CS3102 Theory of Computation

Closure Properties of Regular Languages

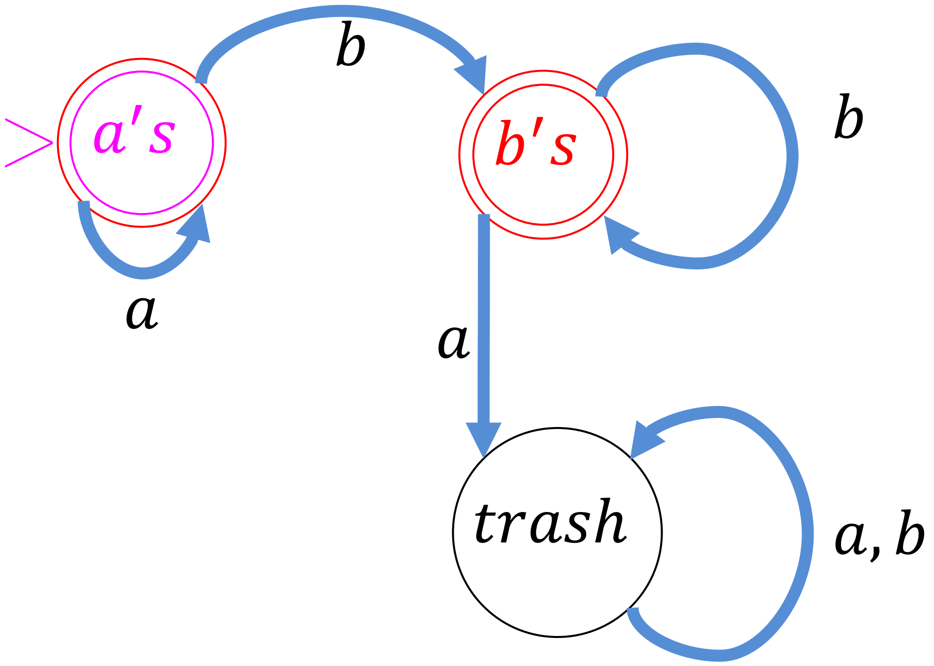
- Complement
- Intersection
- Union
- Difference
- Reversal
- Concatenation

Closed Under reversal

- Show that the regular languages are closed under reversal
- L^R is the language of all strings from L backwards
  - $L = \{s \in \{a, b\}^* \mid \text{all a's come before all b's}\}$
  - $L^R = \{s \in \{a, b\}^* \mid \text{all b's come before all a's}\}$

Let's Draw it!

