CS3102 Theory of Computation

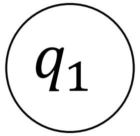
Closure

Which of the following sets are closed under the provided operation? If closed, argue why. If not, give a counterexample

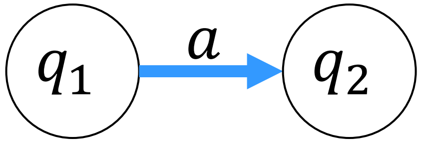
1. $\mathbb{Z}$, Multiplication by 0
2. $\mathbb{N}$, subtraction
3. $\mathbb{Q}$, squaring
4. $\mathbb{Q}$, square-root
5. $\{L \subseteq \Sigma^* | L \text{ is finite}\}$, Kleene Star
6. $\{\varepsilon, ab, aabb, aaabbb, \dots\}$, concatenation

Finite State Automata

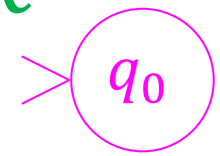
- Basic idea: a FA is a "machine" that changes states while processing symbols, one at a time.
- Finite set of states: $Q = \{q_0, q_1, \ldots q_7\}$



- Transition function: $\delta: Q \times \Sigma \to Q$



- Initial state: $q_0 \in Q$



- Final states: $F \subseteq Q$

- Finite state automaton is $M = (Q, \Sigma, \delta, q_0, F)$
- Accept if we end in a Final state, otherwise Reject

TripleA

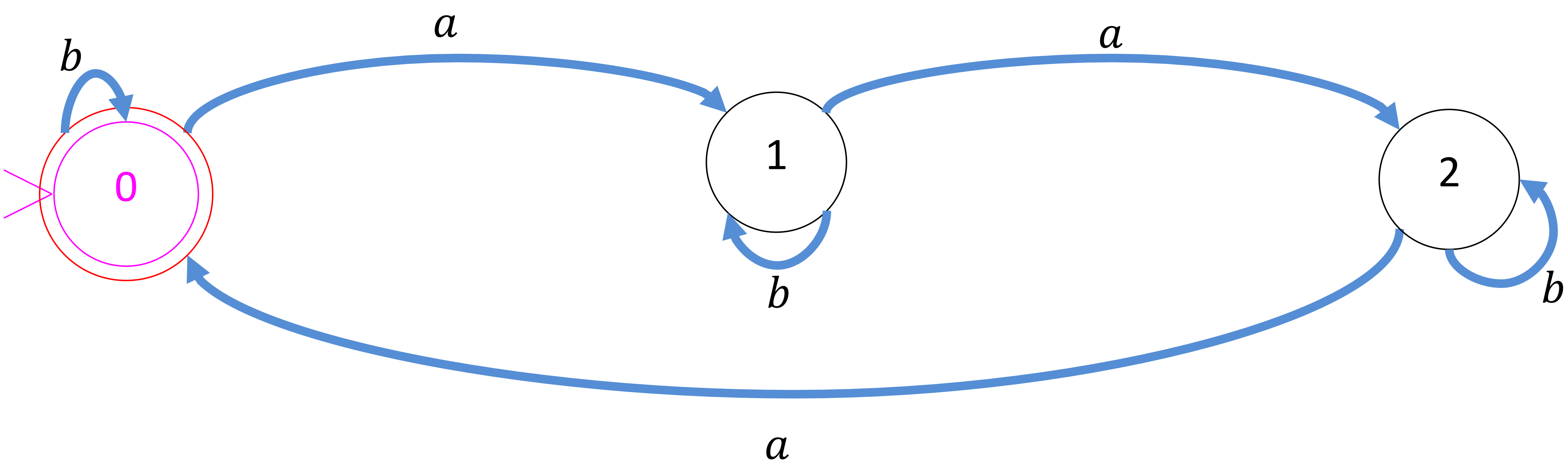
- Let’s make a finite state automaton which accepts strings where the number of a’s is a multiple of 3
- It should accept:
  - $b, aaa, abaa, \ldots$
- It should reject:
  - $a, ab, baa, aba, aaaabbb, \ldots$

TripleA using FSA

1. What's our alphabet? (pick Σ)
2. What should our states be? (pick Q)
3. Which states are the accept states? (pick $\mathbf{F}$)
4. Which state is the start state? (pick q_0)
5. How should we transition? (pick δ)

TripleA

Strings with a multiple of 3 many a's



Take-aways

- For a FSA M, the language of M (denoted $L(M)$) refers to the set of strings accepted by the machine
  - $L(M) = \{s \in \Sigma^* | M \text{ accepts } s\}$
- The set of all languages decided by some FSA is call the **Regular Languages**
  - Equivalent to the languages describable by regular expressions
- A particular language decided by some FSA is called a **Regular Language**
- All regular languages can be decided by a Java program using only constant memory (relative to length of word)

Closure Properties

- A set is **closed** under an operation if applying that operation to members of the set results in a member of the set
  - Integers are closed under addition
  - Integers are not closed under division
  - Σ^* is closed under concatenation
  - The set of all languages are not closed under cross product

