Cheatography

b Cheat Sheet

by mrbhalerao via cheatography.com/141437/cs/31834/

DFA (Deterministic Finite Automaton)

Automaton Representation	M=(Q,Σ,δ,q0,F)
Q: Set of states	{q0,q1,q2}
Σ: input alphabet	{a,b} & ε ∉ Σ
δ: transition function	δ(q,x)=q'
q0: initial state	
F: set of accepting states	{q2}
Language of Automaton	L(M)={ w∈Σ : δ(q0,w)∈F}

Languages are regular if a DFA exists for that language.
DFA: Each state has one transition for evrey alphabet

NFA (Non-deterministic Finite Automaton)

Formal Definition	M=(Q,Σ,Δ,S,F)
Q: Set of states	{q0,q1,q2}
Σ: input alphabet	{a,b}
Δ: transition function	Δ(q,x)={q1,q2,...} (include ε)
S: initial states	{q0}
F: set of accepting states	{q2}
Language of Automaton	L(M) = {w1,w2,...,wn}

NFA: Each state can have different transition with the same language output

NFA to DFA Conversion

1. Set initial state of NFA to initial state of DFA

2. Take the states in the DFA, and compute in the NFA what the union Δ of those states for each letter in the alphabet and add them as new states in the DFA.

For example if (q0,a) takes you to {q1,q2} add a state {q1,q2}. If there isn't one, the add state null

NFA to DFA Conversion (cont)

3. Set every DFA state as accepting if it contains an accepting state from the NFA

The language for the NFA (M) and DFA (M') are equivalent L(M)=L(M')

Properties of Regex

L1 ∪ L2	Initial state has two ε transitions, one to L1 and one to L2
L1L2L1L2	L1 accept state tranitions (ε) to L2 initial state
L1*	New initial state transitions to L1 initial state. New accept state transitions (ε) from L1 accept state. Initial to accept state transition transitions (ε) and vice versa.
L1R (reverse)	Reverse all transitions. Make initial state accepting state and vice versa.
!(L1) Complement	Accepting states become non-accepting and vice versa
L1 ∩ L2	!(!(L1) ∪ !(L2))

P.S. To turn multiple states to one accept state in an NFA, just add a new accept state, and add transition to the old accept states with language ε.

Intersection DFA1 ∩ DFA2

Transitions M1 : q1→ →q2 M2 : p1→ →p2	DFA M: (q1,p1)→ a →-(q2,p2)
Initial State M1: q0 M2 : p0	DFA M: (q0,p0)
Accept State M1 : qi M2 : pj,pk	DFA M: (qi,pj) , (qi,pk)

Regular Expressions

L(r1+r2)	= L(r1) ∪ L(r2)
L(r1•r2)	= L(r1)L(r2)
L(r1*)	= (L(r1))*
L(a)	= {a}

Precedence: * → → • → +

NFA to Regular Language

1. Transform each transition into regex (e.g. (a,b) is a+b

2. Remove each state one by one, until you are left with the initial and accepting state

3. Resulting regular expression: r = r1 *r2(r4+r3r1*r2)* where:
r1: initial → initial
r2: initial → accepting
r3: accepting → initial
r4: accepting → accepting

Proving Regularity with Pumping Lemma

Prove than an infinite language L is not regular:

1. Assume L is regular

2. The pumping lemma should hold for L

3. Use the pumping lemma to obtain a contradiction:

a. Let m be the critical length for L
b. Choose a particular string w∈L which

satisfies the length condition |w|≥m

c. Write w=xyz
d. Show that w'=xy z∉L for some i≠1