$\{s \in \{a,b\}^* \mid \text{all a's come before all b's}\}$

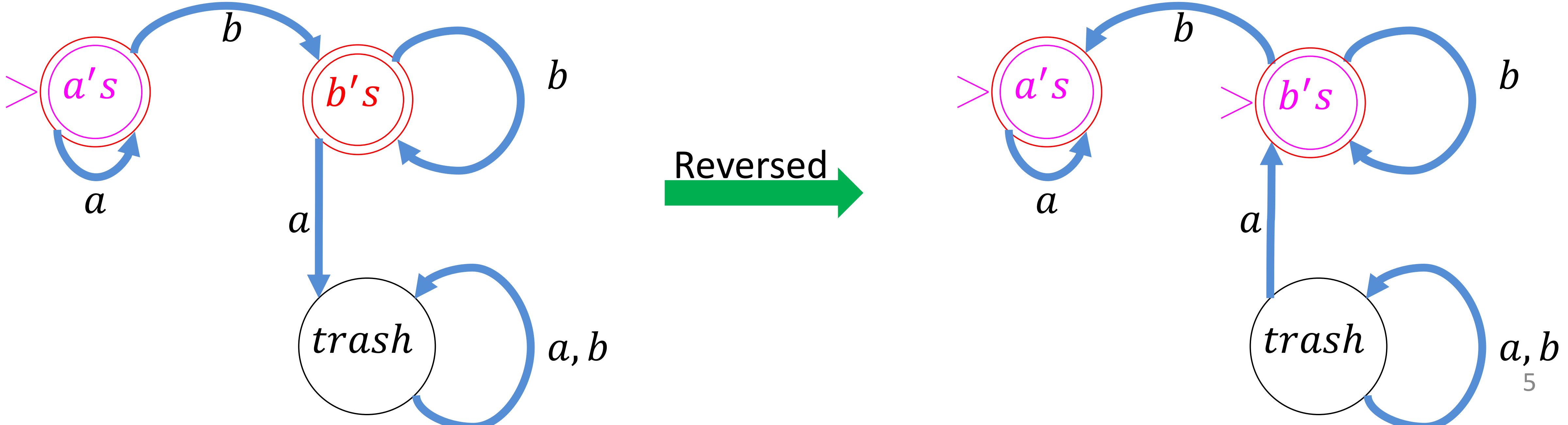


How to do reversal

- “reverse” the automaton

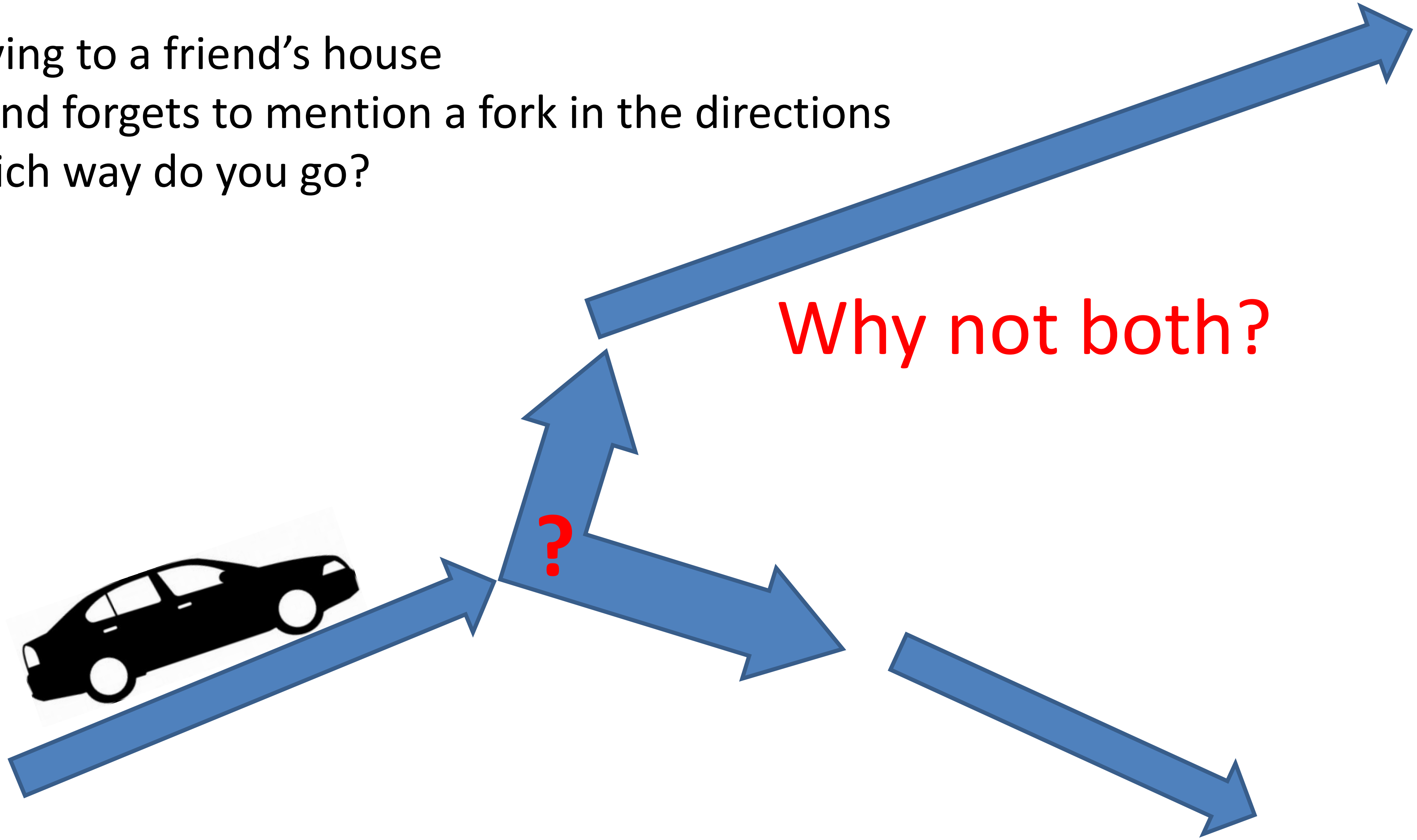
Problem(s)?

- Final states become start states
- Start state becomes final
- Reverse direction of all arrows



Nondeterminism

Driving to a friend's house
Friend forgets to mention a fork in the directions
Which way do you go?



Nondeterminism in computation

- Your computer/machine/algorithm can "be in two places at once"
- Java example:

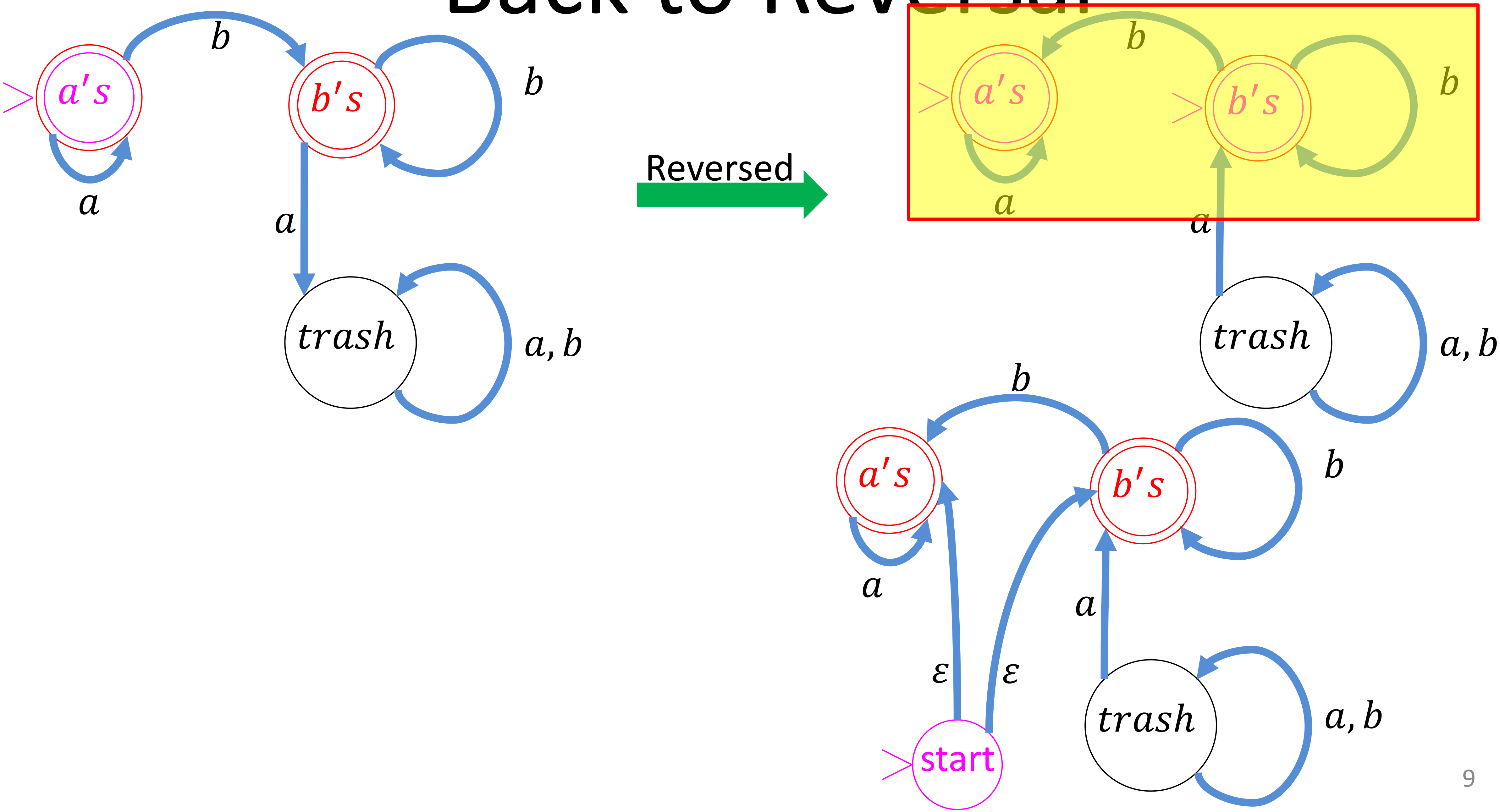
```
public static boolean isComposite(int n){
    for(int i = 2; i < n; i++){
        if(n % i == 0){
            return true;
        }
    }
    return false;
}
```