Closure Properties of Regular Languages

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

Closed under Complement

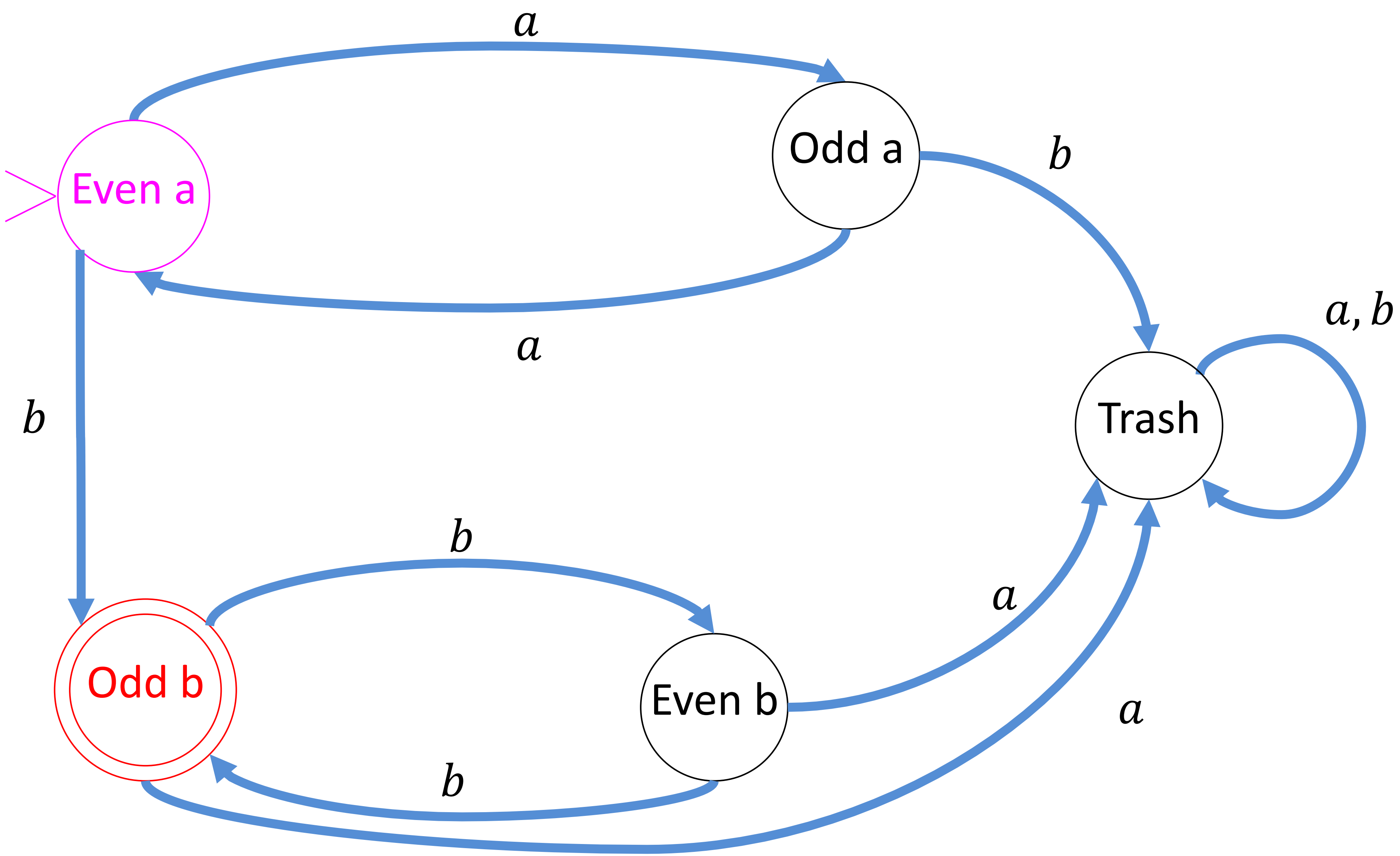
- If a language is regular then its complement is regular
- If a language has a FSA, it’s complement does as well
- If there is a FSA which accepts exactly the strings in the language, there is a FSA which accepts exactly the strings not in the language

Closed under Complement

- Idea: Every string ends in some state. If that was originally an accept state then reject, else accept.
- New final states are the old non-final states

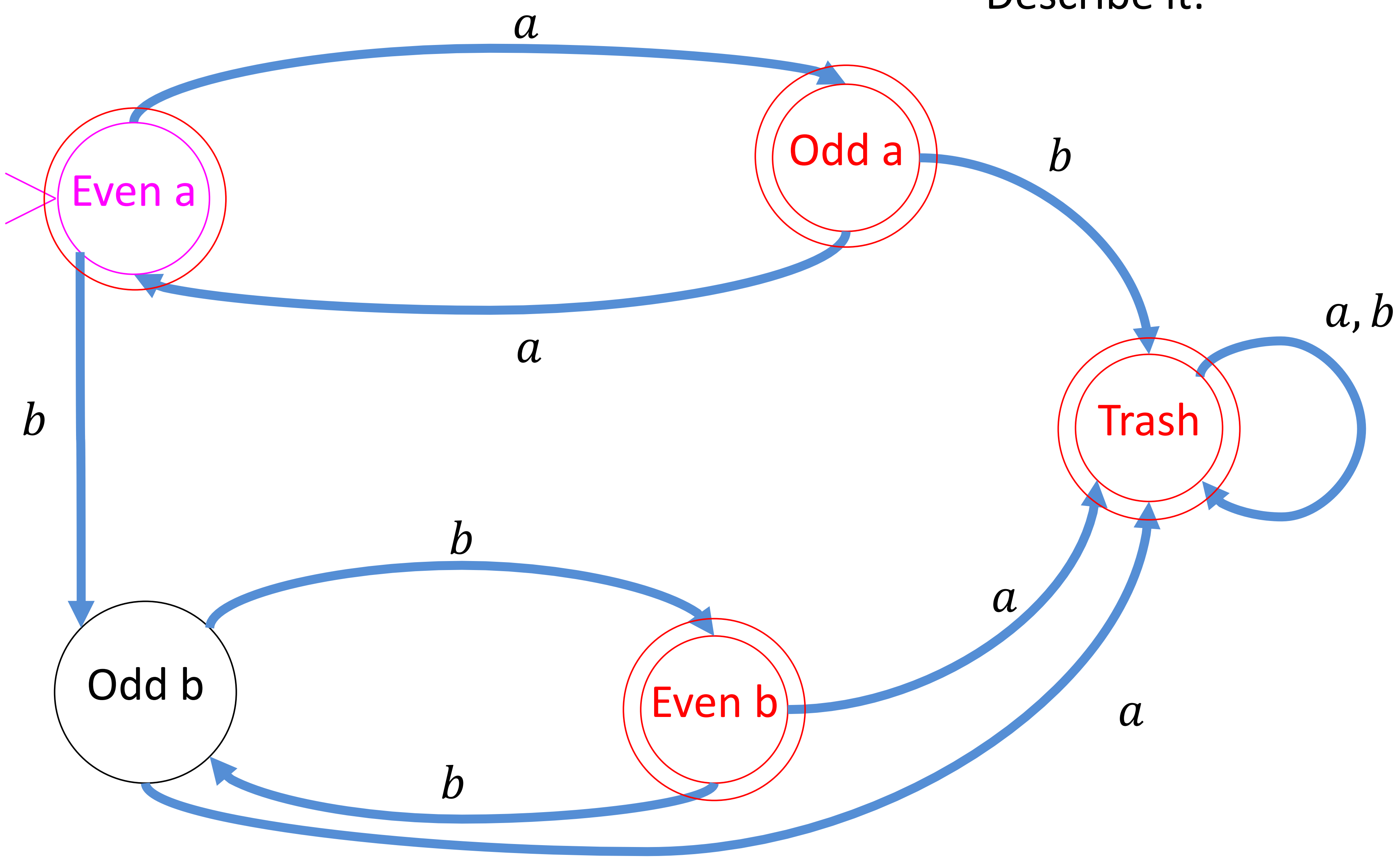
EvenAoddB

Strings with an even number of a's followed by an odd number of b's



Complement of EvenAoddB

Describe it:

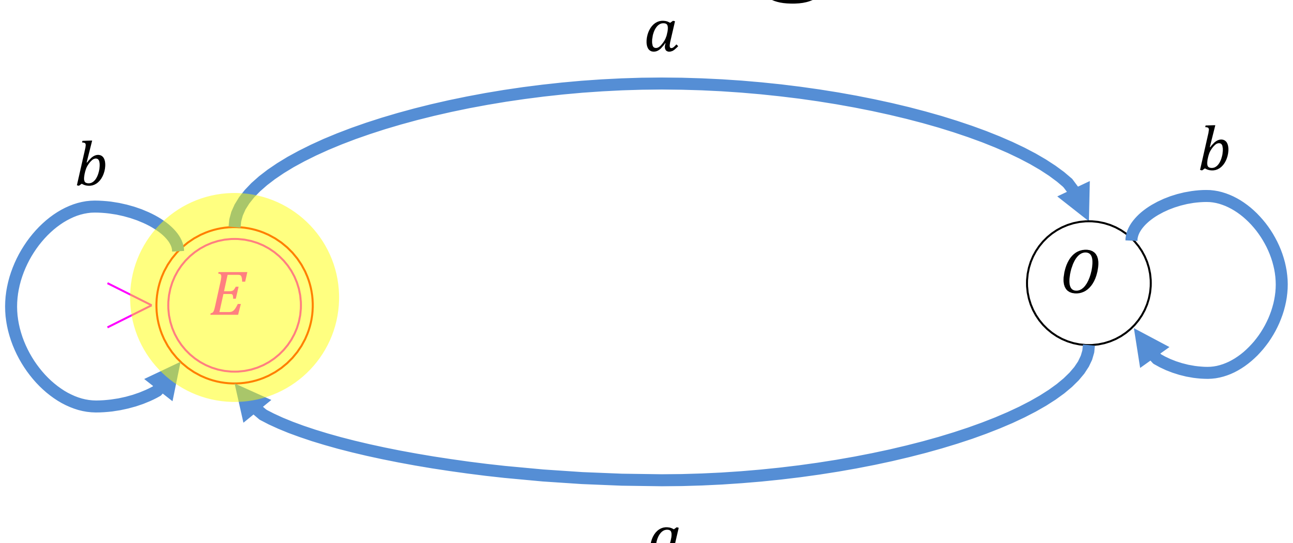


Closed under Intersection

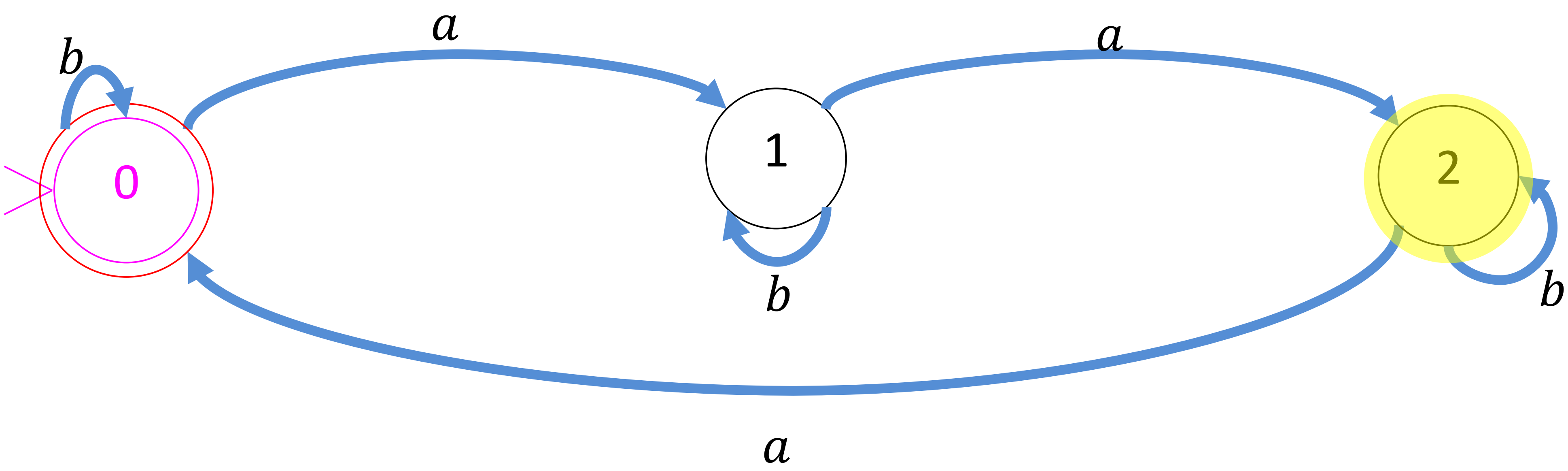
- Let’s find an automaton for TripleA∩EvenA
- This automaton should accept a given string if and only if BOTH these other automata accept
- We need to make one automaton that operates as if it was two
- Idea: This automaton’s states each represent a pair of states (one from each source automaton)

