Decision Problems 🡪 Language instead

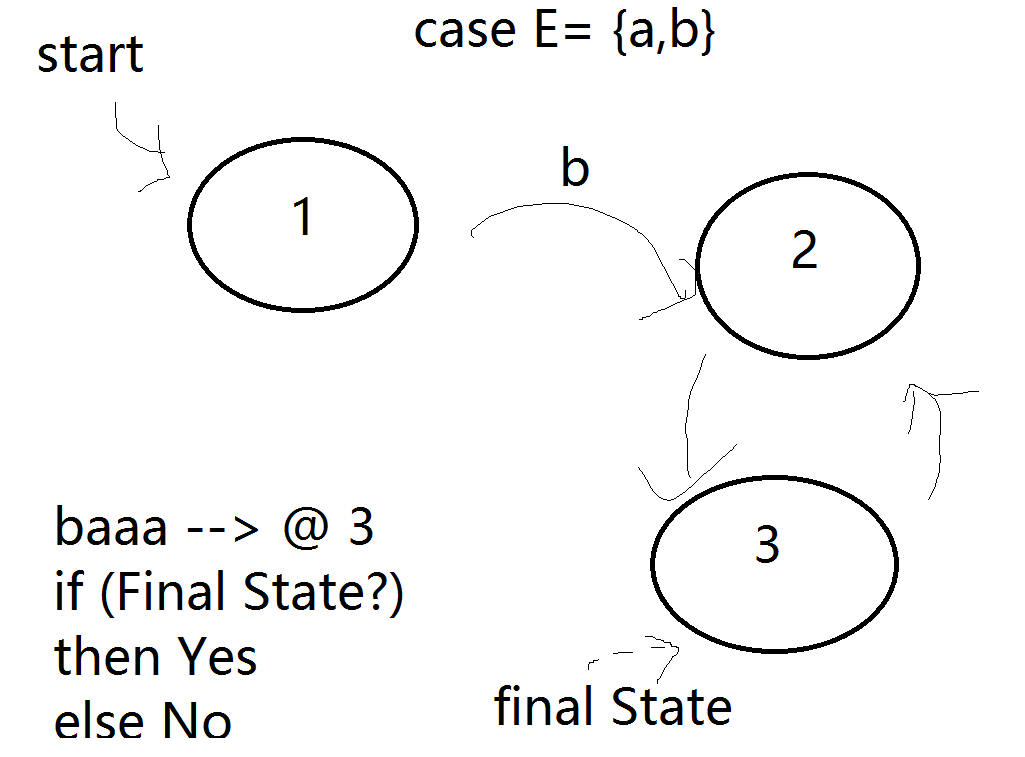
D: E\* -> {0,1}

Computable

Finite State Machines:

Start

1 🡪 2 🡪 3



Starting at the start state,

Follow transitions labeled by symbols, form the shj

If you end up in final, you accept the string, otherwise you don’t accept

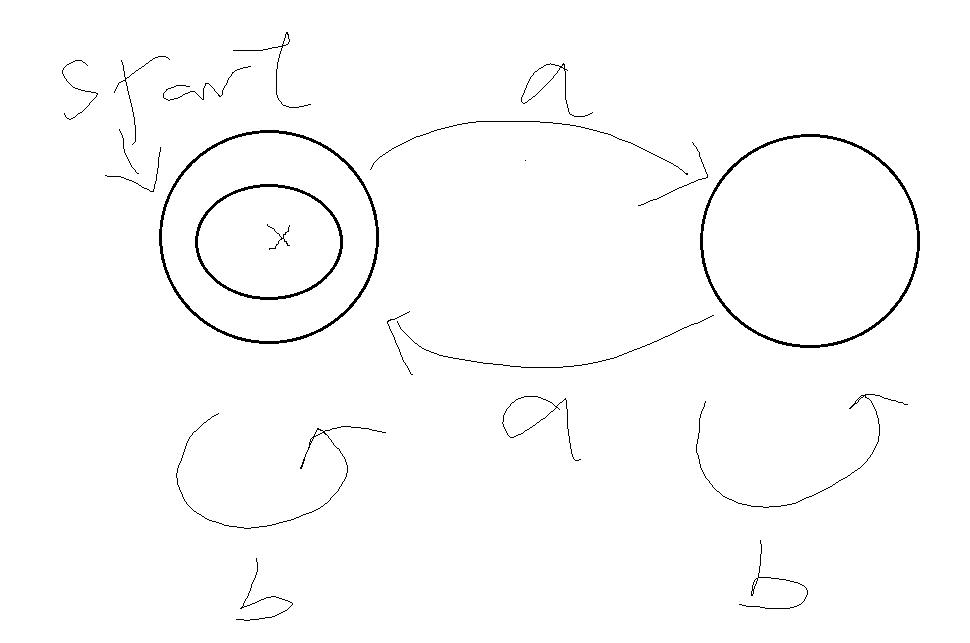
If you get stuck, you don’t accept it

A machine accepts a string u, if there is a way to follow transitions in M using the symbols in u and end up in final states

Pattern: the language accepted by the machine to the left is {ba^i | i is odd} = {ba, baaa, baaaaa , ...}

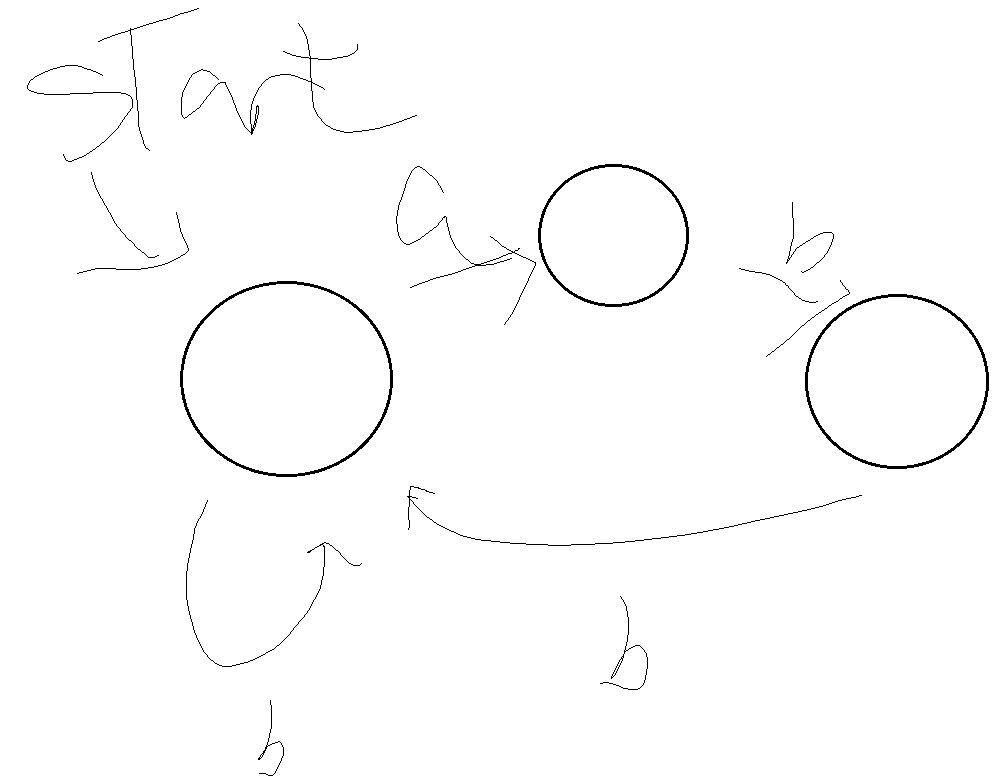
The language accepted by a finite state machine M