- We don't know which value might divide n, so we try each possibility one at a time
- Nondeterministic approach: let i take all values at once, return true if any divide n

Nondeterminism in Automata

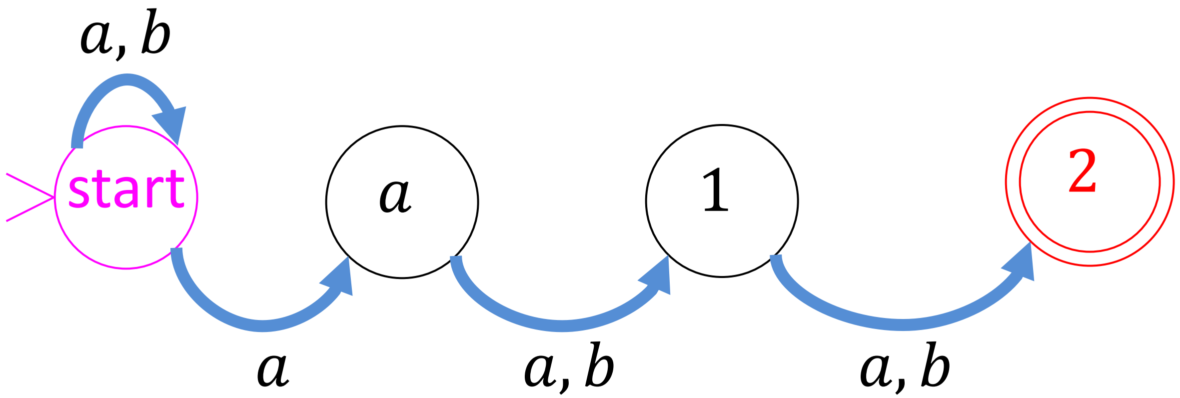
- Your machine can be in multiple states at once
- Accepts if any if the states it ends in are accepting states
- Relax restrictions:
  - Exactly one transition per symbol (can make multiple without consuming a symbol)
  - There must be exactly one outgoing transition for each symbol for every state (will allow 0 to many of them)
- Keep restriction:
  - One start state

Back to Reversal



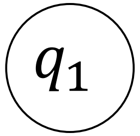
a is Third From Last

- Draw a nondeterministic finite state automaton (NFA) for the language of all strings where their third from last character is an a.

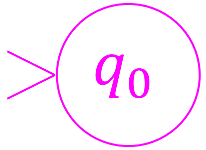


Nondeterministic Finite State Automata

- Basic idea: a NFA is a "machine" that changes states while processing symbols, one at a time.
- Finite set of states: $Q = \{q_0, q_1, \dots q_7\}$

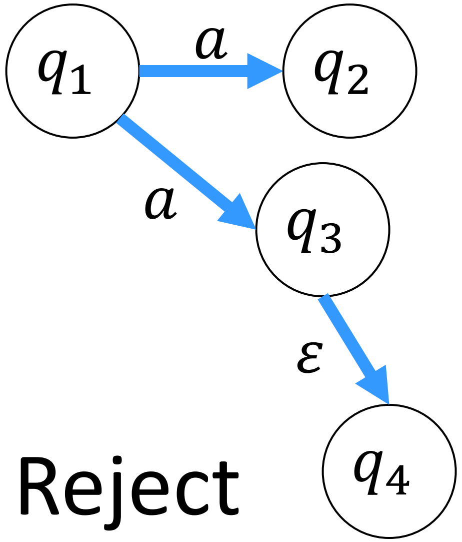


- Transition function: $\delta: 2^Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$
- Initial state: $q_0 \in Q$



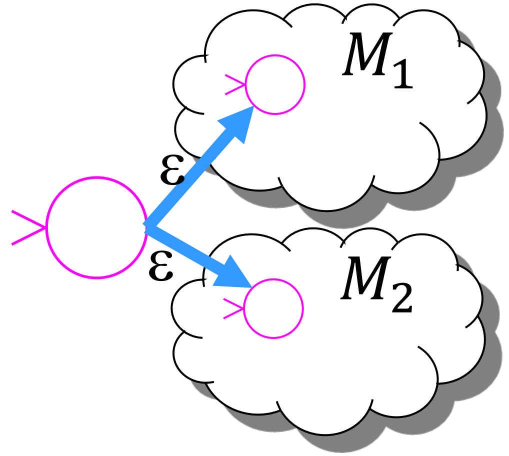
- Final states: $F \subseteq Q$

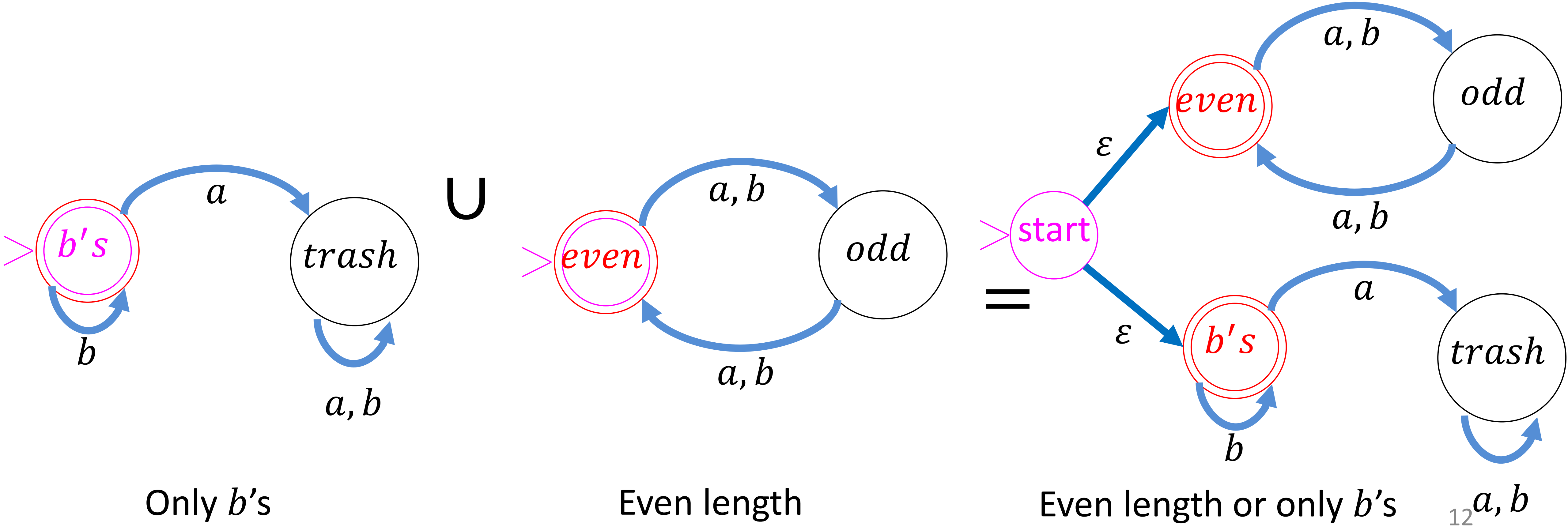
- Finite state automaton is $M = (Q, \Sigma, \delta, q_0, F)$
- Accept if any states we end in are Final, otherwise Reject only when none of the states are final
- If no transition defined, that "branch" rejects



