Math 322

13.4 Language Recognition

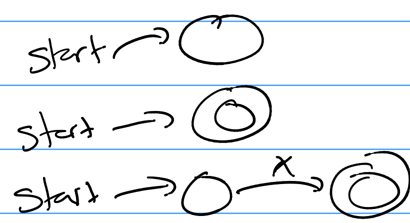
Regular Sets: **Basis:** $\emptyset$, λ, x

Inductive: A, B are regular sets
AB, $A \cup B$, A^* are regular sets

Thm A set is regular iff it is recognized by a FSA.

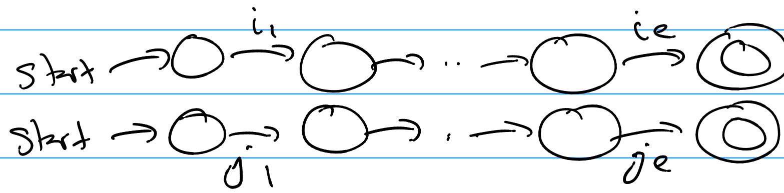
Proof: **Basis Machines:**

(1) $M_\emptyset$:
(2) M_λ :
(3) M_x :

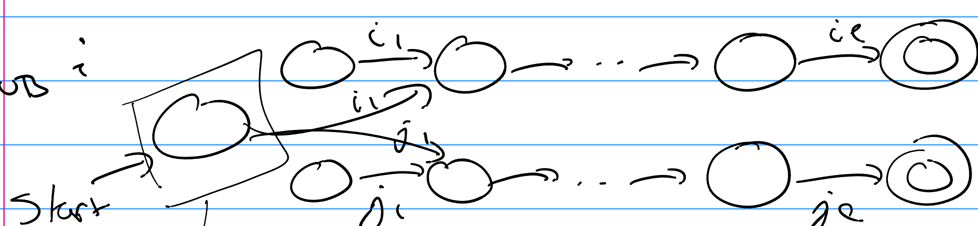


Inductive Machines:

assume M_A :
M_B :

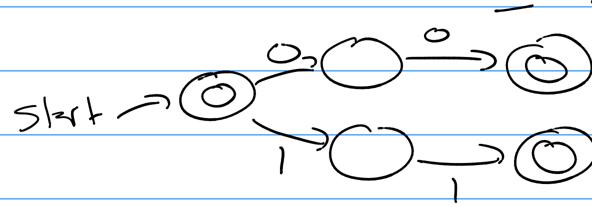


(1) $M_{A \cup B}$:

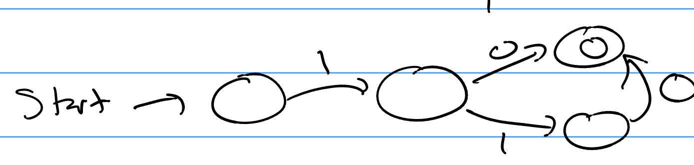


when is this final? if M_A or M_B start state was final.

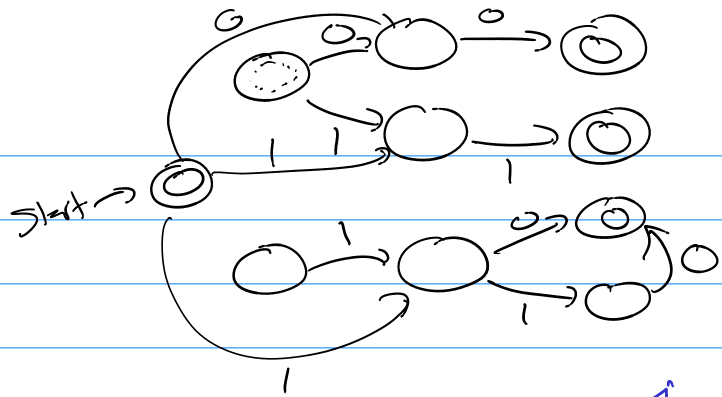
ex M_A :



