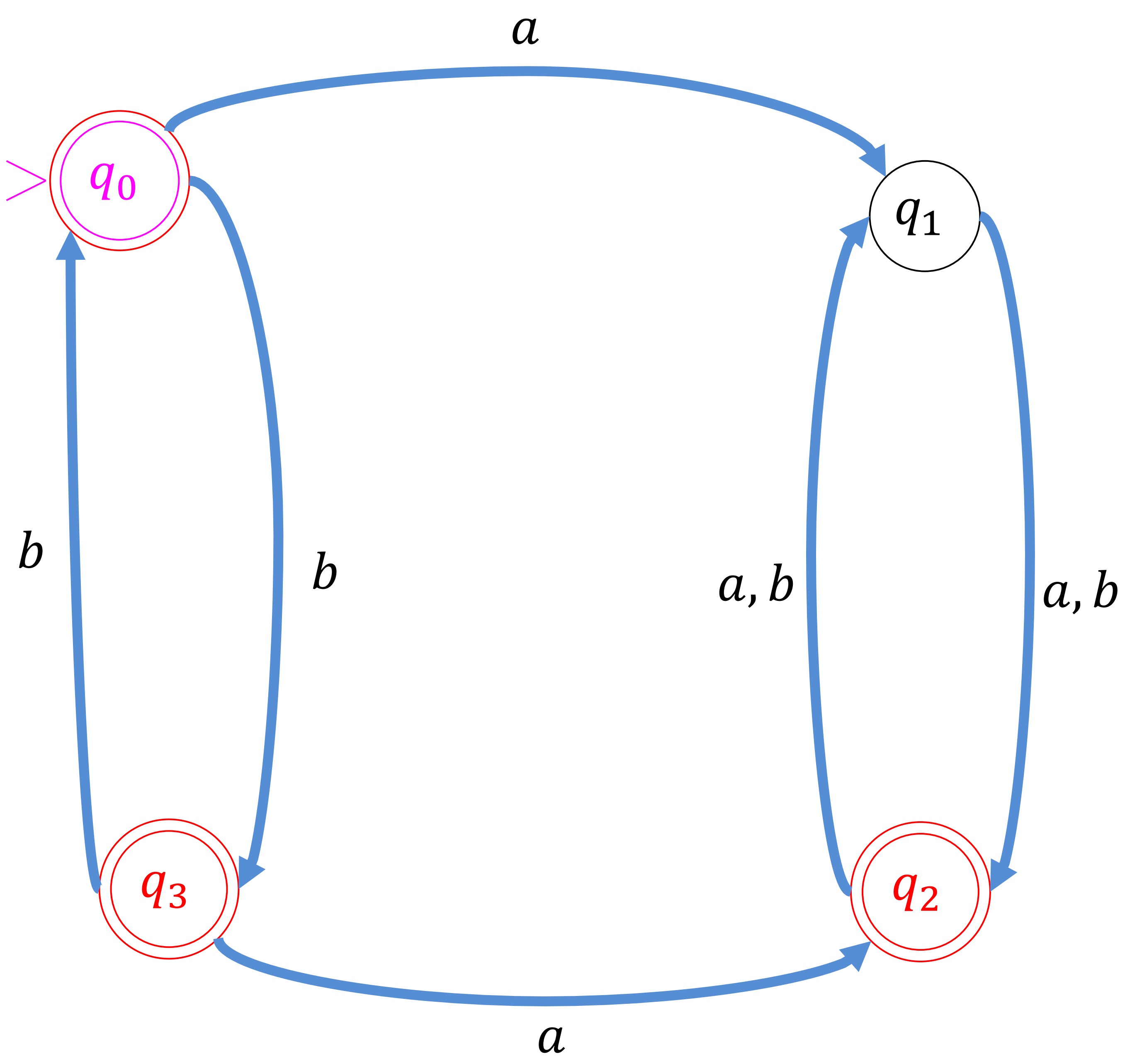
CS3102 Theory of Computation

Some strings it accepts:

Some strings it rejects:

Describe the language:



Finite State Automaton

- Simple model of computation
- Represents computation without memory
- Kind of decider
  - We call the set of strings it accepts the “language” of the machine
- Our machine reads the input string only once, and one character at a time
- After reading each character, enters a new “state”
- State transition rules depend only on the current state and the current character (no looking back!)
- There are only finitely many states

Gumball Machine

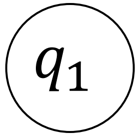
- Our gumball machine takes only pennies and nickels and does not give change
- Each gumball costs 7 cents
- $\Sigma = \{p, n\}$ (penny, nickel)
- We need to decide the language of sequences of coins adding up to at least 7 cents

Gumball Machine

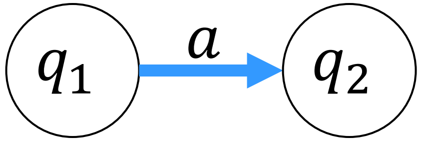
- What are all the possible “states” the machine could be in?
- 0c, 1c, 2c, 3c, 4c, 5c, 6c, 7+c
- Which “state” should the machine start in?
- Which “state” means we’ve sold a gumball?
- 6c plus a penny is always 7c, no matter how I got to 6c ($pppppp$, or pn, or np)