Union using Non-Determinism

Introduce a new start state, ε transition to original start states.





Only b's

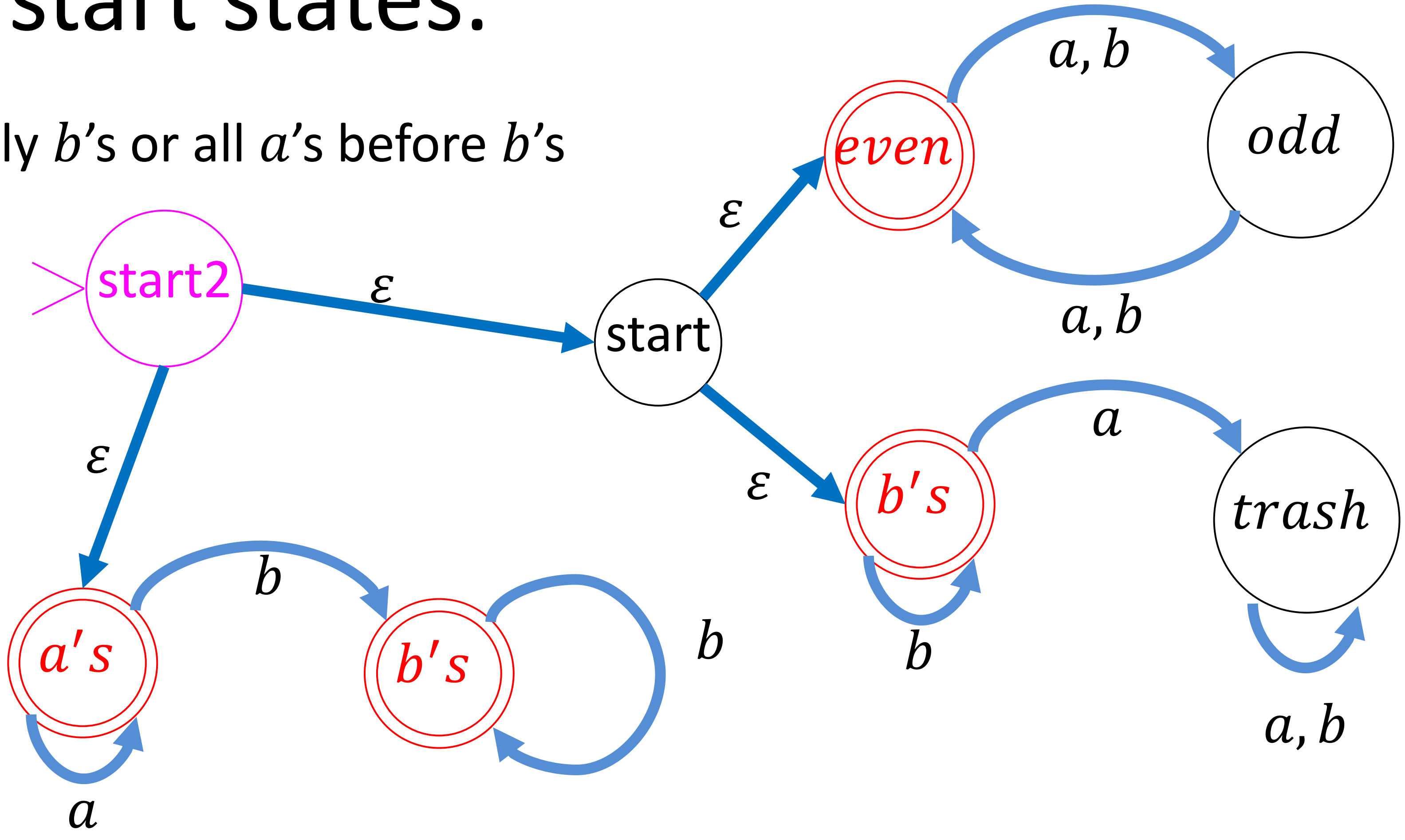
Even length

Even length or only b's

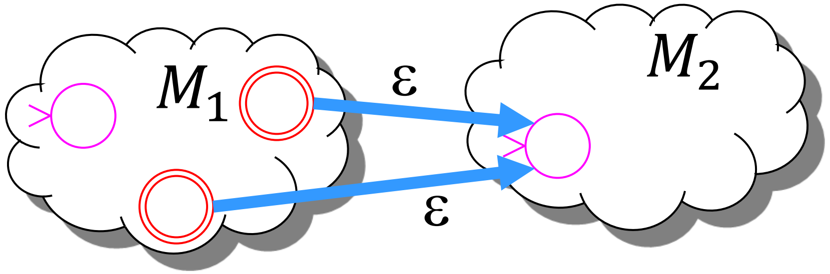
Union using Non-Determinism

Introduce a new start state, ε transition to original start states.