Running Both Machines

Input: $aaaaaaaa$



EvenA

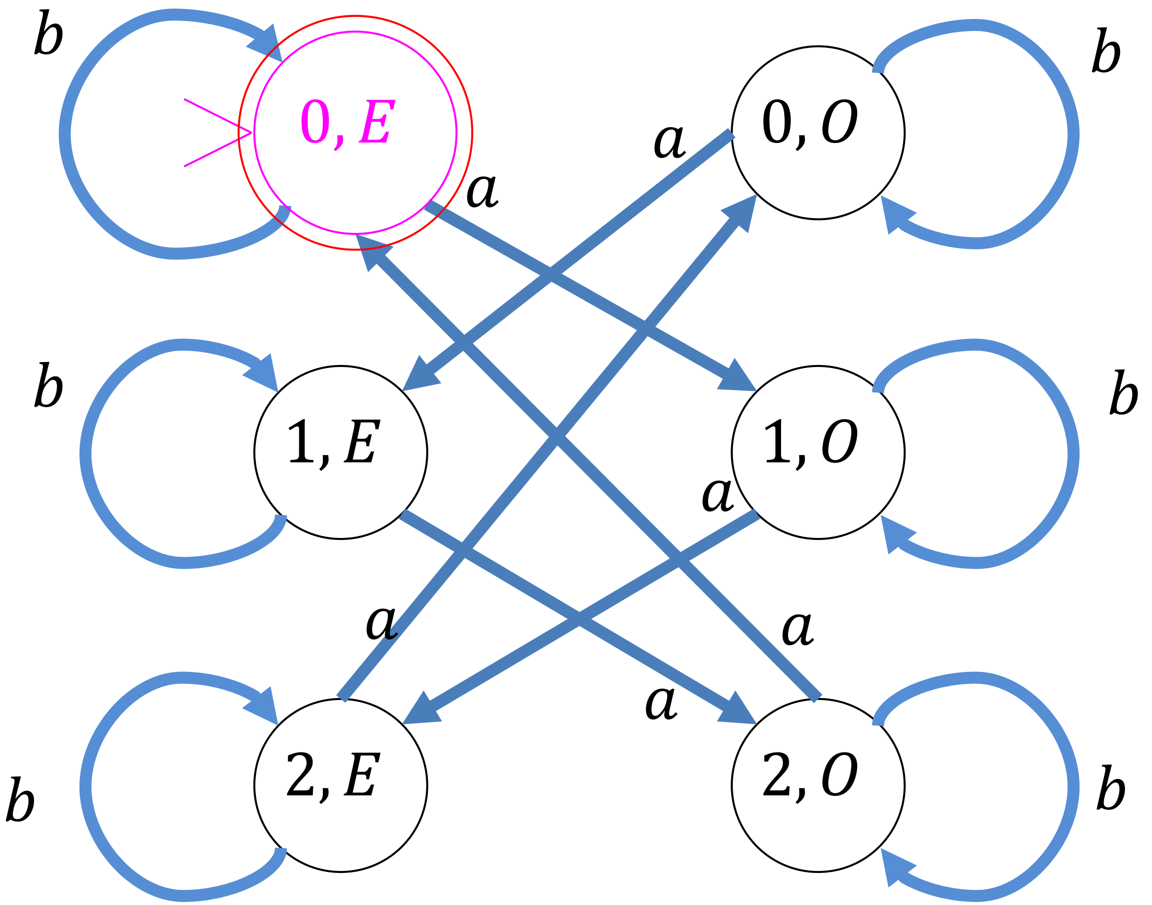


TripleA

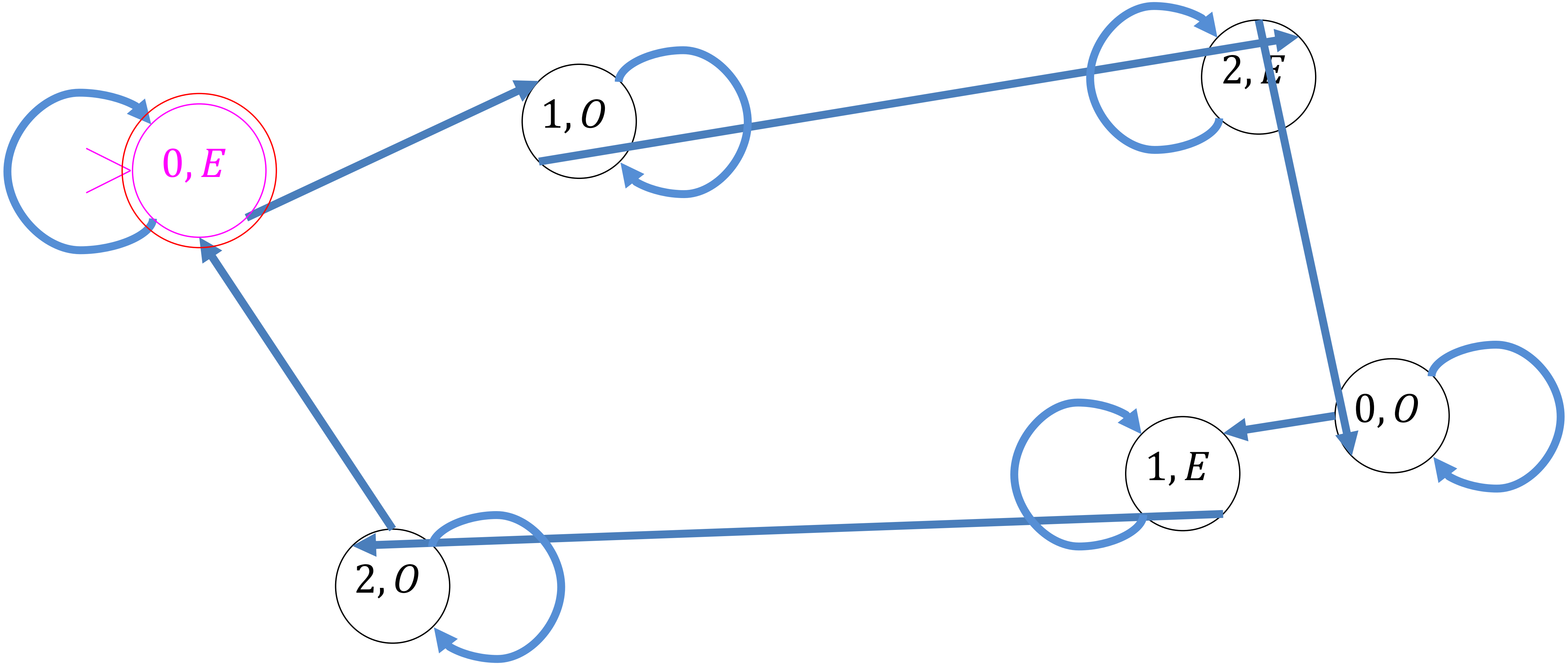
Now at the same time

- States: Pairs of states from source machines:
  - $\{(0, E), (1, E), (2, E), (0, O), (1, O), (2, O)\}$
- Start State: The one that's the pair of source starts
  - $(0, E)$
- Final States: Those pairs where both were final
  - $\{(0, E)\}$
- Transitions: One arrow represents transitioning in both machines

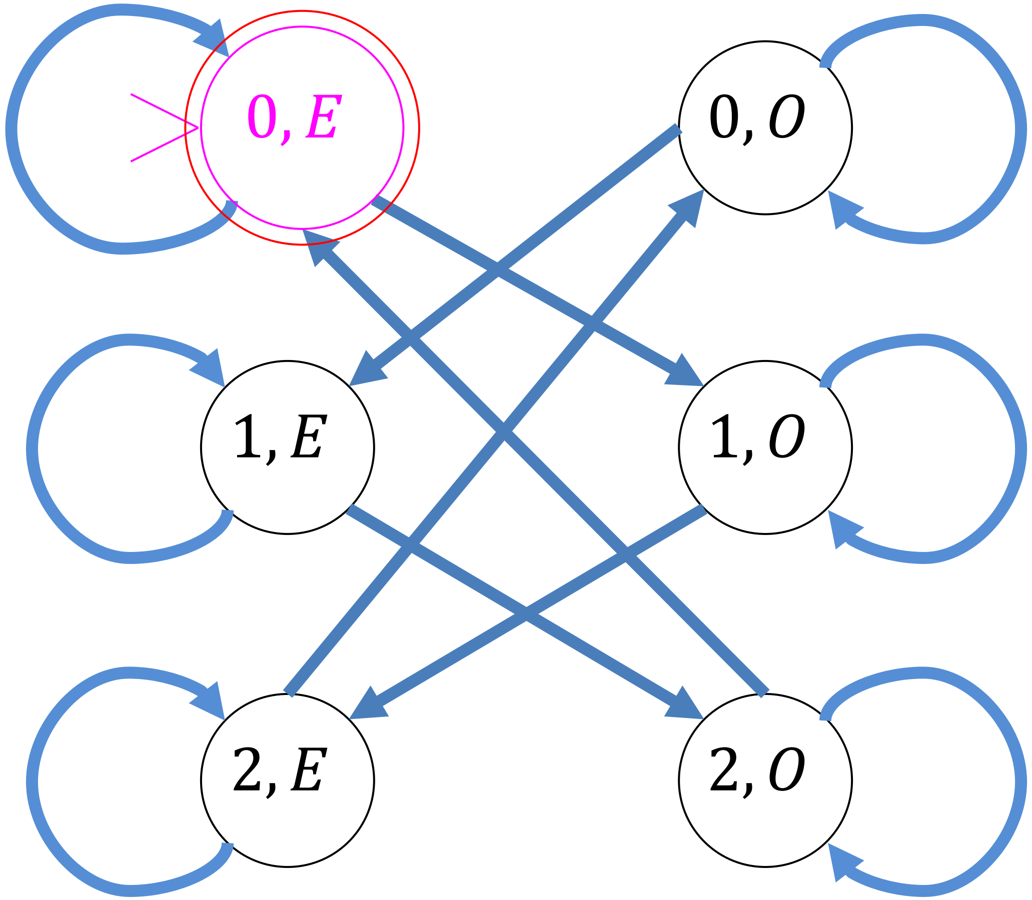
Let's Draw it!