L (M) = {u belongs to E\*| M accept u}

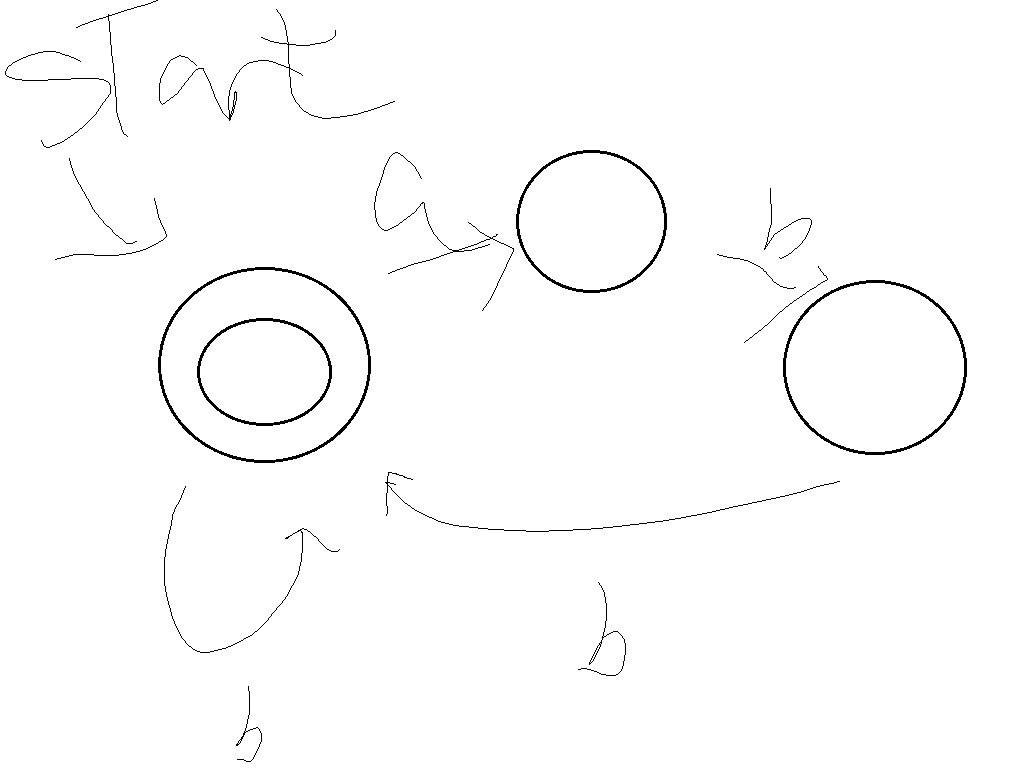


Language accepted in all strings with an even # of a’s (even includes 0 too)

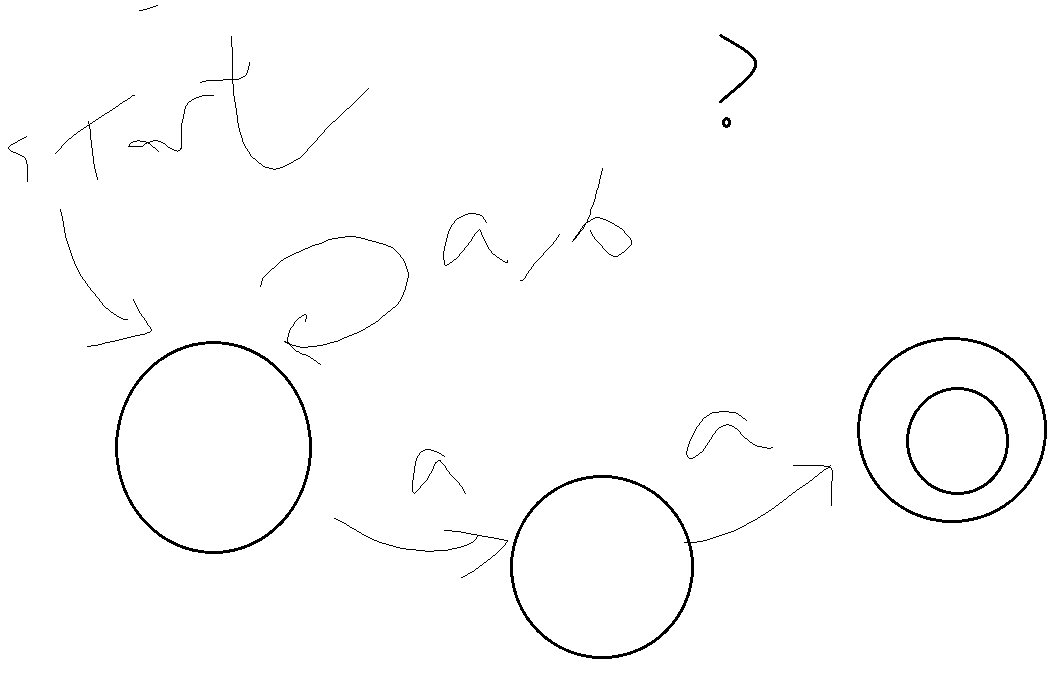
Can have no final State or multiple, only one start



Accept no strings because there isn’t a final stage



L: Strings that a immediately followed by at least 2 bs



L: Any strings that end with 2 a’s

Finite State Machines (Finite automata)