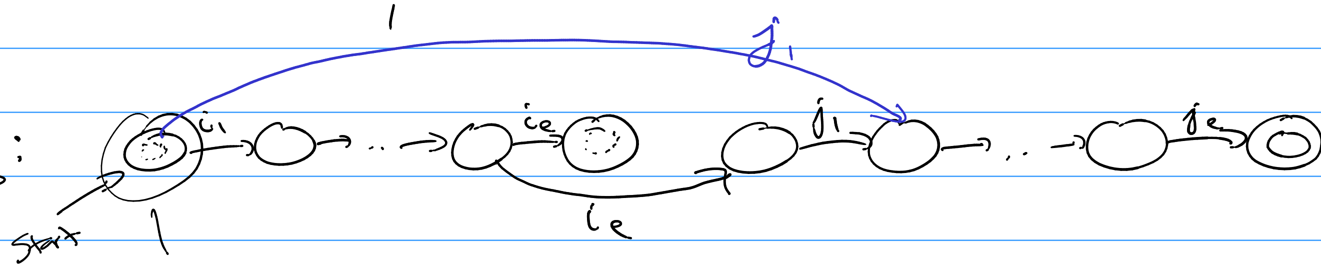
M_B =



$M_{A \cup B}$:



② M_{AB}:

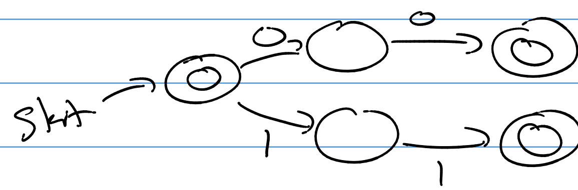


→ this final state?

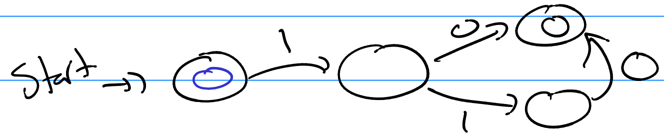
Yes, then add blue edge with j_1 input symbol.

ex

M_A:



M_B:

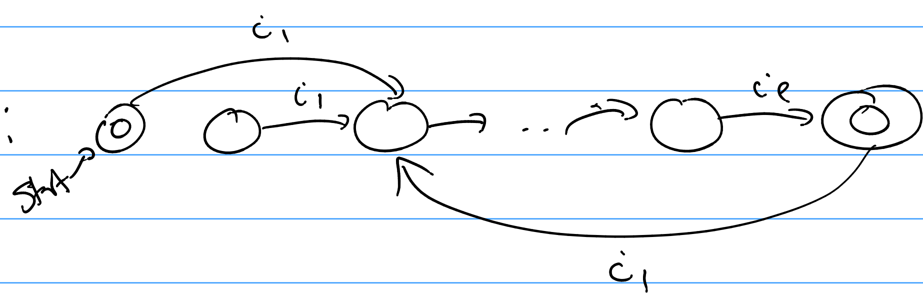


M_{AB}:

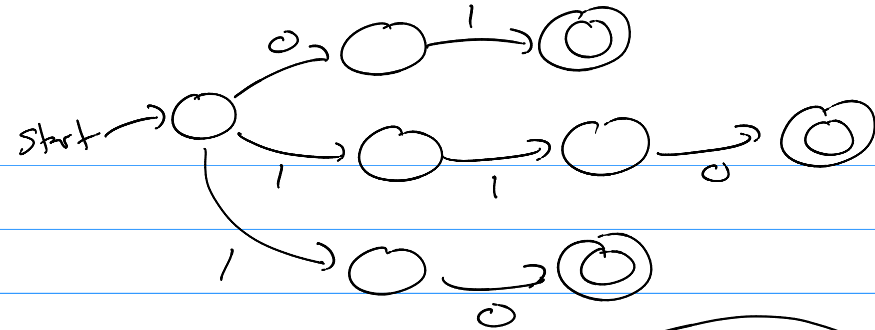
③ A^*

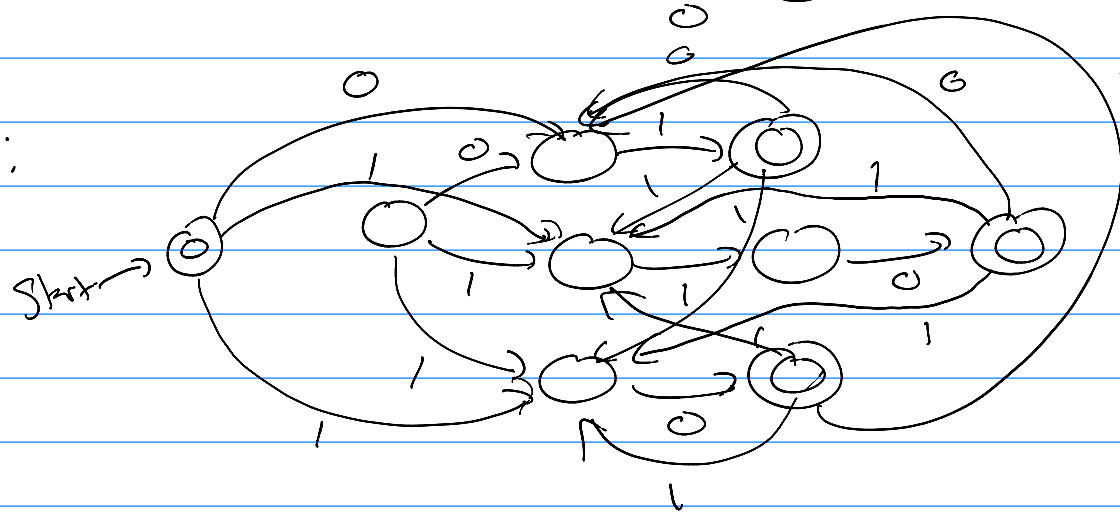
M_A:

M_{A^*}:



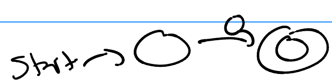
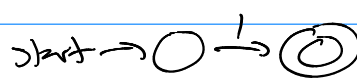
ex

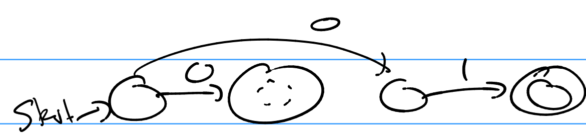
M_A: 

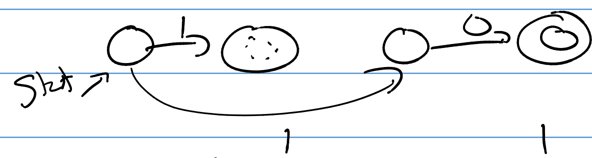
M_{A^*}: 