Let's Draw it!



Let’s Draw it!



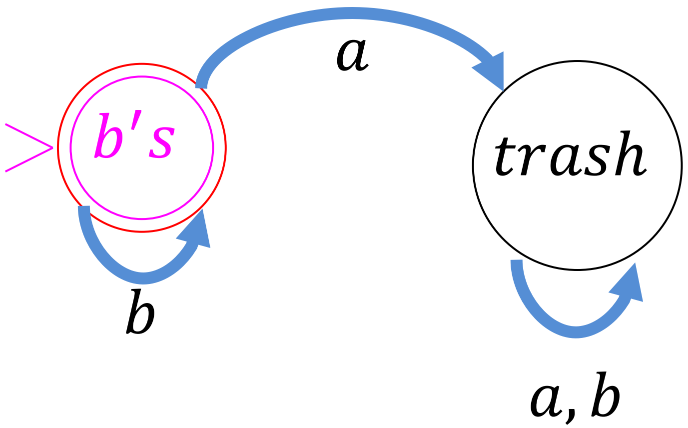
Cross Product Construction (intersection)

- Basic idea: a single FSA that operates the same as two would on the same input

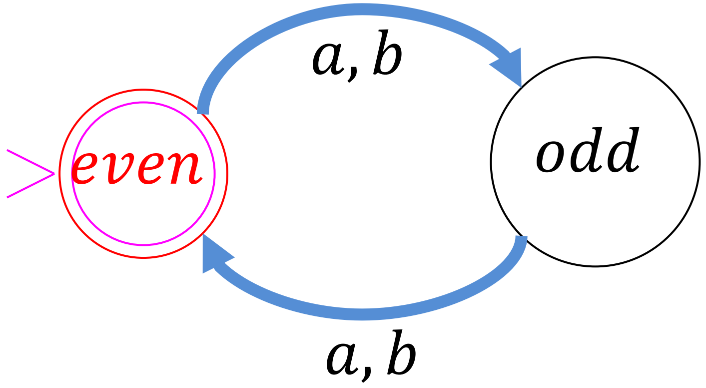
- To build $M_\times$ to simulate both: $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
- Finite set of states: $Q_\times = Q_1 \times Q_2$
- Transition function: $\delta_\times((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
- Initial state: $(q_{01}, q_{02}) \in Q_\times$
- Final states: $F_\times = F_1 \times F_2$
- Finite state automaton is $M_\times = (Q_\times, \Sigma, \delta_\times, (q_{01}, q_{02}), F_\times)$