Finite State Automata

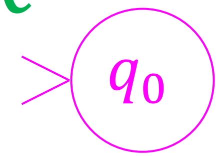
- Basic idea: a FA is a "machine" that changes states while processing symbols, one at a time.
- Finite set of states: $Q = \{q_0, q_1, \ldots q_7\}$



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



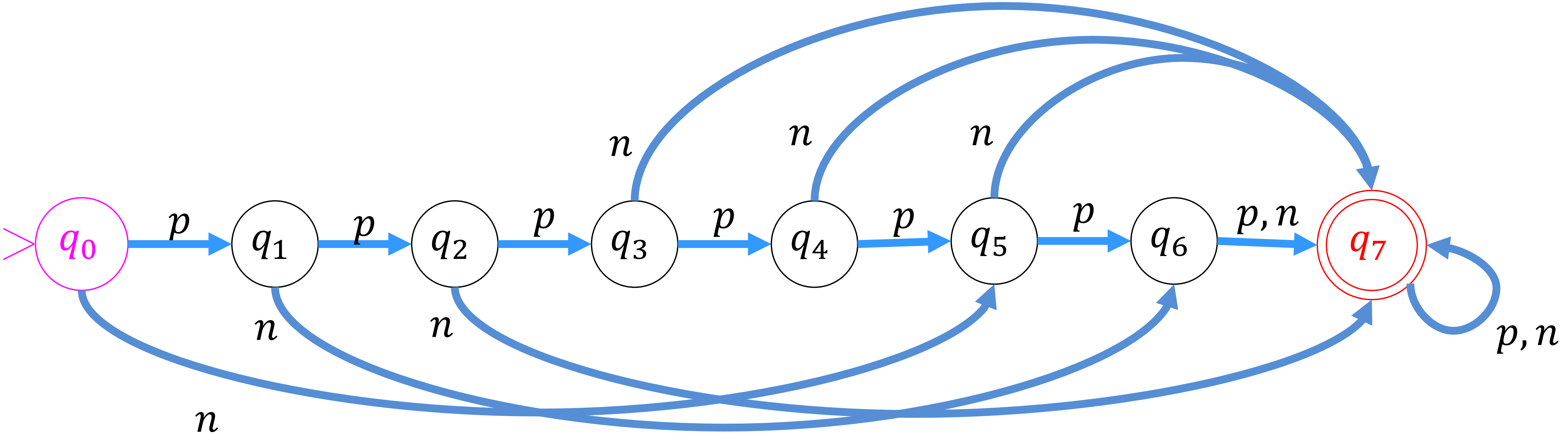
- Initial state: $q_0 \in Q$



- 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

FSA for our Gumball Machine



Strings this accepts:

$pppppppp$

$nnnnnnn$

pnp

ppn

Strings this rejects:

ppp

n

np

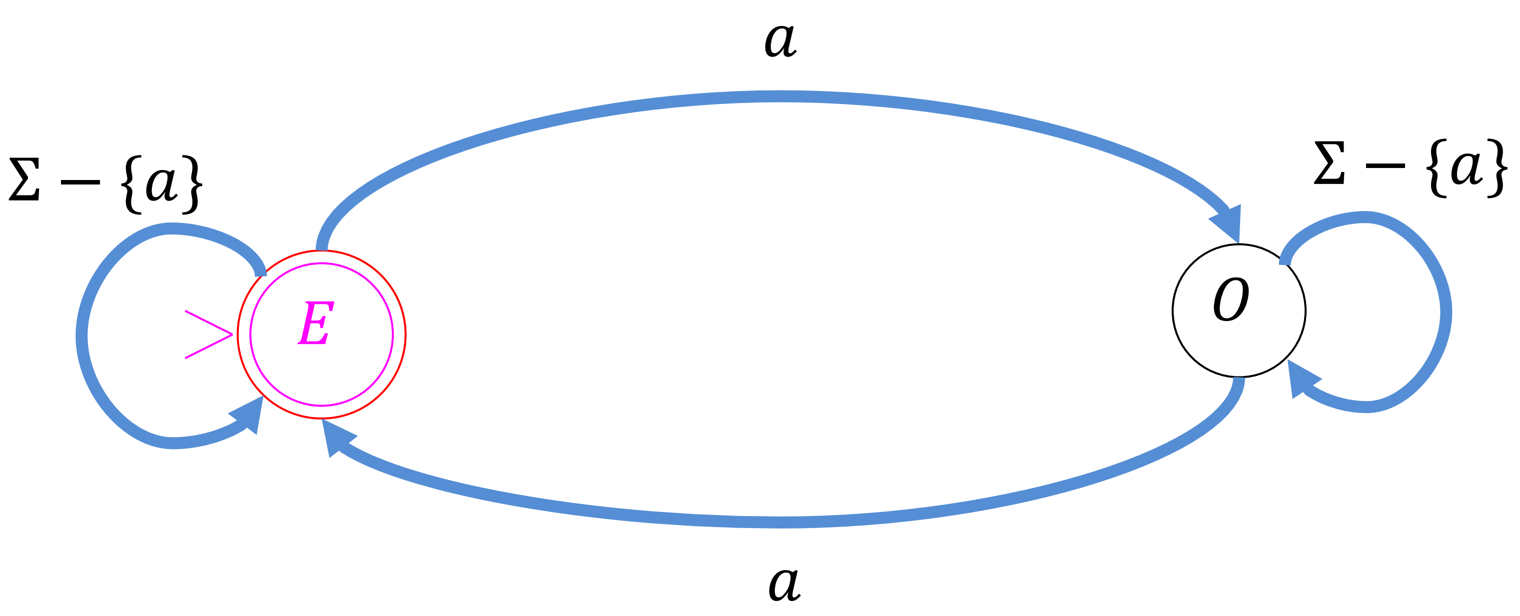
EvenA

- In HW1 you were asked to give a decider for EvenA (accepts all strings with an even number of A’s)
- How did you do it?

EvenA 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 F)
4. Which state is the start state? (pick q_0)
5. How should we transition? (pick δ)

Let’s Draw It!

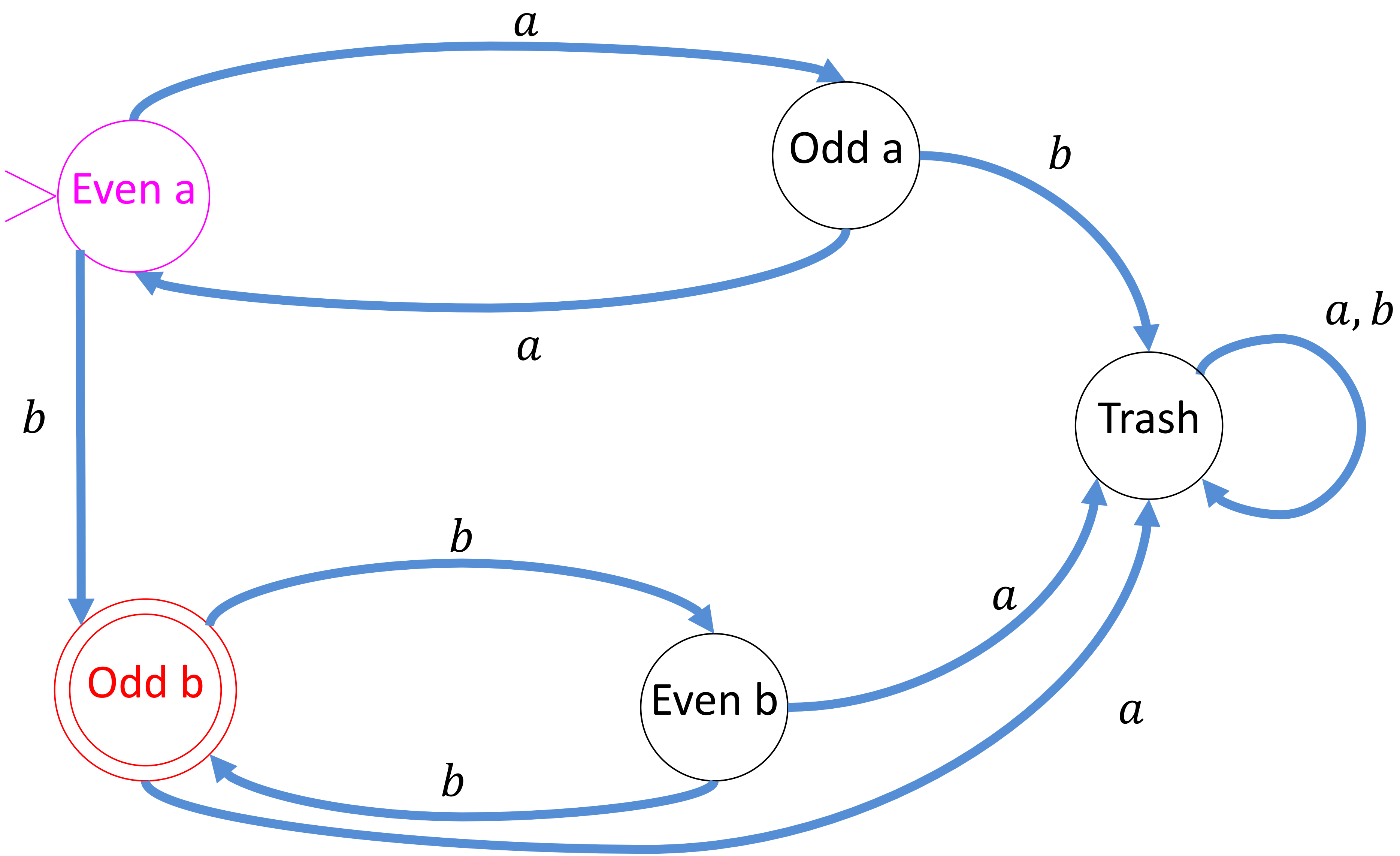


EvenAOddB

- Let’s make a finite state automaton which accepts strings that have an even number of a’s followed by an odd number of b’s (in that order)
- It should accept:
  - $b, bbb, aab, aaaabbbbb, \ldots$
- It should reject:
  - $bb, ab, baa, aba, aaabb$