DEF: A finite automatom in a tuple

M = (Q, )

Q 🡪 a finite set of tuples

Sum 🡪 Alphabets

-> a set of tuples <p,a,q> (p belongs to Q, q belongs to Q, a belongs to SUM)

S -> start state (s )

F -> sets of final states (F < Q)

Ex : u ({1,2}, {a,b}, {<1,b,1>, <1,a,2>, <2,b,2>, <2,a,1>}, 1, {1})

Def: FA M = (Q, ) accepts a string u = a1 … ak if there exists states q1, … qk start state = S, qK F, and for all 1 < i < k , (qi-1, ai, qi .

The language accepted by M

L (M) = {u \*| M accepts u}

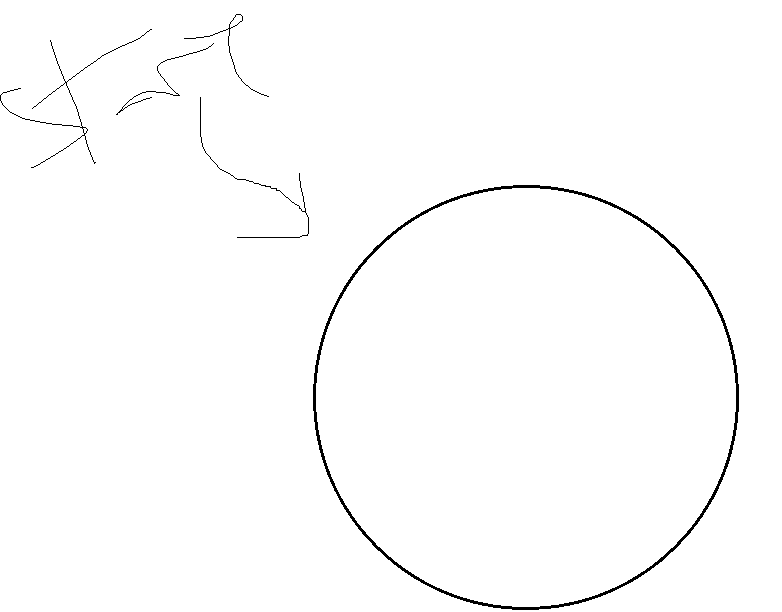
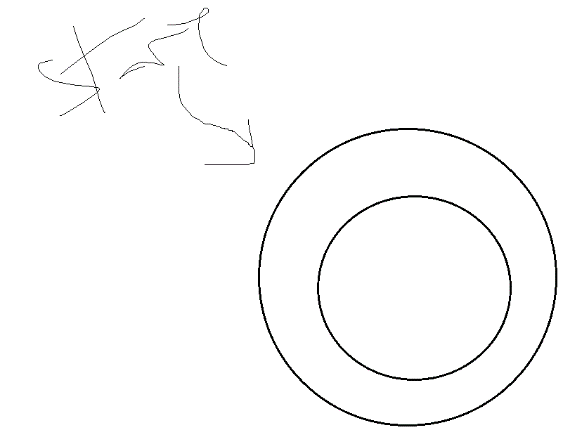
Theorem: A language is accepted by some finite automaton exactly when it’s regular

Easy: If A is regular, then Finite Automaton accepted A

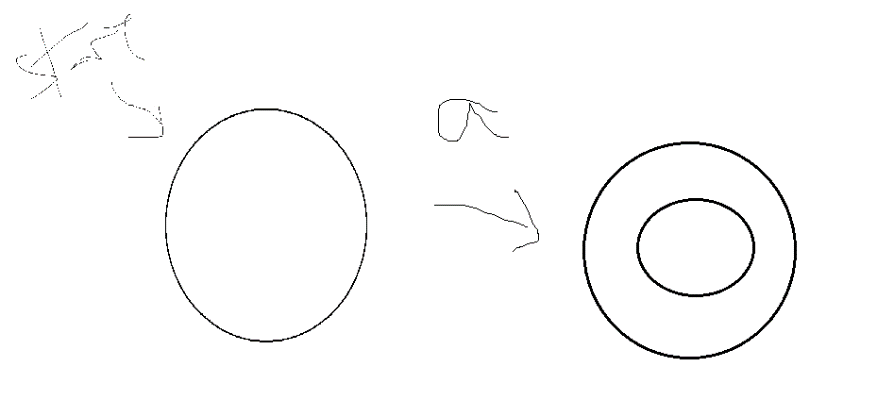
Hard: if A in accepted by Finite Automaton