Even length or only b's or all a's before b's

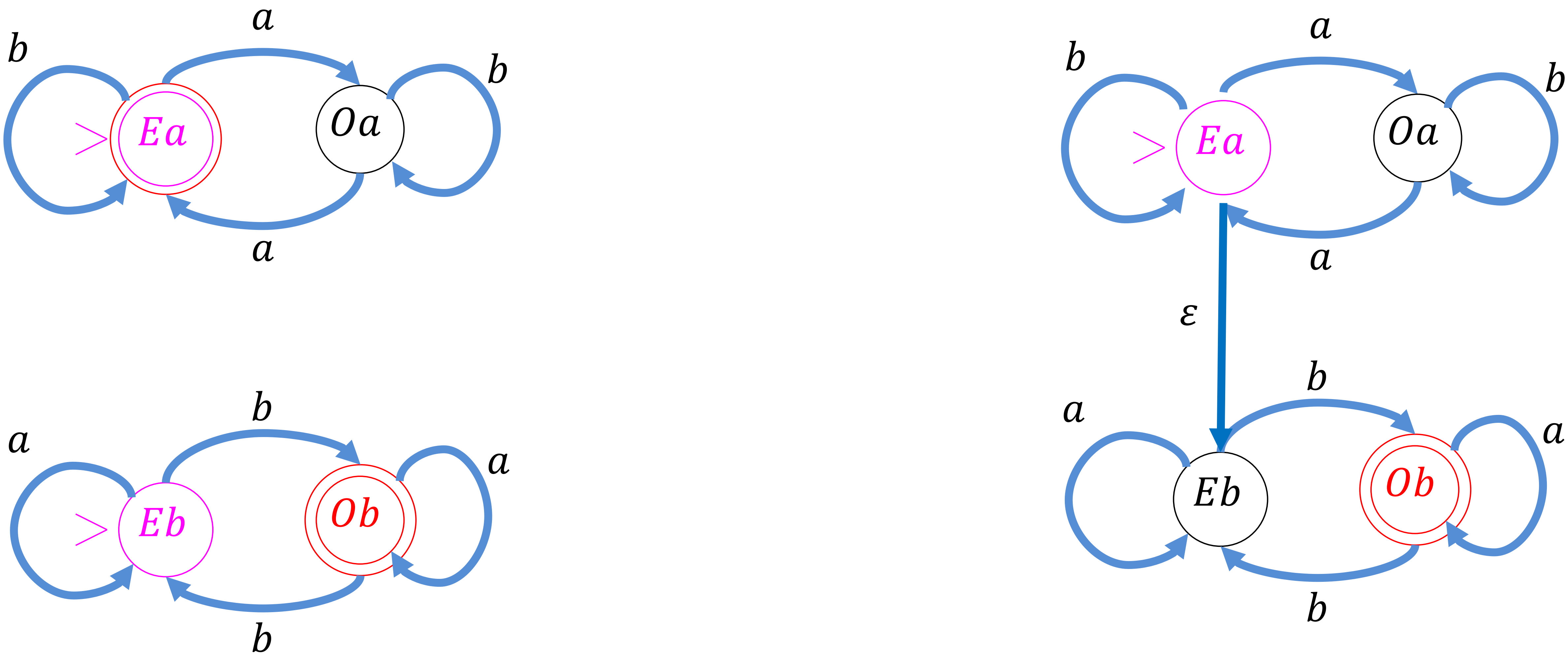


Closed Under Concatenation

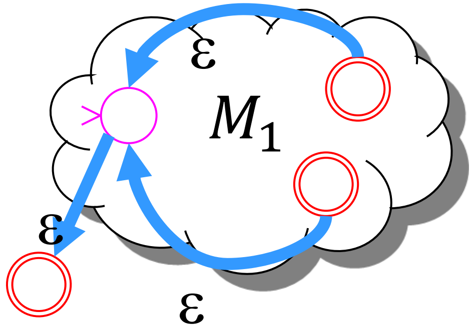


- Show that the regular languages are closed under Concatenation
- Given we have FSAs for L_1 and L_2, find one for $L_1 \cdot L_2$
- Idea: when processing a string, any time we see the current prefix is from L_1, check if suffix is from L_2.
- The prefix is from L_1 whenever M_1 is in a final state
- Add ε-transitions from each final state of M_1 to start state of M_2
- Make all final states from M_1 non-final

