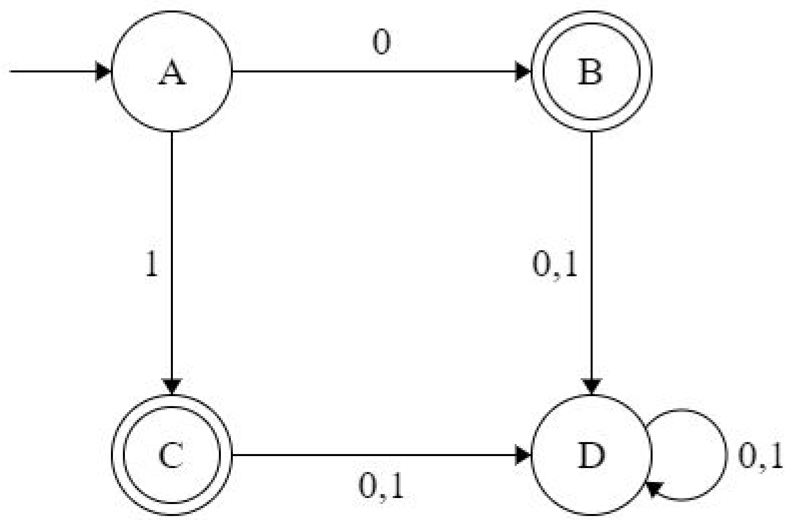
Regular languages and Finite Automata

Formal definition of Finite Automaton(5 tuples)

1. Q, a set of states,
2. Σ, the alphabet,
3. $\delta : Q \times \Sigma \rightarrow Q$, the transition function,
4. $q0 \in Q$, the start state, and
5. $F \subseteq Q$, the set of accepting states.

Eg.



Q = {A,B,C,D}
Σ = {0,1}
q0 = A
F = {B,C}

δ	0	1
A	B	C
B	D	D
C	D	D
D	D	D

At state A, on input 0, go to state B.
At state B, on input 1, go to state D, etc.

Construct transition table:

	a	b	c
X	X, Y, Z	X	Z
Y	Ø	Ø	Ø
Z	Ø	Ø	Y, Z

New transition table for DFA:

Procedure:

Start with q0 of NFA(X in this case), note down the states reachable from X(XYZ, X and Z)

Since DFA can only have a single transition between a pair of states, new states have to be created. (eg for NFA, X on input a can go to either X, Y or Z, with equivalent DFA, this have to be represented by a new state XYZ.)

From the reachable states, note down their reachable states on all inputs.

Repeat the process until no more reachable states can be noted.

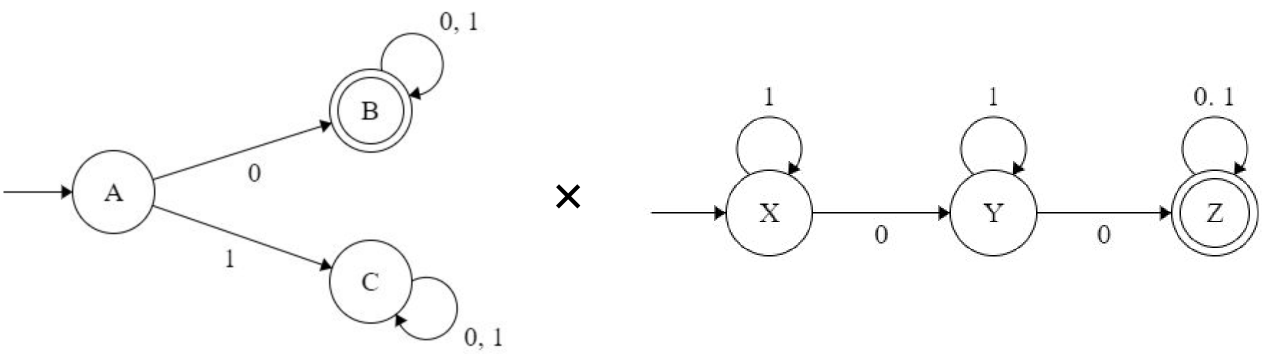
q0 for DFA is the same as q0 for NFA, F for DFA state that contains F for NFA. There might be needs to add a sink state(recall for each state in DFA there will be transitions representing all possible inputs.)

