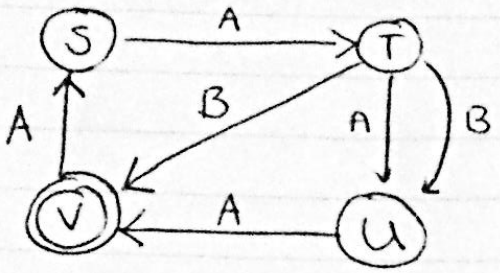
CO519, Regular Languages

1. NFA & !DFA:



A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of:

- a finite set (Q)
- a finite set of input (Σ) [alphabet]
- transition function: $(\delta : Q \times \Sigma \to Q)$
- a start state $(q_0 \in Q)$
- a set of accept states $(F \subseteq Q)$

DFA is represented formally by a 5-tuple $(Q, \Sigma, \Delta, q_0, F)$ consisting of:

- a finite set (Q)
- a finite set of input (Σ)
- transition ~~funct~~ relation $\Delta : Q \times \Sigma \to P(Q)$
- an ~~idiot~~ initial state (start state) $q_0 \in Q$
- a set of States F distinguished as accepting (or final) States $F \subseteq Q$

Here, $P(Q)$ denotes the power set of Q.
From the definition, there is a different between the transitions (functions): For any input symbol into DFA transits to a single particular state while NFA and trans to multiple States. i.e. in DFA the next possible state is distinctly set while in NFA each pair of State and input symbol can have many possible next sta

Advantages:

NFA
- used to empty string transition while DFA cannot use empty string transitions
- Easier to construct
- Back tracking is allowed
- Less space/memory