ϵ-NFA : $\delta'(q, a) = \epsilon{-}closure(\delta(\epsilon{-}closure(q), a))$

   $\epsilon - closure(\{q_1, q_2, ..\})$ is the set of states which are reachable from the set of states $\{q_1, q_2, ..\}$ by multiple ϵ.

3. For the new found state eg:$\{q_1, q_2\}$ (in this example) construct new transition as set of states coressponding to NFA.

4. **Final states** : $\{W$:final state of $\epsilon - NFA \in \epsilon - closure(W)\}$

   number os states in NFA and ϵ-NFA is same

1.2.3 ϵ-NFA to DFA conversion

Algorithm to convert
if ϵ-NFA $= (Q, \Sigma, \delta, q_0, F)$ and DFA $= (2^Q, \Sigma, \delta', q_0, F')$

1. **Initial state** of DFA is ϵ-closure(initial states of ϵ-NFA).

2. **transition function** of DFA : $\delta'(q, a) = \epsilon - closure(\delta(q, a))$

3. **Final states** of DFA is all the states which contain any of the final states of ϵ-NFA.

Chapter 2

Context Free Language and Push Down Automata

2.1 Definition

Reduced CFG (non-redundent CFG):Reduced CFG is a CFG without any useless symbols.
Simplified CFG:simplified CFG is a CFG without any null productions, unit productions, useless symbols.
Null productions:$X \to \epsilon$ X is a variable.
Unit productions:$A \to B$ A and B are variable.
Useless symbols:symbols/terminals that cant be used in any derivation of string.

2.2 Conversion of CFG into Simplified CFG