B's only or even length

$\{b\}^* = \{\varepsilon, b, bb, bbb, \dots\}$



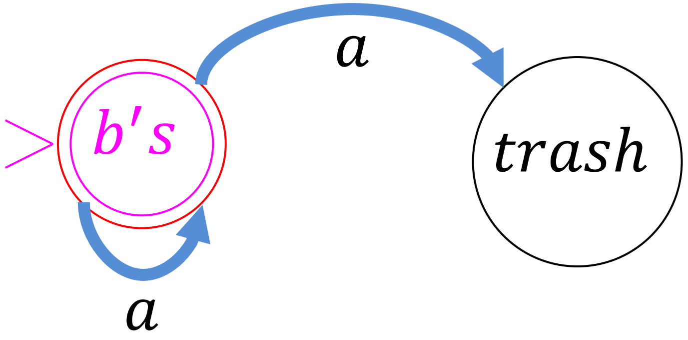
$\{aa, ab, ba, bb\}^*$



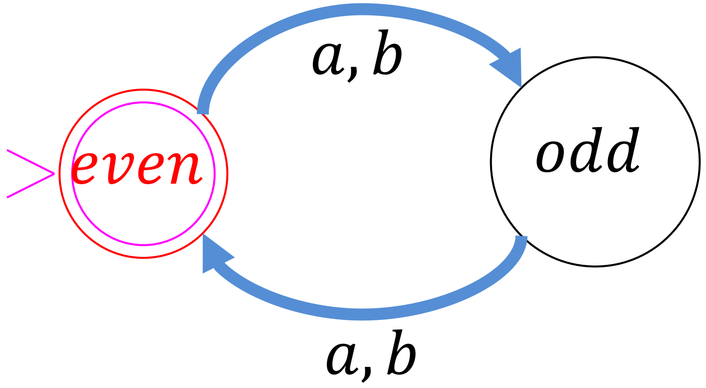
a

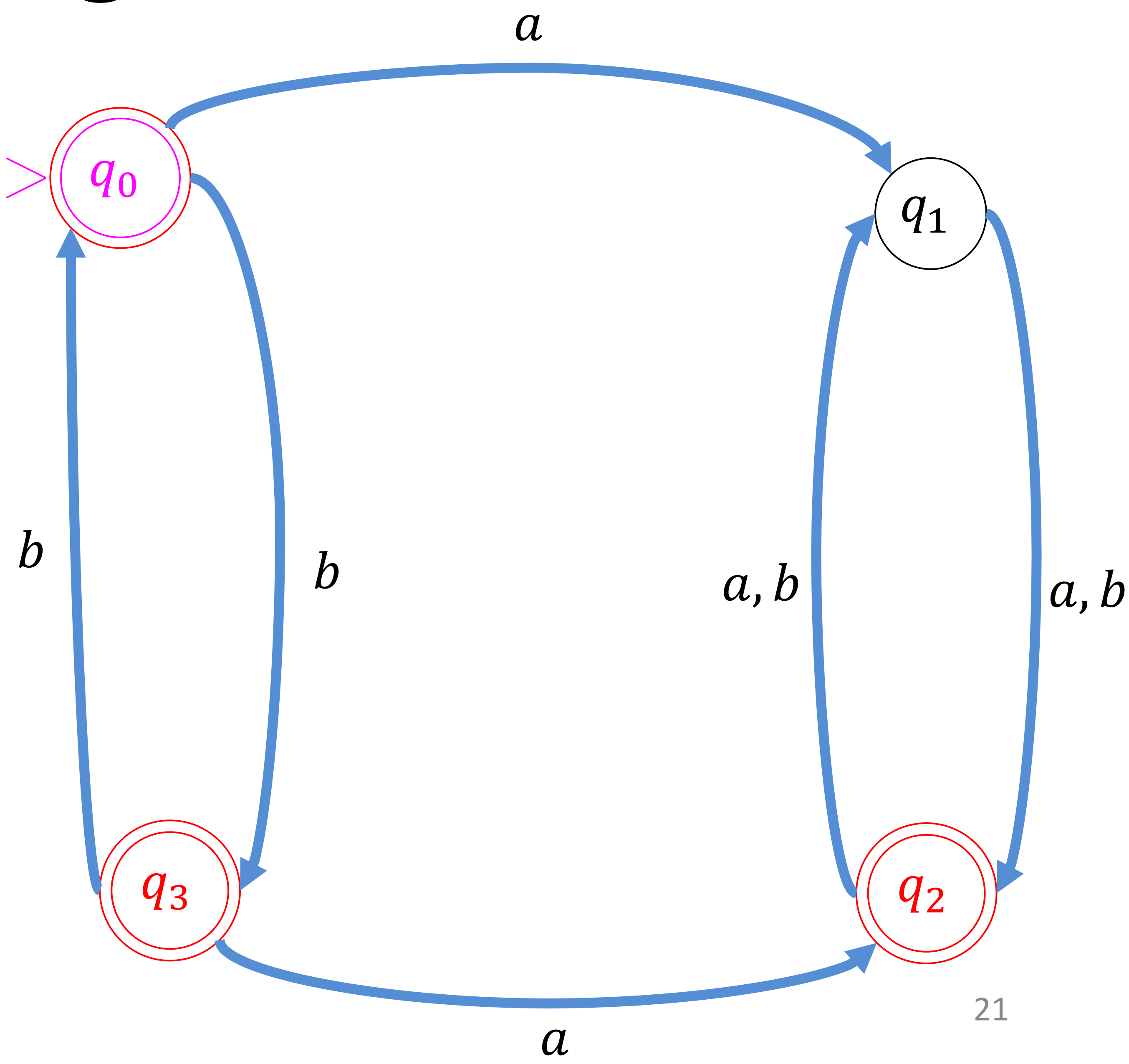
B’s only or even length

$\{b\}^* = \{\varepsilon, b, bb, bbb, \ldots\}$



$\{aa, ab, ba, bb\}^*$





Cross Product Construction (Union)

- Basic idea: a single FSA that operates the same as two would on the same input

- To build $M_\times$ to simulate both: $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
- Finite set of states: $Q_\times = Q_1 \times Q_2$
- Transition function: $\delta_\times((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
- Initial state: $(q_{01}, q_{02}) \in Q_\times$
- Final states: $F_\times = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- Finite state automaton is $M_\times = (Q_\times, \Sigma, \delta_\times, (q_{01}, q_{02}), F_\times)$

Cross Product Construction (Difference)

- Basic idea: a single FSA that operates the same as two would on the same input

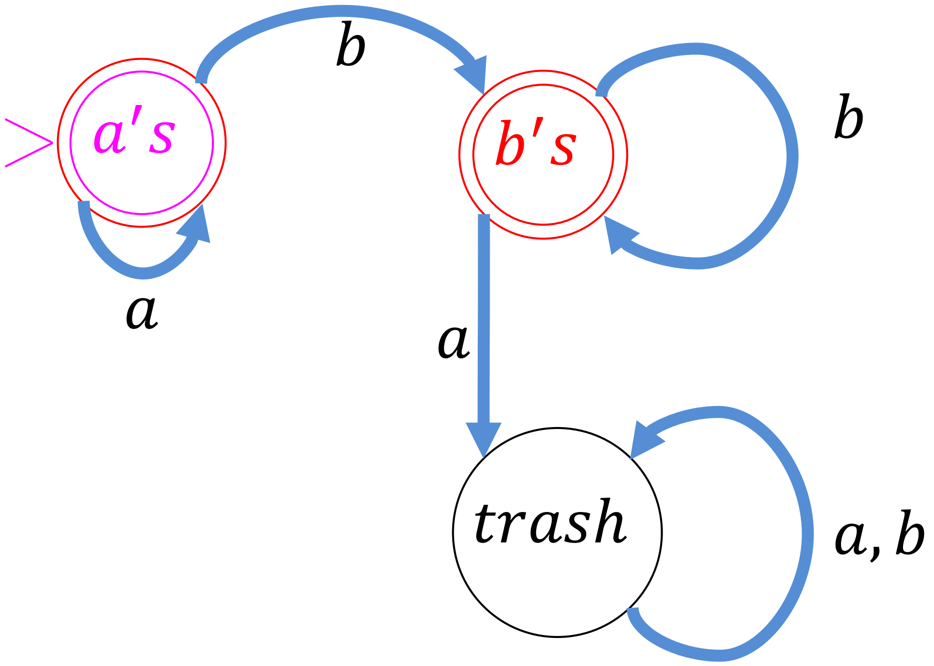
- To build $M_\times$ to simulate both: $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
- Finite set of states: $Q_\times = Q_1 \times Q_2$
- Transition function: $\delta_\times((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
- Initial state: $(q_{01}, q_{02}) \in Q_\times$
- Final states: $F_\times = F_1 \times (Q_2 - F_2)$
- Finite state automaton is $M_\times = (Q_\times, \Sigma, \delta_\times, (q_{01}, q_{02}), F_\times)$

Closed Under reversal

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

Let's Draw it!