We define a recursive procedure that constructs a finite automaton for any regular expression

1 L(1) = {} 0 L (0) =



a L(a) ={a}

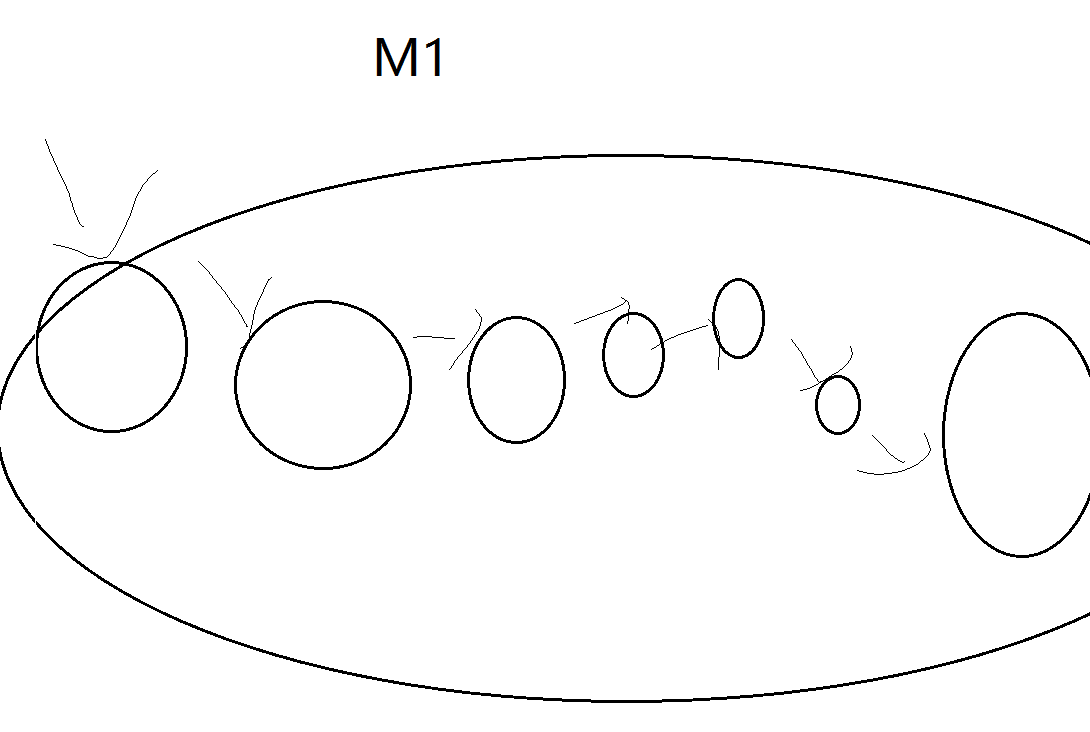