Example: EvenA·OddB

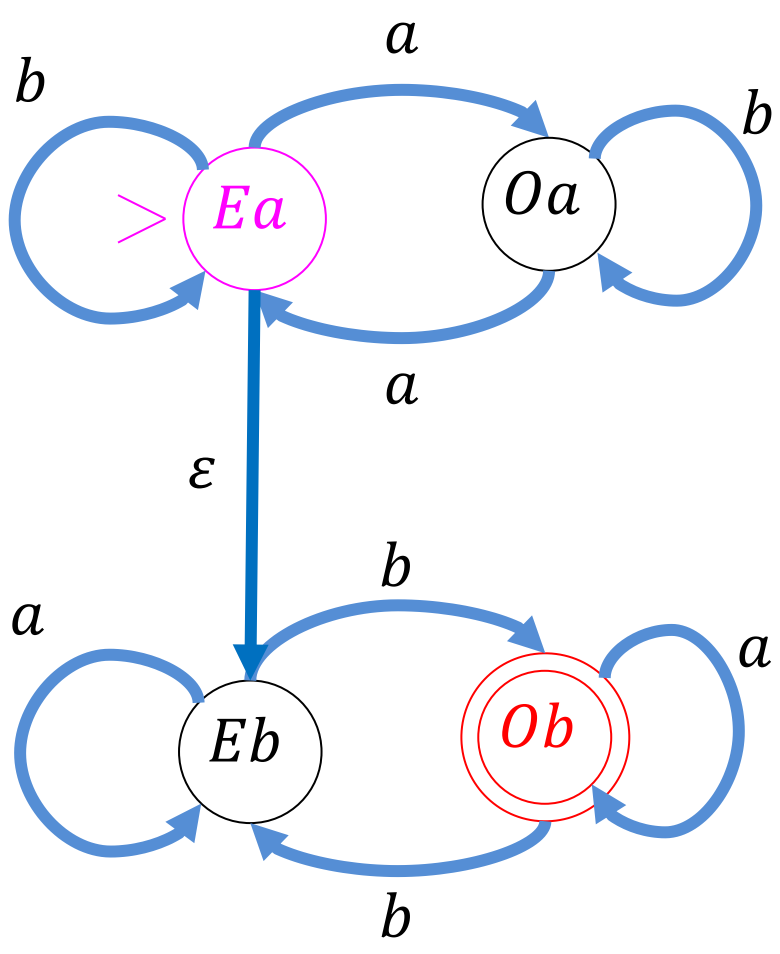


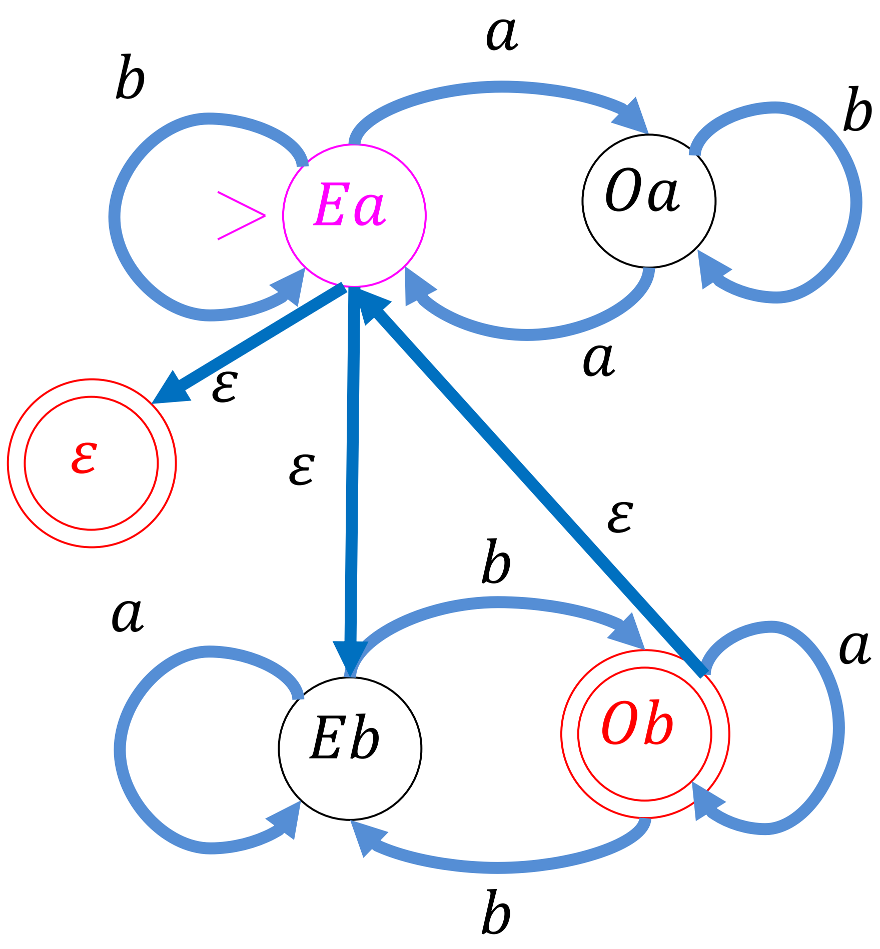
Closed under Kleene Star



- Once you have a prefix in the language, you may or may not start another one.
- Always contains the empty string
- Given a FSA for L_1, give one for L_1^*
- Draw ε-transitions from every final state to the start state.
- Add new final state for ε, with an ε-transition from the start state

EvenAOddB*



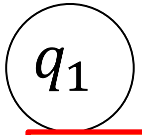


NFAs = DFAs

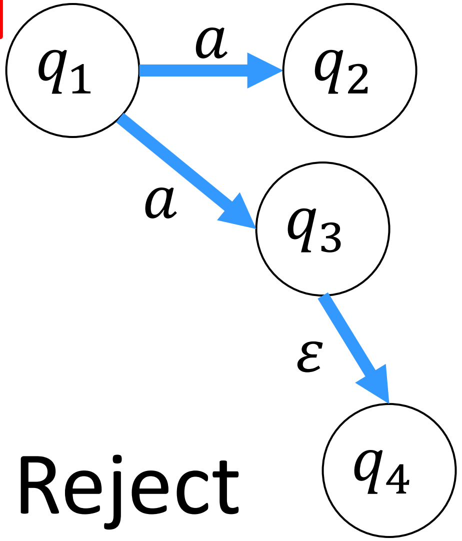
- We can convert any NFA into a DFA
- Powerset Construction
  - Idea: Make a new DFA whose states each represent a subset of states from the original machine.

Nondeterministic Finite State Automata

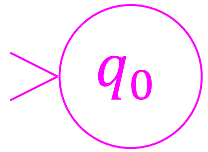
- Basic idea: a NFA is a "machine" that changes states while processing symbols, one at a time.
- Finite set of states: $Q = \{q_0, q_1, \dots q_7\}$



- Transition function: $\delta: \boxed{2^Q} \times (\Sigma \cup \{\varepsilon\}) \rightarrow \boxed{2^Q}$



- Initial state: $q_0 \in Q$



- Final states: $F \subseteq Q$

- Finite state automaton is $M = (Q, \Sigma, \delta, q_0, F)$
- Accept if any states we end in are Final, otherwise Reject only when none of the states are final
- If no transition defined, that "branch" rejects