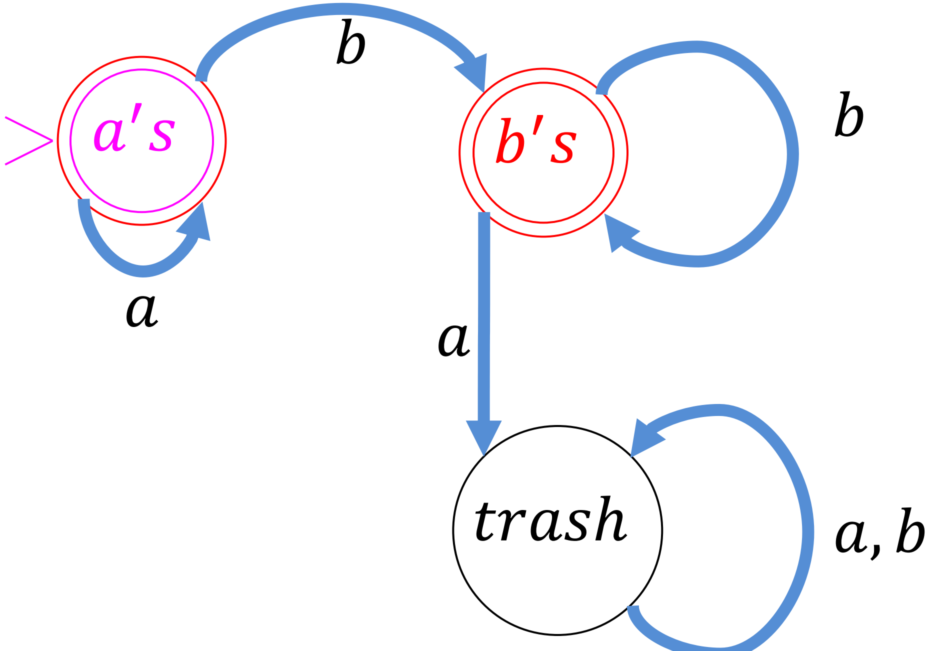
$\{s \in \{a, b\}^* \mid \text{all a's come before all b's}\}$



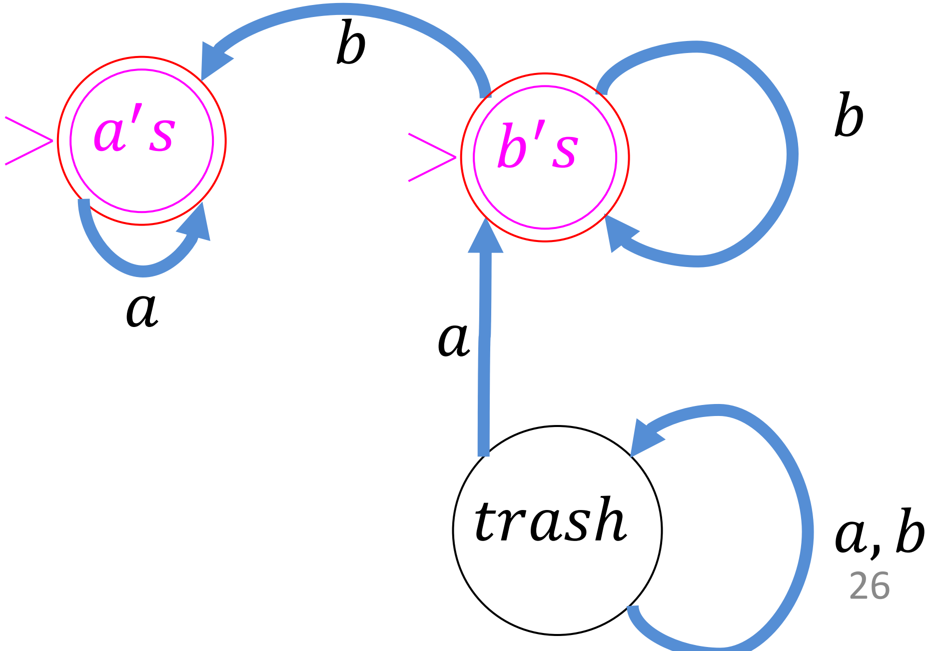
How to do reversal

- “reverse” the automaton
- Final states become start states
- Start state becomes final
- Reverse direction of all arrows

Problem(s)?



Reversed

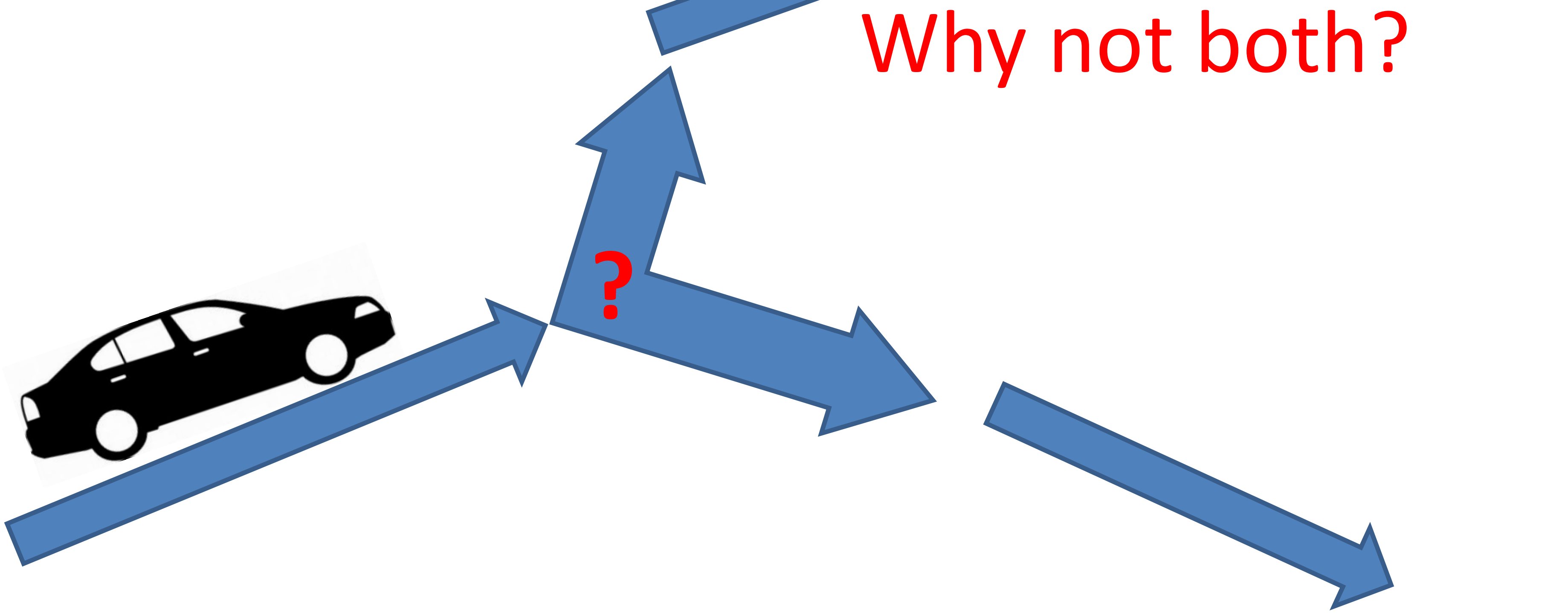


Nondeterminism

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:

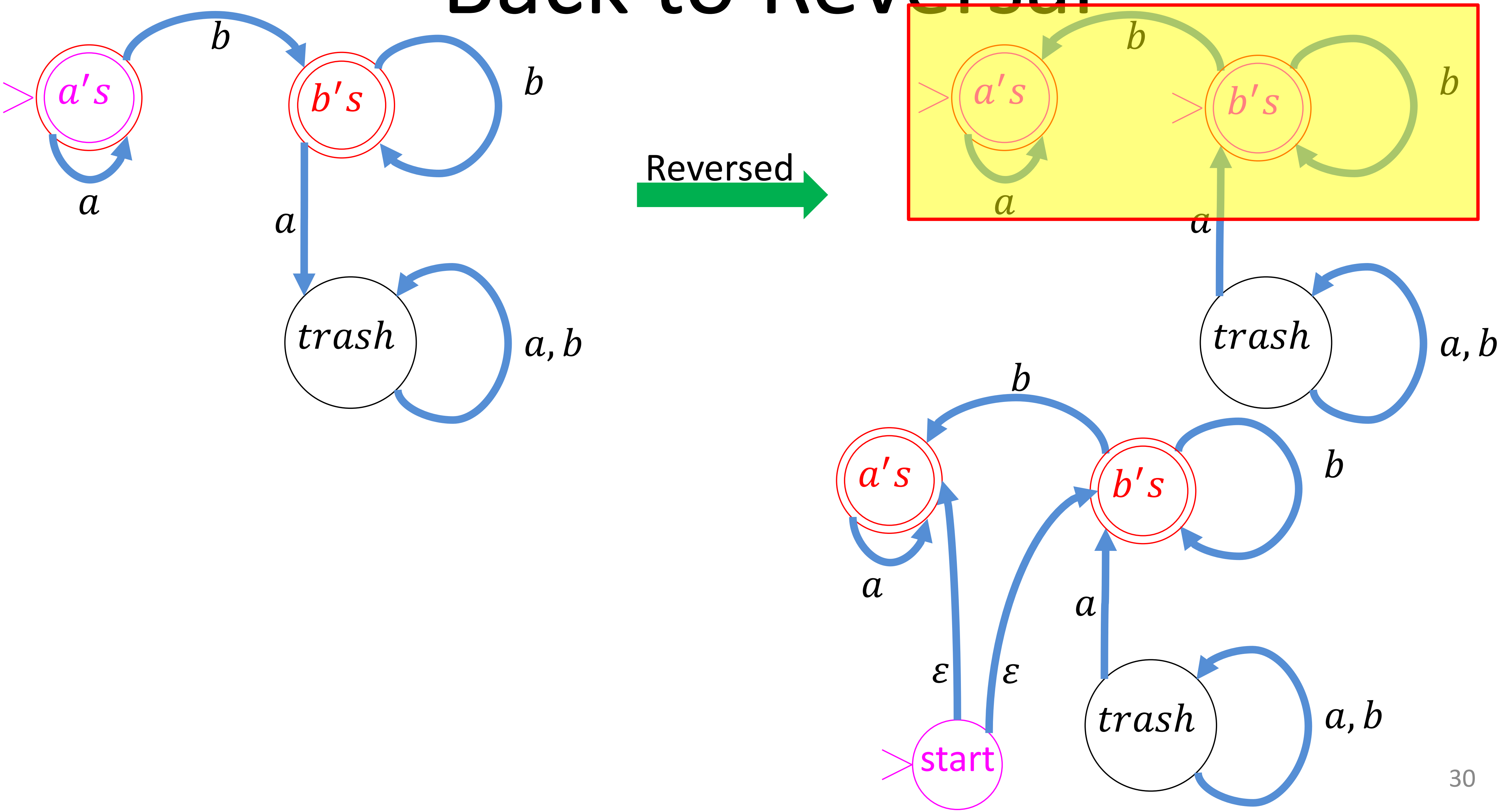
```
public static boolean isPrime(int n){
  for(int i = 2; i < n; i++){
    if(n % i == 0){
     return false;
    }
  }
  return true;
}
```

- 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 if Automata

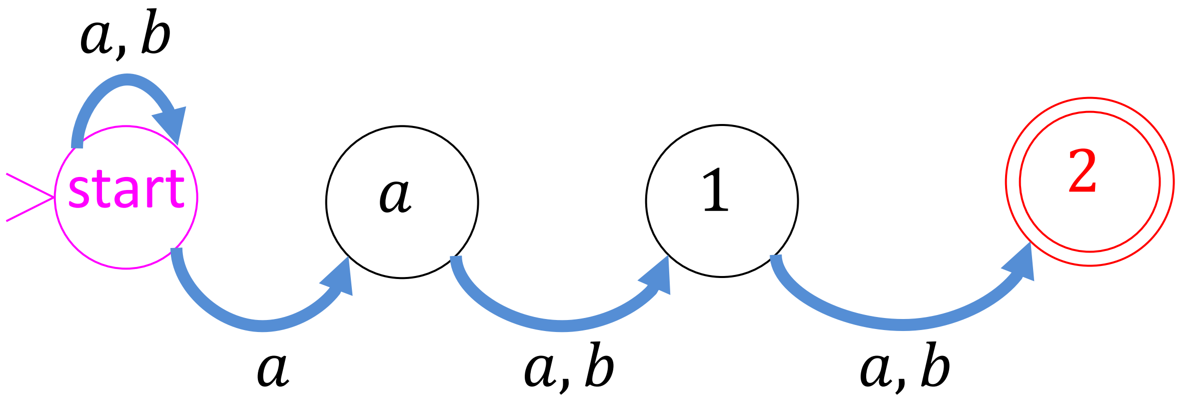
- Your machine can be in multiple states at once
- Accepts if any of the states it ends in are accepting states
- Relax restrictions:
  - Exactly one transition per symbol (can make multiple without consuming a symbol)
  - There must be exactly one outgoing transition for each symbol for every state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

Back to Reversal



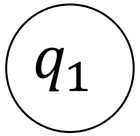
a is Third From Last

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

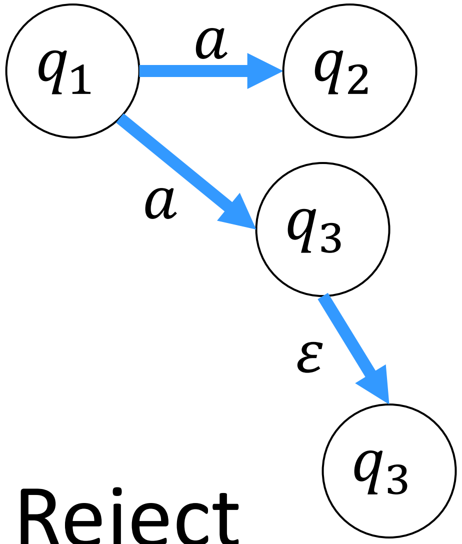


Nondeterministic Finite State Automata

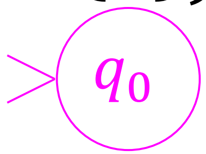
- Basic idea: a NFA is a "machine" that changes states while processing symbols, one at a time.
- Finite set of states: $Q = \{q_0, q_1, \dots q_7\}$



- 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