EvenAoddB

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



EvenAOddB 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 F)
4. Which state is the start state? (pick q_0)
5. How should we transition? (pick δ)

TripleA

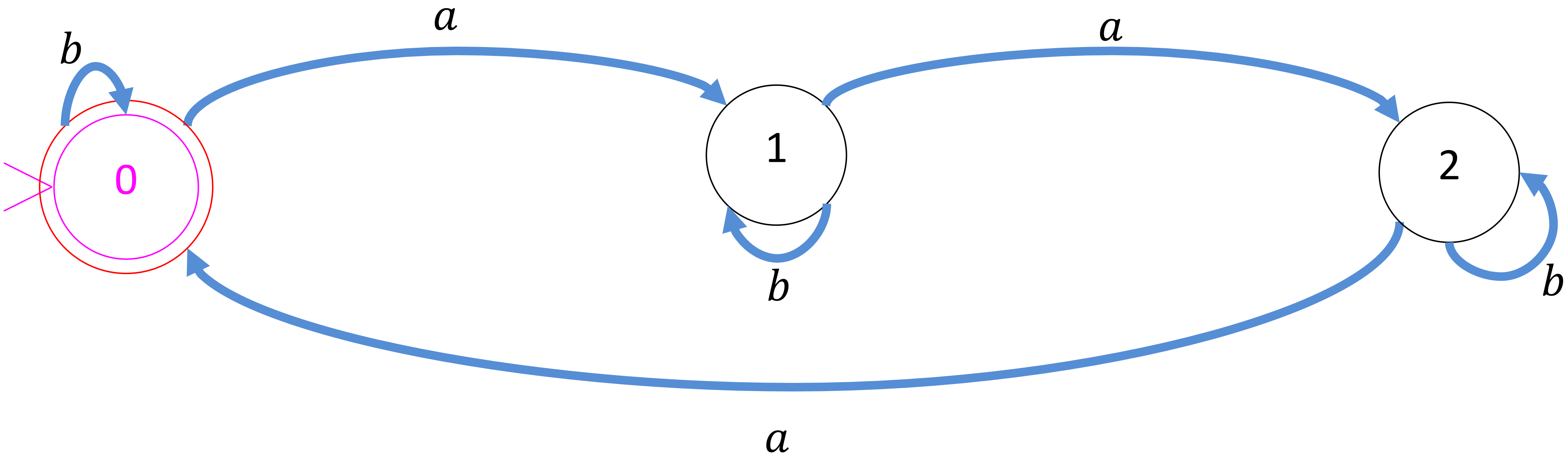
- Let's make a finite state automaton which accepts strings where the number of a's is a multiple of 3
- It should accept:
  - $b, aaa, abaa, \ldots$
- It should reject:
  - $a, ab, baa, aba, aaaabbb, \ldots$

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 F)
4. Which state is the start state? (pick q_0)
5. How should we transition? (pick δ)

EvenAoddB

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
- Concatenation

Closed under Complement

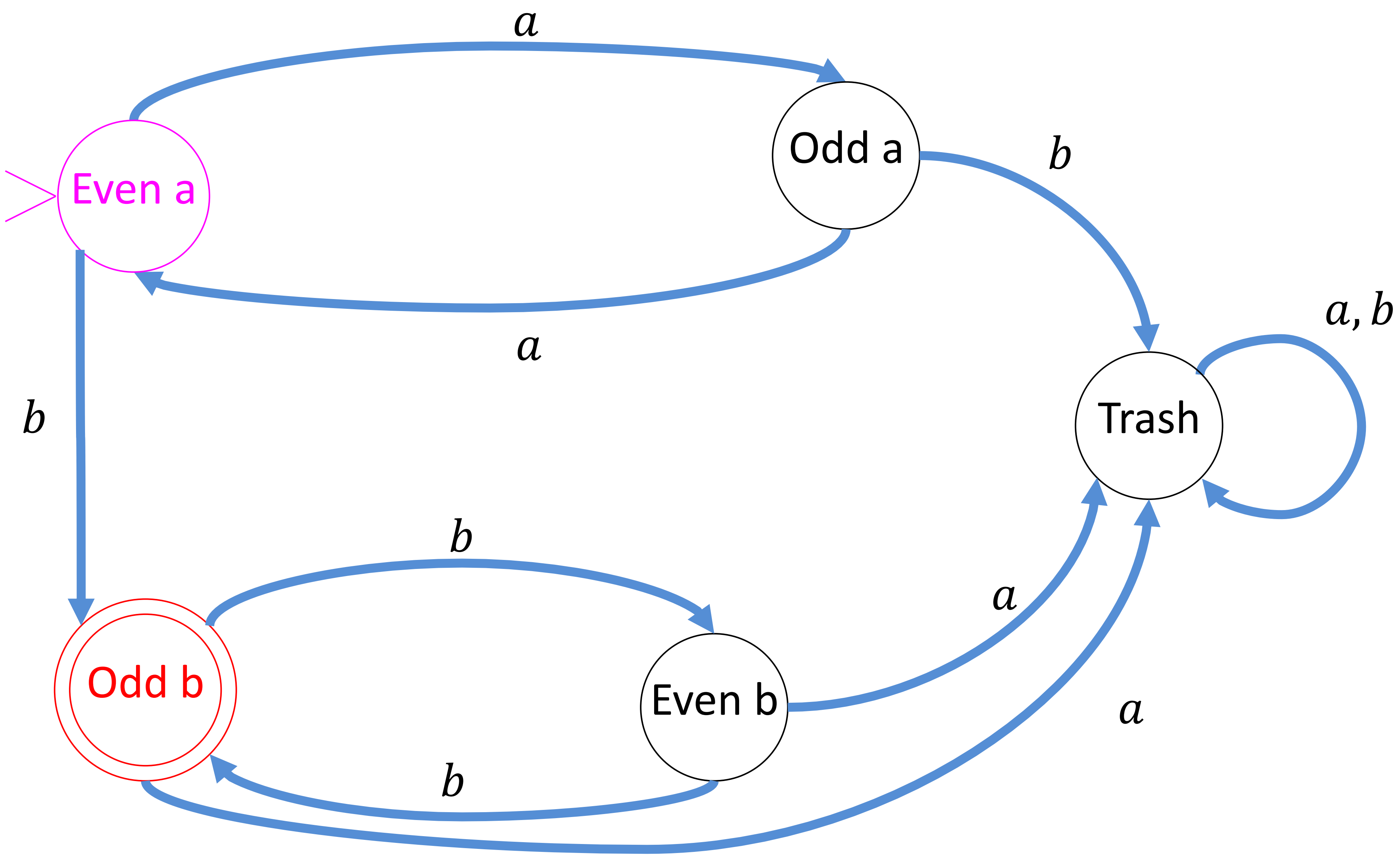
- If a language is regular then its complement is regular
- If a language has a FSA, it’s complement does as well
- If there is a FSA which accepts exactly the strings in the language, there is a FSA which accepts exactly the strings not in the language