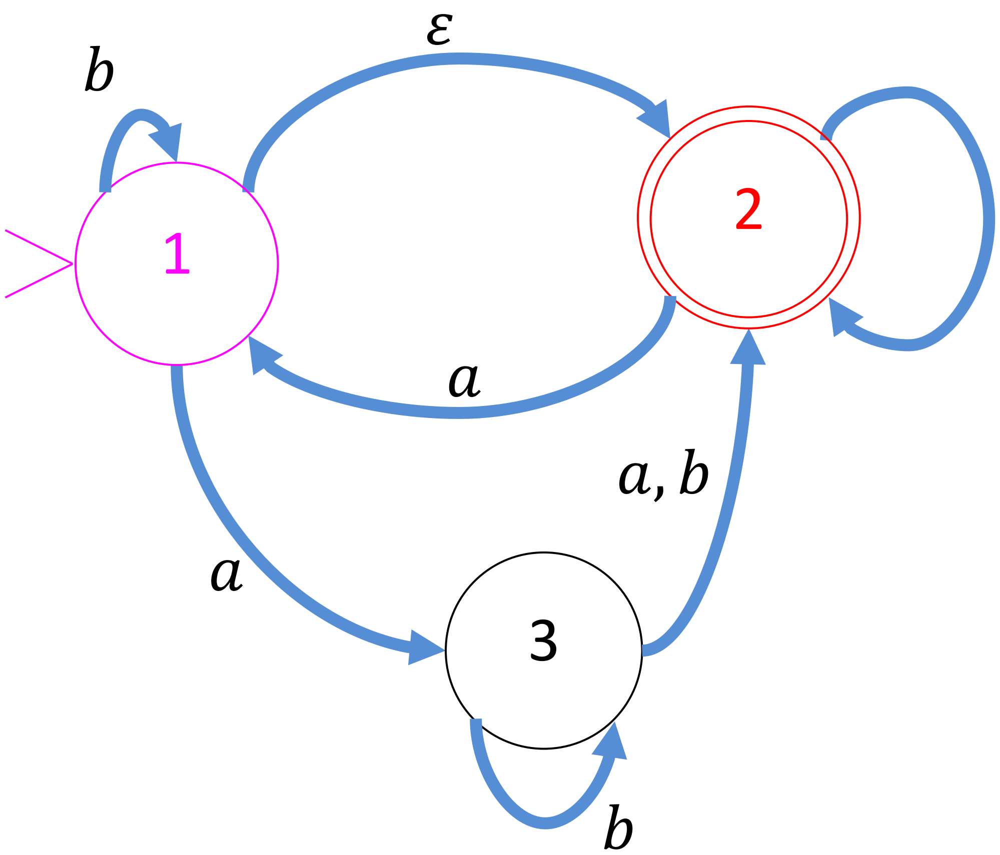
Powerset Construction

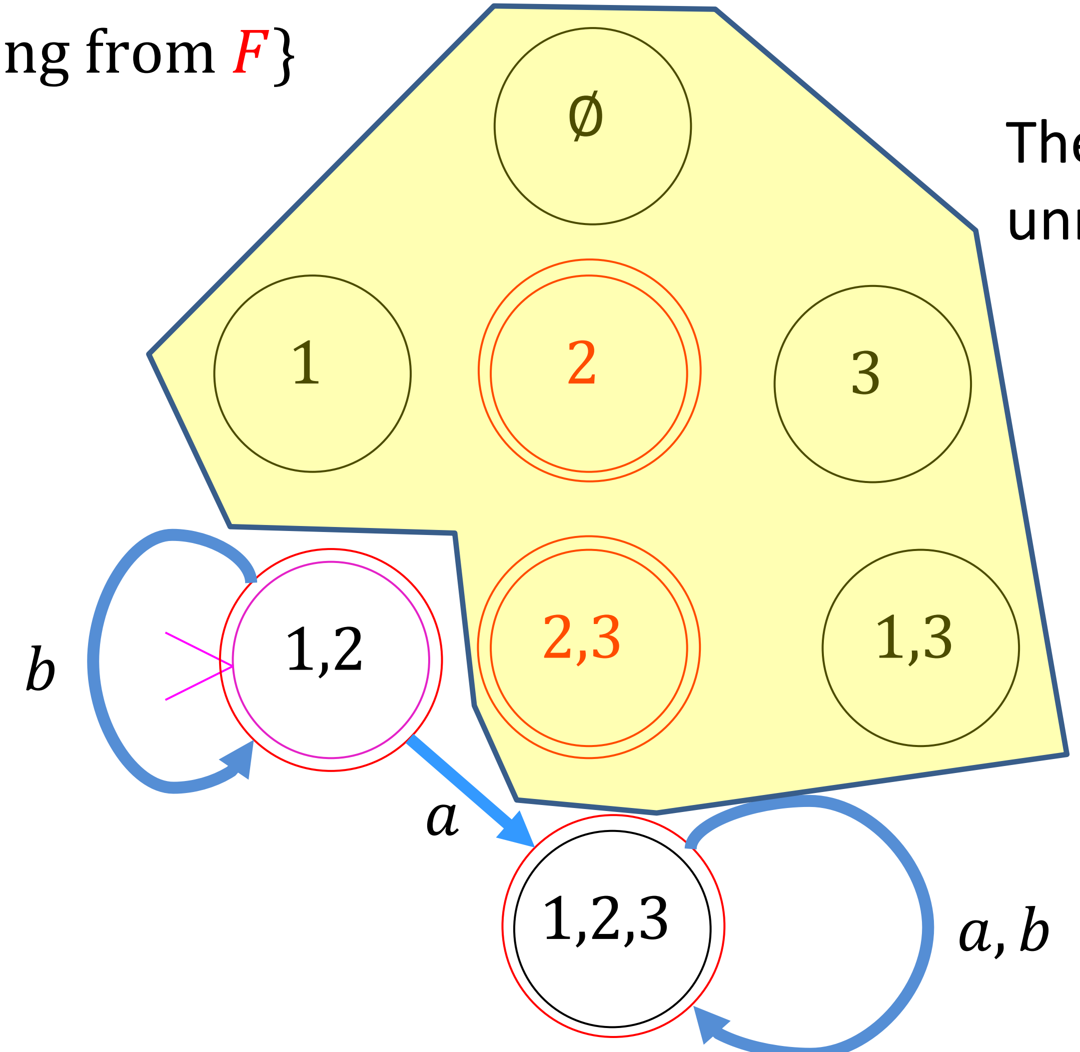
$Q_{power} = 2^Q$

$\delta_{power}(q_1, \sigma) = \bigcup_{q \in q_1} \delta(q, \sigma)$ (all states that any active state transitions to)

$q_{o,power}$ = everything reachable from q_0

$F_{power} = \{\text{any state containing something from } F\}$





These states are unreachable.

Regular Expressions

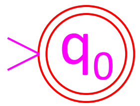
- A way to describe strings using operations on characters
- Pieces:
  - Literals: Characters from Σ, ε All finite languages are regular!
  - Concatenation $(R_1 R_2)$ Preserves Regularity!
  - Union $(R_1 + R_2)$ Preserves Regularity!
  - Kleene Star (R^*) Preserves Regularity!
    - One or more (R^+), same as RR^* Preserves Regularity!
- Example: $aa(a+b)^*bb$
  - Any string that starts with 2 a's and ends with 2 b's
- Why can we only get regular languages?