	a	b	c
X	XYZ	X	Z
Z	Ø	Ø	YZ
YZ	Ø	Z	YZ
XYZ	XYZ	XZ	YZ
XZ	XYZ	X	YZ

Product Construction

Each state is a pair of states
Transition is the product of transitions
q0 is the product of q0s
F -union: includes one of F
-intersection: include both Fs

Eg:



States: AX, AY, AZ, BX, BY, BZ, CX, CY, CZ

Transitions: A on 0 goes to B, X on 0 goes to Y, so AX on 0 goes to BY, etc

	0	1
AX	BY	CX
AY	BZ	CY
AZ	BZ	CZ
BX	BY	BX
BY	BZ	BY
BZ	BZ	BZ
CX	CY	CX
CY	CZ	CY
CZ	CZ	CZ

Chomsky Normal Form

CFG is in Chomsky Normal Form(CNF) is all its R is in the form

- $A \rightarrow BC$
- $A \rightarrow a$
- $S \rightarrow \varepsilon$

$A, B, C \in V$, $a \in \Sigma (a \neq \varepsilon)$, S(start variable) $\in V$

CNF Conversion:

1. Add new S_0 and production $S_0 \rightarrow S$
2. If any $A \rightarrow aB$ appears, do $A \rightarrow XB$, $X \rightarrow a$.
3. Convert $A \rightarrow B_1, B_2, B_3, ...B_n$ to $A \rightarrow B_1A_1$, $A_1 \rightarrow B_2A_2$, $A_2 \rightarrow B_3A_3$, …, $A_{n-2} \rightarrow B_{n-1}B_n$.
4. If $A \rightarrow \varepsilon$, substitute A where A appears in any RHS with ε, remove ε production unless A = S
5. If $A \rightarrow B$ and $B \rightarrow XYZ$, replace with $A \rightarrow XYZ$

Pumping Lemma for context free grammar

If L is a CFL, there is a number $p > 0$ such that for every $s \in L$ where $|s| \geq p$, s can be divided into five parts $s = uvwxy$ such that:

- $|vx| \geq 1$
- $|vwx| \leq p$
- $uv^nwx^ny \in L$ for all $n \in \mathbb{N}$

Procedure: Given L → assume L is a CFL → L has a pumping length p → find s where $|s| \geq p$ → show $uv^nwx^ny \notin L$ for some n → show there's no way to divide s into nvwxy that satisfies the 3 conditions → s cannot be pumped.

Eg. Given $L = \{a^nb^nc^n\}$, assume exist $p > 0$, let $s = a^pb^pc^p \in L$, then either:

- $vwx = a^i$ for some $1 \leq i \leq p$
- $vwx = a^ib^j$ for some $1 \leq i + j \leq p$
- $vwx = b^i$ for some $1 \leq i \leq p$
- $vwx = b^ic^j$ for some $1 \leq i + j \leq p$
- $vwx = c^i$ for some $1 \leq i \leq p$

In all cases, uvvwxxy has too many of one or two characters. Third character is not repeated enough.

Turing recognisable

A language is decidable if and only if it's Turing recognisable and co-Turing recognisable(it's complement is recognisable)

A and Ā are T-recognisable, let M_1 and M_2 be their recogniser respectively

M = On input w

1. Run M_1 and M_2 on w
2. If M_1 accept, accept. If M_2 accept, reject

A is decidable

E_{TM} is undecidable

E_{TM} = {<M> | M is a TM, L(M) = ø}

Show if some TM R decides E_{TM}, then TM S decides A_{TM}

Bulid M_1:

M_1 = "On input x:

1. If x ≠ ω, reject
2. If x = ω, run M on w and accept if M does"

S = On input (M, v)

1. Use the description of M to build M_1 as noted above
2. Run R on input <M_1>
3. If R accepts, reject, if R rejects, accept.

EQ_{TM} is undecidable

EQ_{TM} = {<M, N> | M and N are TMs with L(M) = L(N)}

Exist TM V decides EQ_{TM}, and TM X decides E_{TM}

X = On input <M> where M is a TM

1. Run V on <M, N> where N is a TM that reject all inputs
2. If V accepts, accept, if V rejects, reject

Thus if V decides EQ_{TM}, there exists X that decides E_{TM}.