Closed under complement

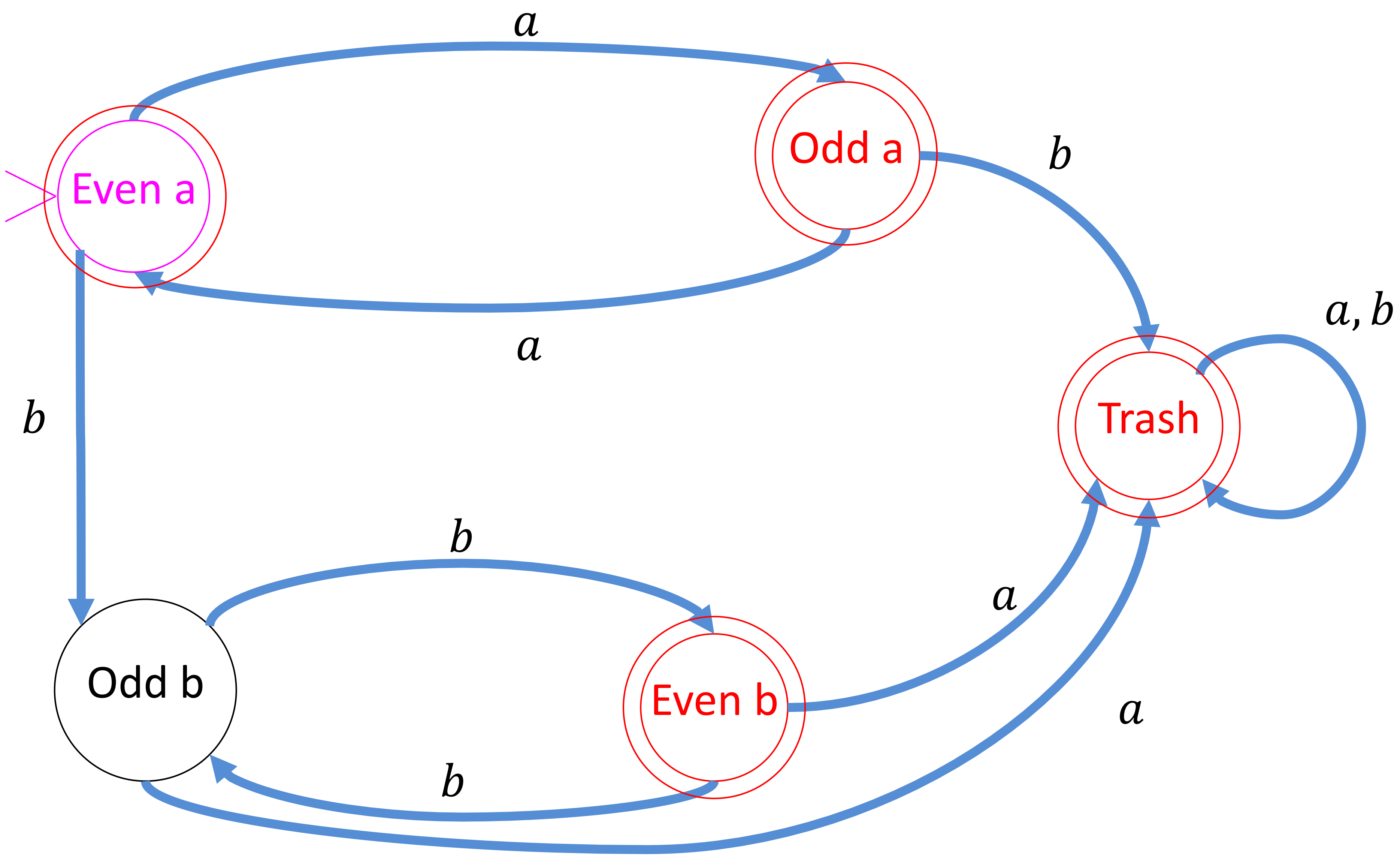
- Idea: Every string ends in some state. If that was originally an accept state then reject, else accept.
- New final states are the old non-final states