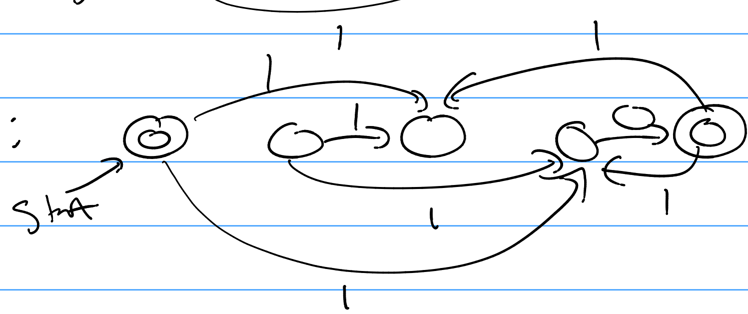
---

ex $0 1 \cup (10)^*$

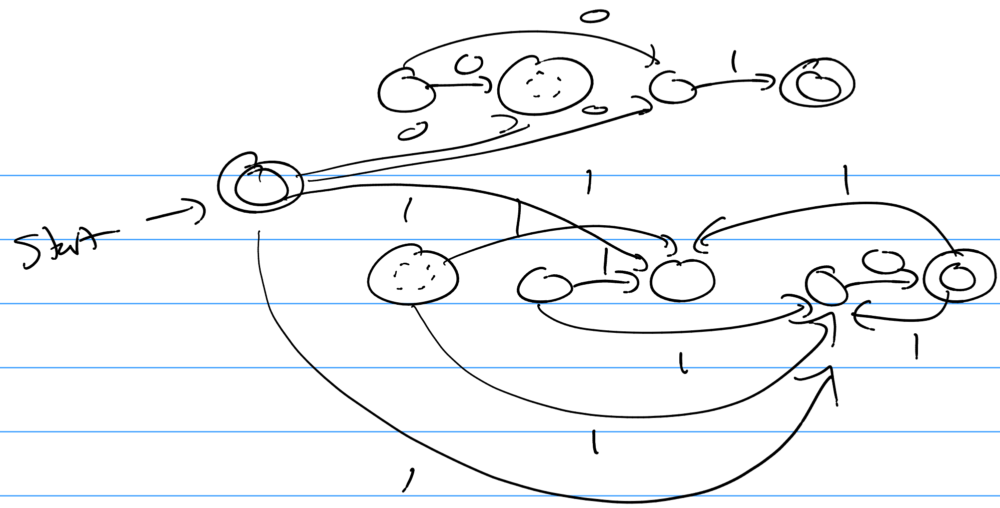
M_0:  M_1: 

M_{01}: 

M_{10}: 

$M_{(10)^*}$: 

$M_{a \cup (10)^*}$:



So.. regular set $\equiv$ recog. by FSA

Thm A set is generated by a regular grammar iff it is a regular set.

Pf Make a FSA feature for each production

regular grammar has productions of..

$S \to \lambda$

NonTerm $\to$ term ($A \to a$)

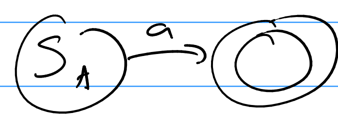
Non Term $\to$ term Non-term ($A \to bB$)

(1) $S \to \lambda$ FSA has:

Start state is final state

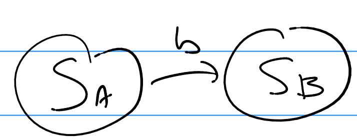
(2) Non Term $\to$ term

$A \to a$

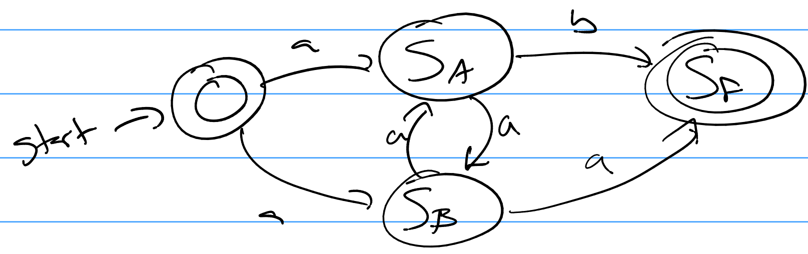


(3) Non term $\to$ term Non term

$A \to bB$



ex $P = \{S \to \lambda,\ S \to aA,\ S \to aB,\ A \to b,\ A \to aB,\ B \to a,\ B \to aA\}$



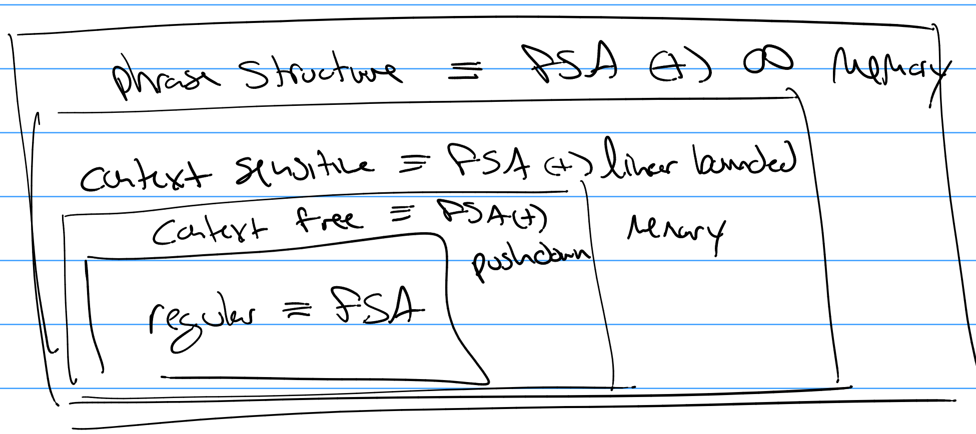
So Set is generated by a regular grammar ≡ recog. by FSA.