Regex to NFA

All literal regular expressions have an FSA:

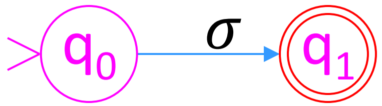
ε

Empty string



$\sigma \in \Sigma$

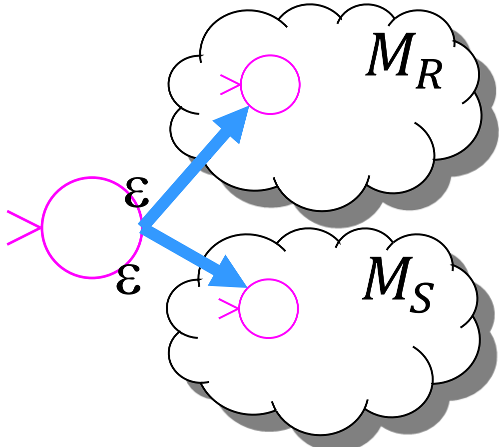
Literal characters



If regular expressions R and S have FSAs, then so do:

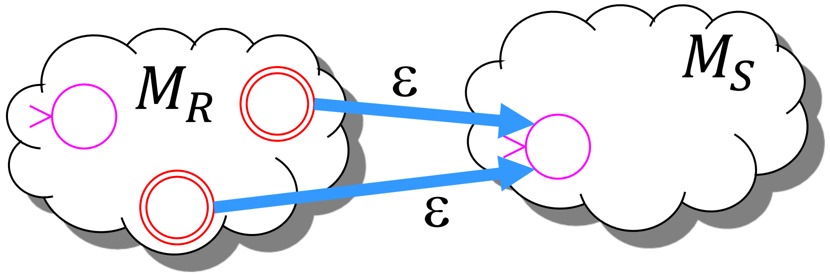
$(R + S)$

union



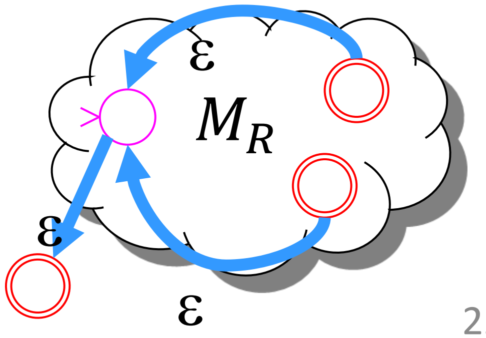
RS

concatenation



R^*

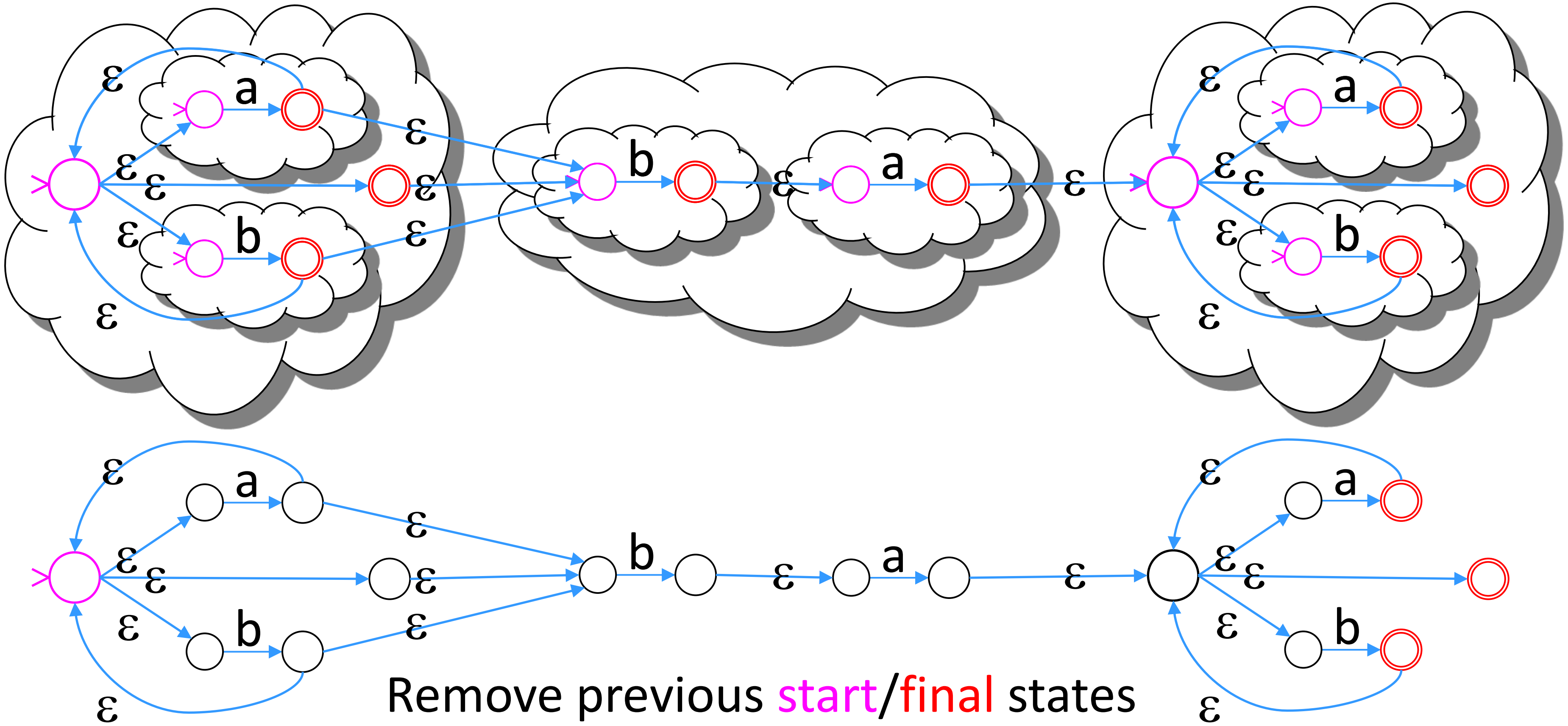
Kleene closure



Regex to NFA

Ex: all strings over $\{a, b\}$ where there is a "b" preceding an "a"

$$(a+b)^*ba(a+b)^*$$



Remove previous start/final states

NFA to Regex