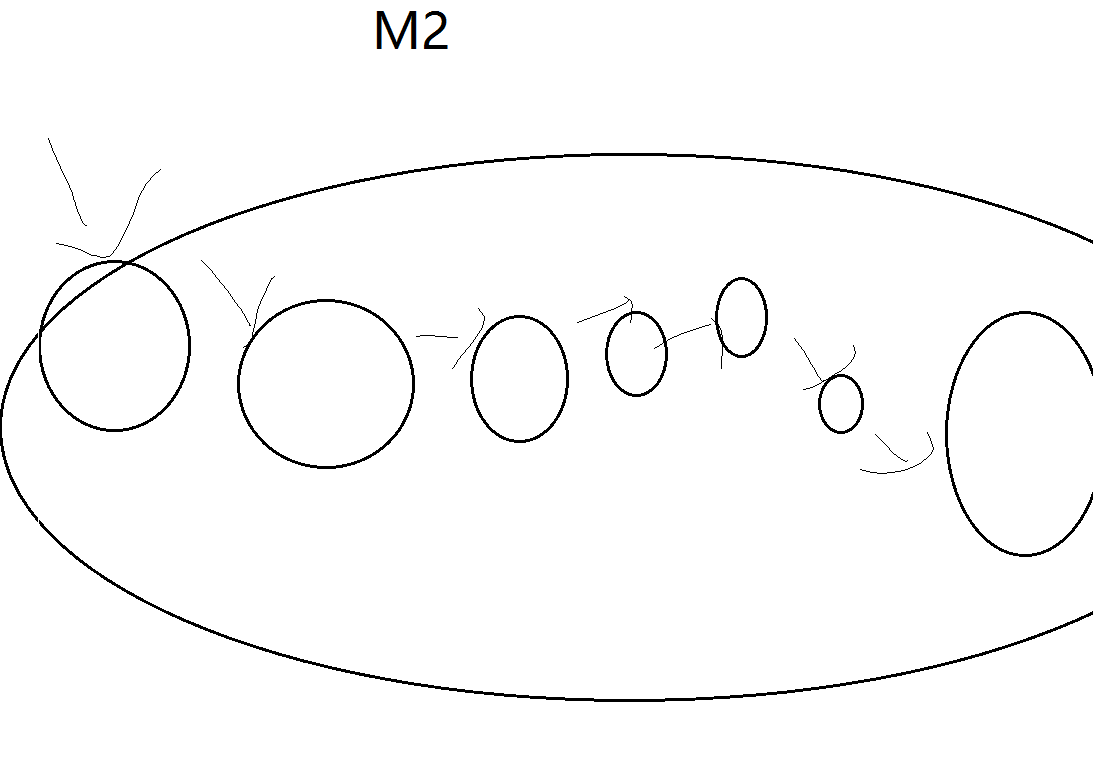
R1 +r2 , r1 . r2 , r1\*

R1+ r2 : L(r1 + r2) = L(r1) L(r2)

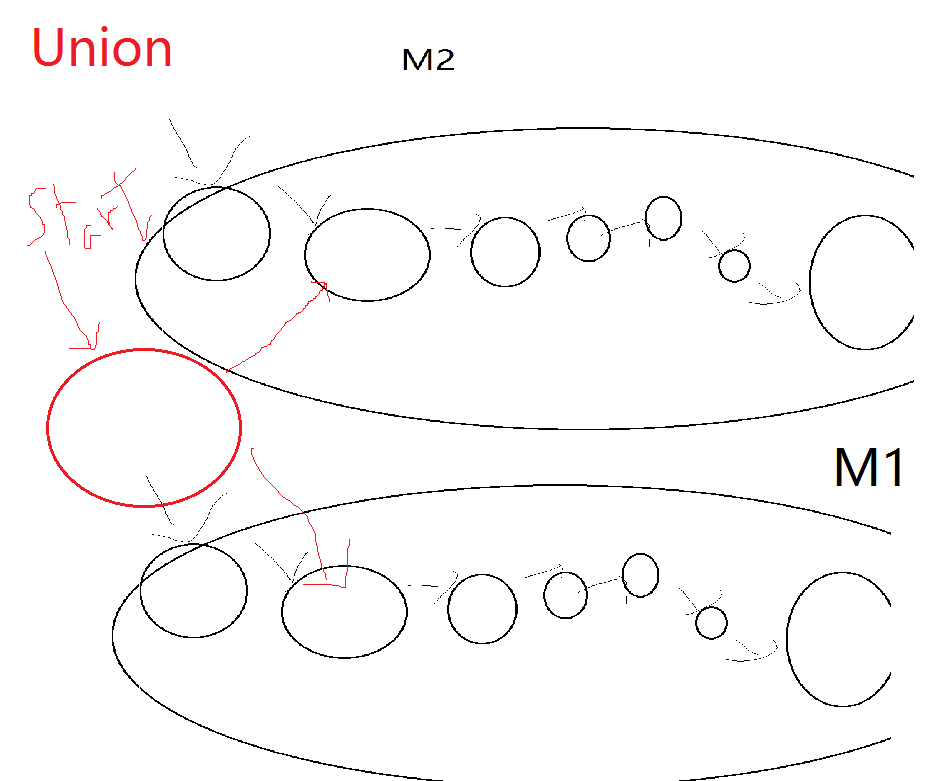
We can recursively construct M1 -> accept L(r1), M2 🡪 accept L(r2)