type3 regular grammar ≡ recog. by FSA

type2 context free grammar ≡ recog. by FSA with pushdown memory

type1 context sensitive grammar ≡ recog. by FSA with linear bounded tape memory

type0 phrase-structure grammar ≡ recog. by FSA with infinite tape memory.



Def: FSA with infinite tape memory = Turing Machine

Turing Machines

$T = (S, I, f, S_0)$

S : finite states

I : alphabet has B, a blank symbol

S_0 : start state

$f : S \times I \to S \times I \times \{\text{right}, \text{left}\}$

f is a partial function (not all domain $(S \times I)$ is defined)

infinite tape memory

① blank tape

② write input string to tape

③ place state machine above left most input symbol

Heart of T is $\boxed{f : S \times I \to S \times I \times \{R, L\}}$

set of 5-tuples

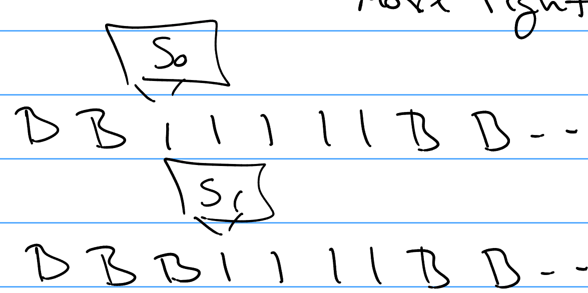
(ex) $S_0, 1 \to S_1, B, R$ $\equiv$ $\boxed{(S_0, 1, S_1, B, R)}$

In S_0, see 1 on tape $\to$ goto S_1

write B to tape

move right

(ex)



What makes a turing machine do any task is its set of 5-tuples.

ex $f = \{ (S_0, 1, S_1, B, R), (S_1, 1, S_2, B, R), (S_0, 0, S_0, 0, R)$
$(S_0, B, S_F, B, R), (S_1, 0, S_F, 0, R) \}$

$\boxed{S_0}$
-- B B 0 0 1 0 1 1 1 0 B B --

$\boxed{S_0}$
-- B B 0 0 1 0 1 1 1 0 B B --

$\boxed{S_0}$
-- B B 0 0 1 0 1 1 1 0 B B --

$\boxed{S_1}$
-- B B 0 0 B 0 1 1 1 0 B B --

$\boxed{S_F}$
-- B B 0 0 B 0 1 1 1 0 B B --

↑

no $(S_F, 1, \ldots)$ input pair so T halts.

Note: States: S_0, S_1, S_2, S_F

looking f I see only S_0, S_1 are possible inputs

→ call S_2, S_F to be final states

→ So T halts in a final State.

Church Turing thesis any problem that has an effective algorithm has a turing machine that will solve that problem.

Number theory

Number theoretic functions $\quad f: \mathbb{Z}^+ \to \mathbb{Z}^+$

ex) unary numbers

$0 = 1$

$1 = 11$

$2 = 111$

$3 = 1111$

$\vdots$

find $T = n_1 + n_2$

experiment: $2+3=5$

$I = \{B, 1, *\}$

-- B B 1 1 1 * 1 1 1 1 B B B ..

run T

B B B 1 1 1 1 1 1 B B B --

$f = \{(S_0, 1, S_1, B, R)$
$(S_1, 1, S_{11}, B, R)$
$(S_{11}, 1, S_{11}, 1, R)$
$(S_{11}, *, S_F, 1, R)$
$(S_1, *, S_F, B, R)\}$

try -- B B B B 1 1 1 1 1 1 B B --

S_F

5

S_0 S_F

-- B B B ~~B~~ 1 B B ..