EvenAoddB

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



Complement of EvenAoddB

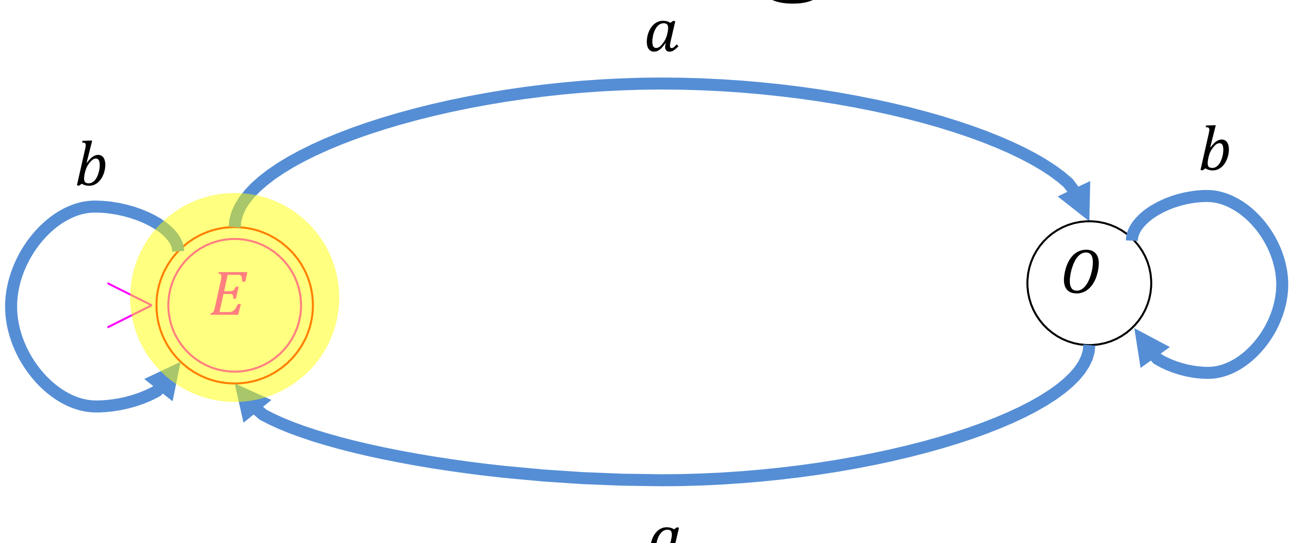


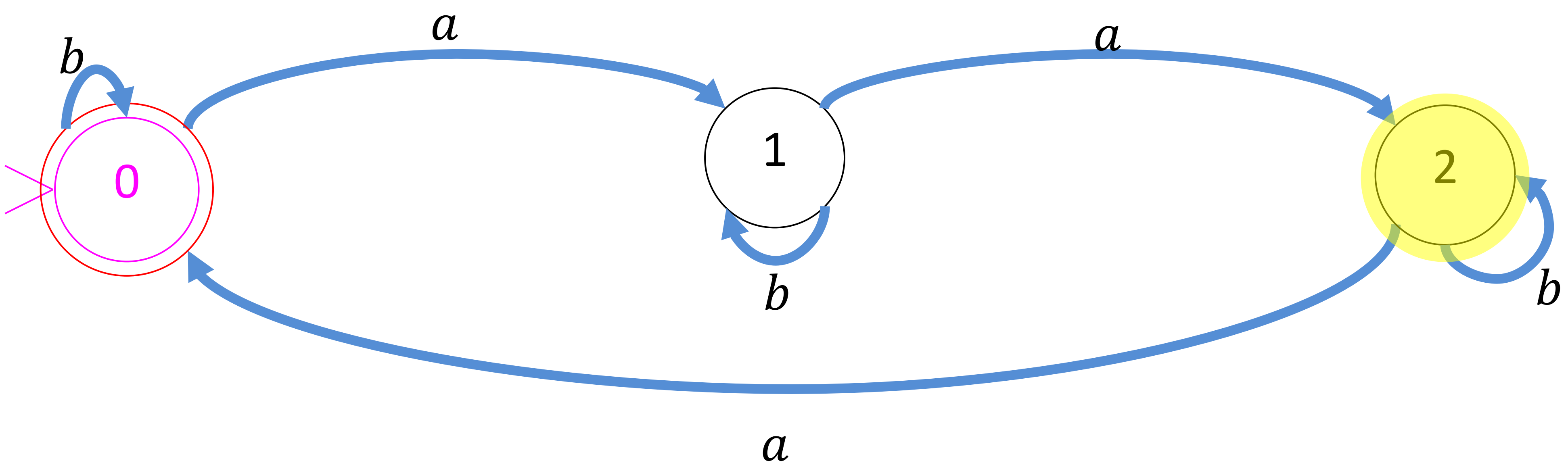
Closed under Intersection

- 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: $aaaaaaaa$