**M1 , M2**

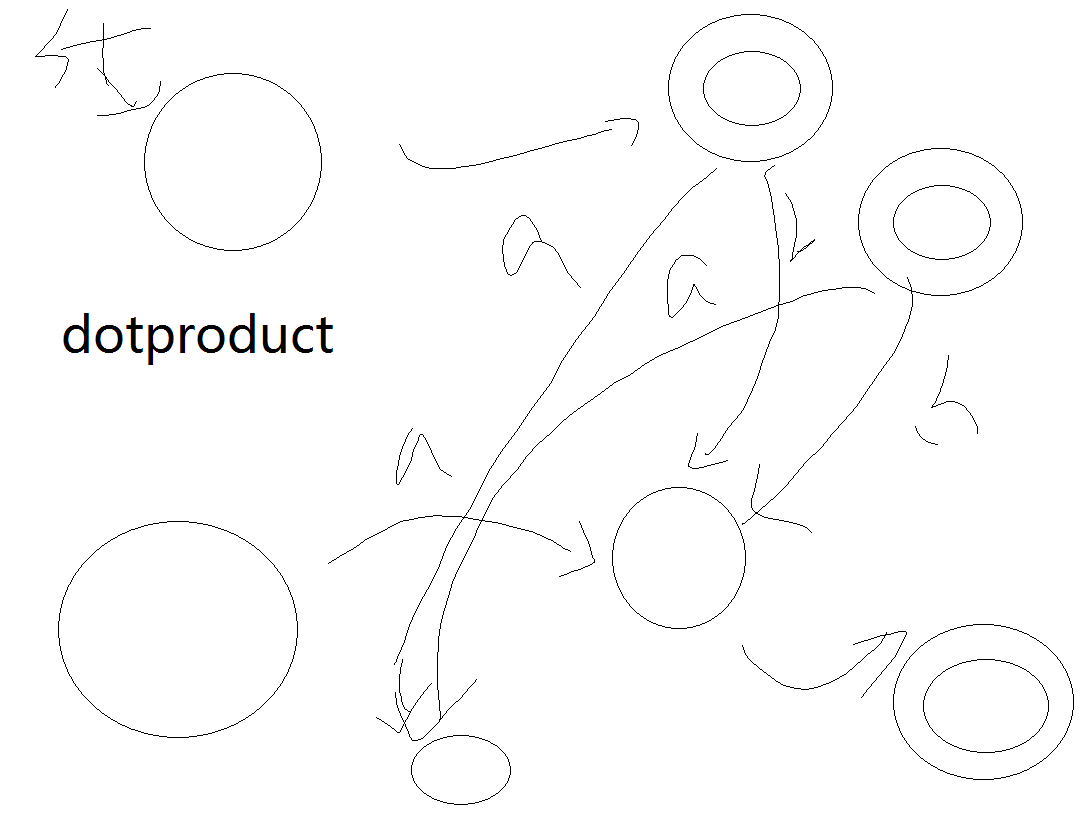
****

****

**New (r1+r2 = union)**

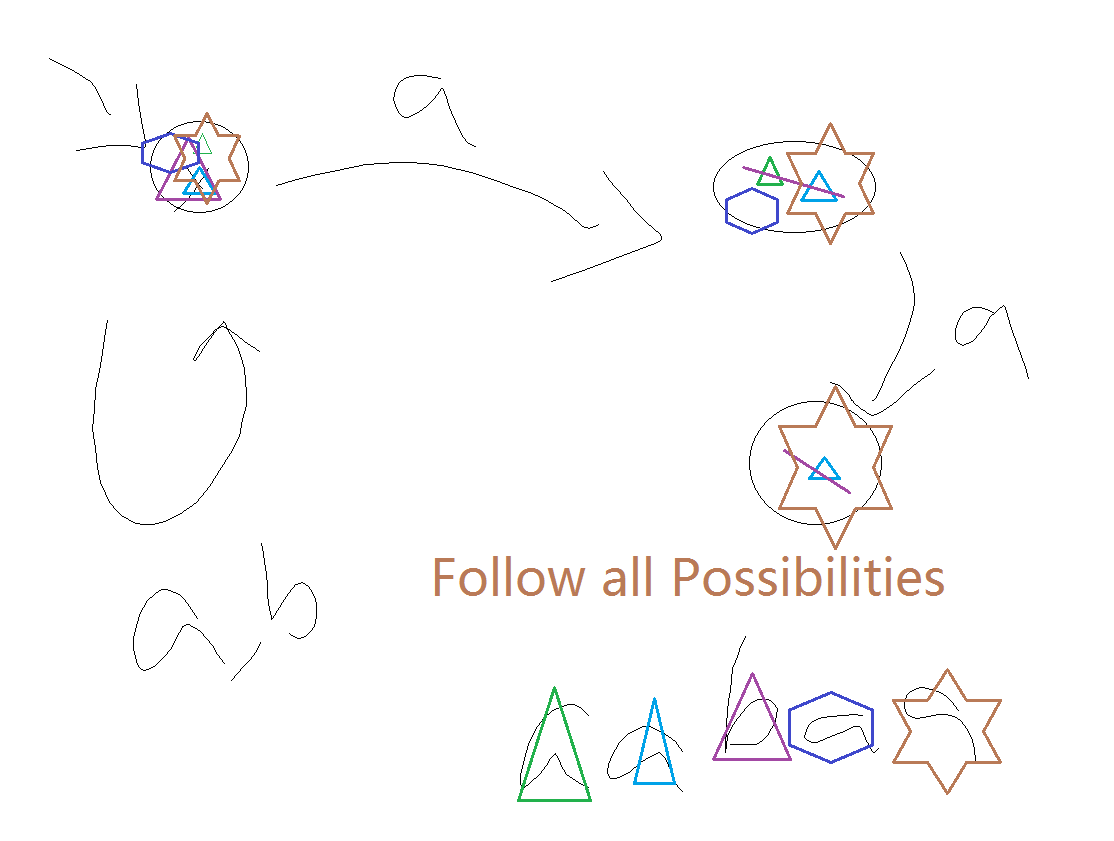
****

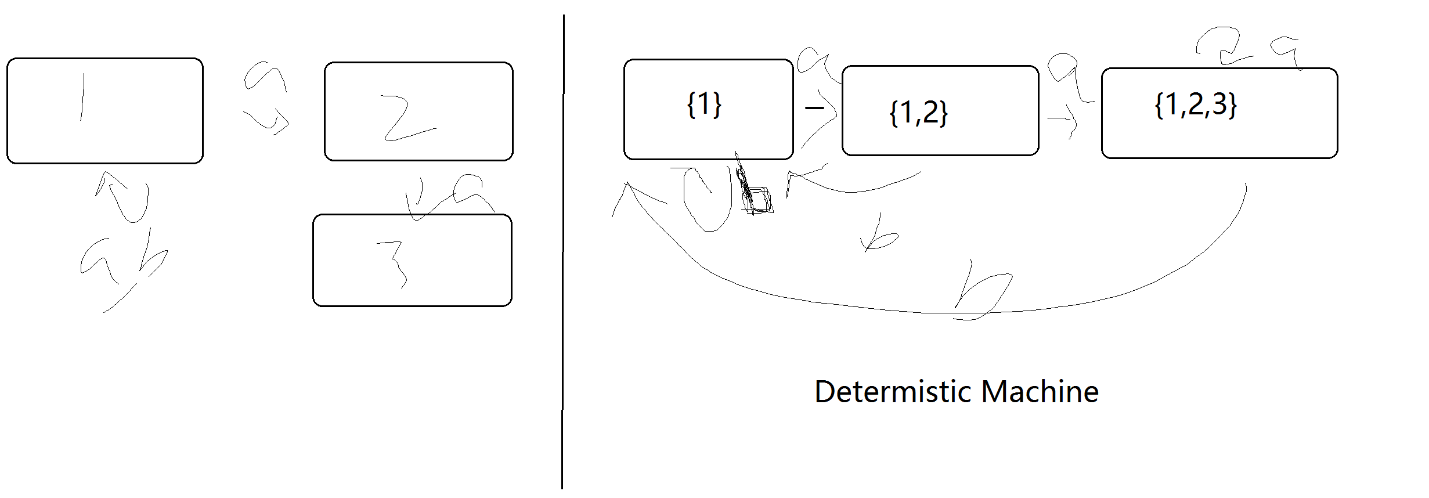
R1 . r2 L(r1,r2)= L(r1)L(r2)



Get rid of starting and ending states for either one, and add new transitions to “gluing” both M together

A finite automaton is deterministic if every state has at most one transition for every symbol in the alphabet.





Deterministic machine:

1) is regular if there is a regular expression with

There is a finite automaton m with L (M)= A

There is a deterministic finite automaton with L(M) = A

There exists a regular language where the smallest non-deterministic FA has N states but the smallest deterministic FA has 2N states.