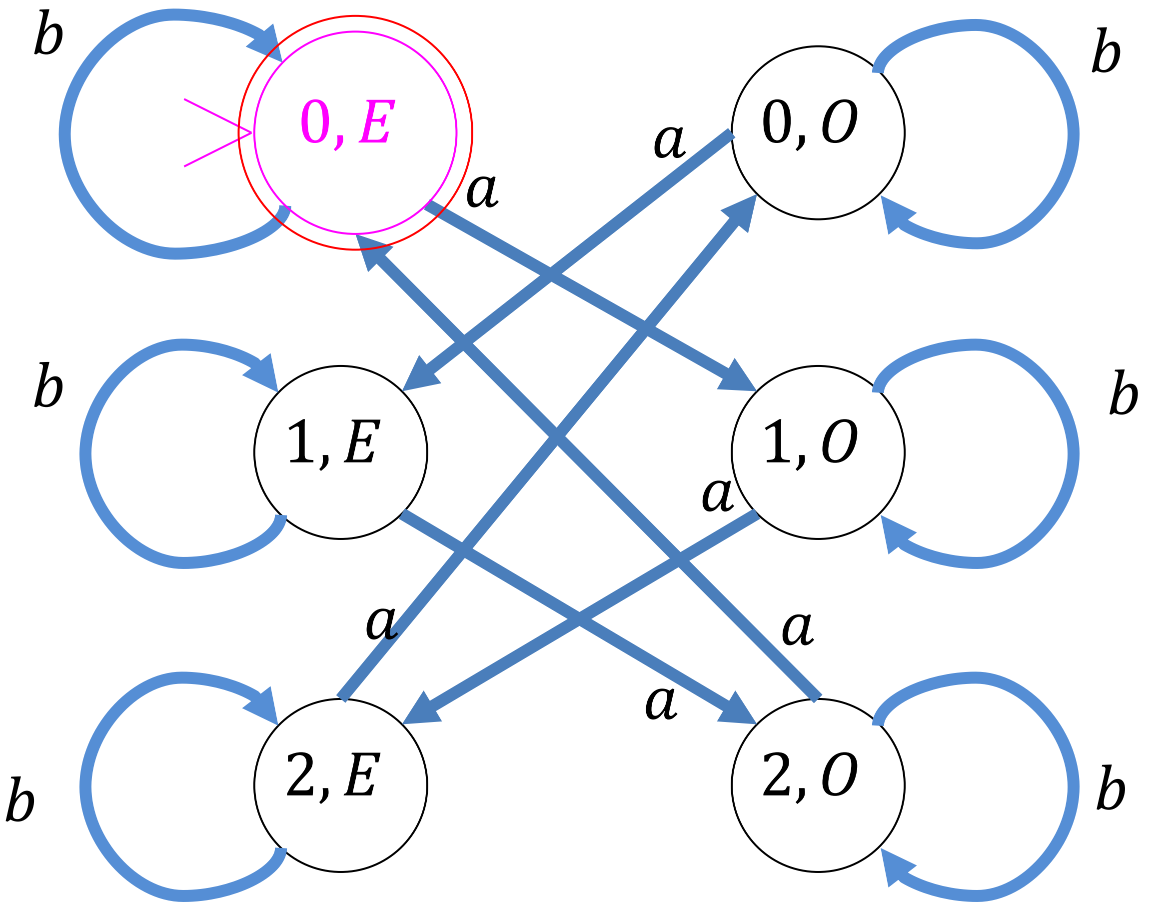




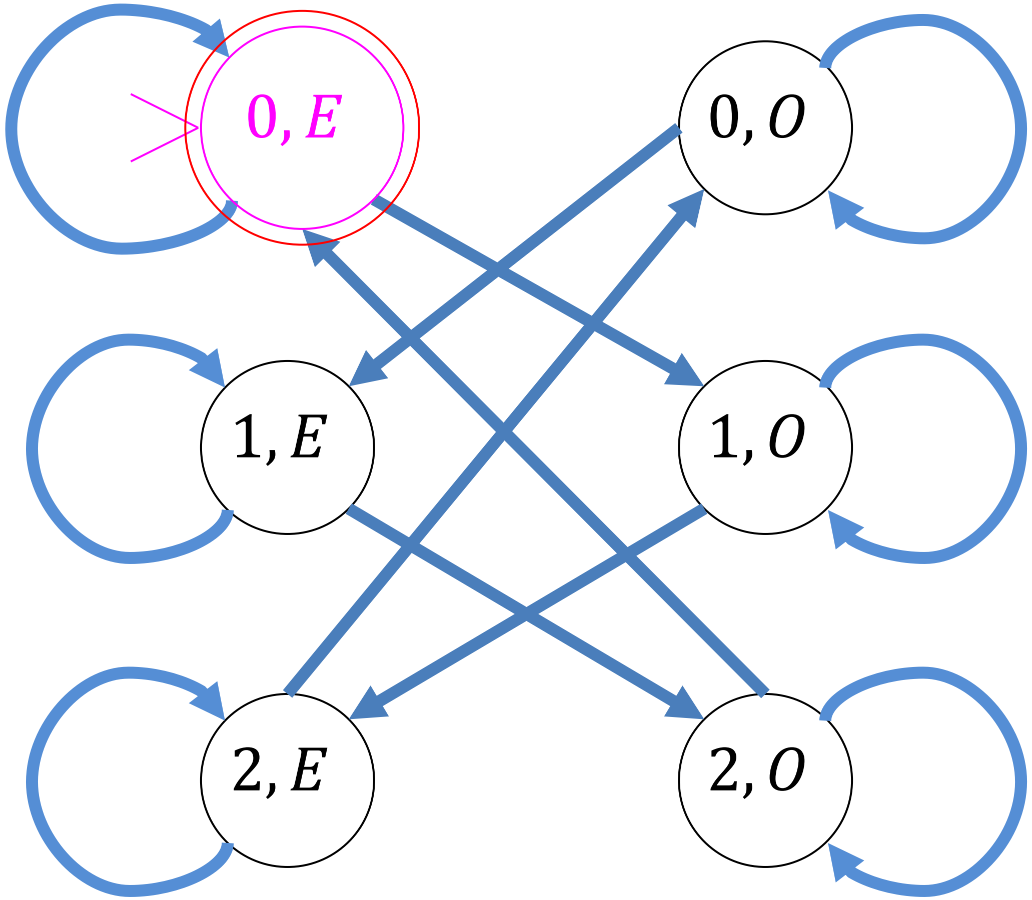
Now at the same time

- 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 bother were final
  - $\{(0, E)\}$
- Transitions: One arrow represents transitioning in both machines

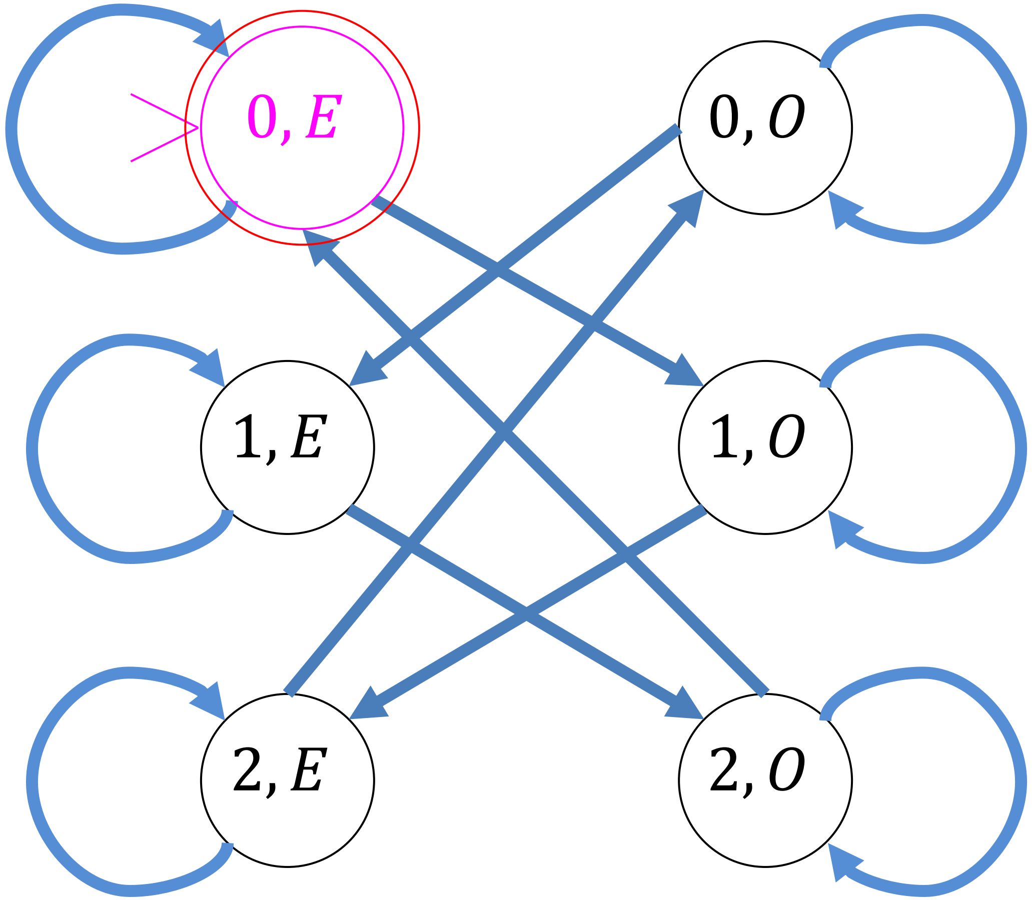
Let's Draw it!



Let's Draw it!



Let’s Draw it!



Cross Product Construction

- 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: (for intersection only) $F_\times = F_1 \times F_2$
- Finite state automaton is $M_\times = (Q_\times, \Sigma, \delta_\times, (q_{01}, q_{02}), F_\times)$