Week 7 Tutorial

Regular Languages

Download the starter files for Problem Set Six, extract them somewhere convenient, and run the provided program. You will need the Automaton Editor to complete today's tutorial exercises.

Part 1: ***Designing DFAs***

Designing DFAs

- ***States*** – pieces of information
  - What do I have to keep track of in the course of figuring out whether a string is in this language?

Designing DFAs

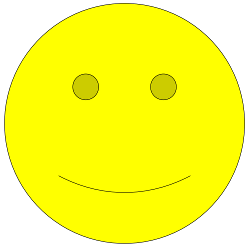
- ***States*** – pieces of information
  - What do I have to keep track of in the course of figuring out whether a string is in this language?
- ***Transitions*** – updating state
  - From the state I’m currently in, what do I know about my string? How would reading this character change what I know?

An Analogy

Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language.

An Analogy

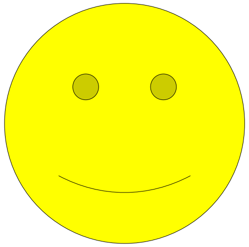
Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language.



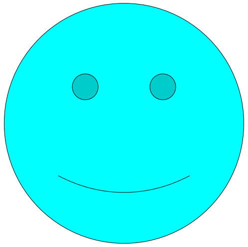
Alice

An Analogy

Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language.



Alice

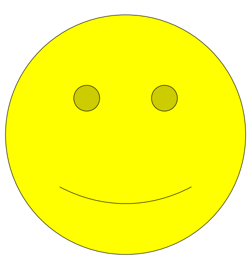


Bob

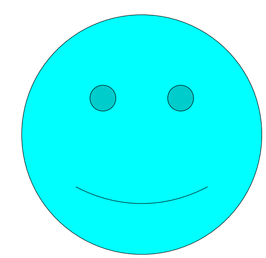
An Analogy

Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language.

$L = \{ w \mid w$ is a natural number divisible by 5 $\}$



Alice

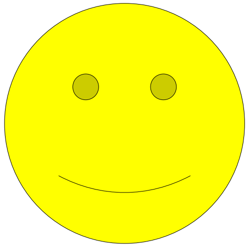


Bob

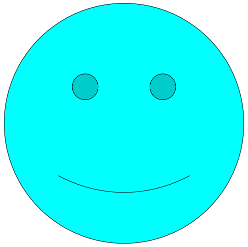
An Analogy

Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language.

$L = \{ \; w \mid w$ is a natural number divisible by 5 $\}$



Alice

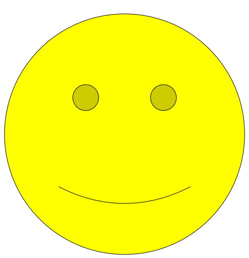


Bob

An Analogy

The catch: Bob can only send Alice one character at a time, and Alice doesn't know how long the string is until Bob tells her that he's done sending input.

$L = \{ w \mid w$ is a natural number divisible by 5 $\}$



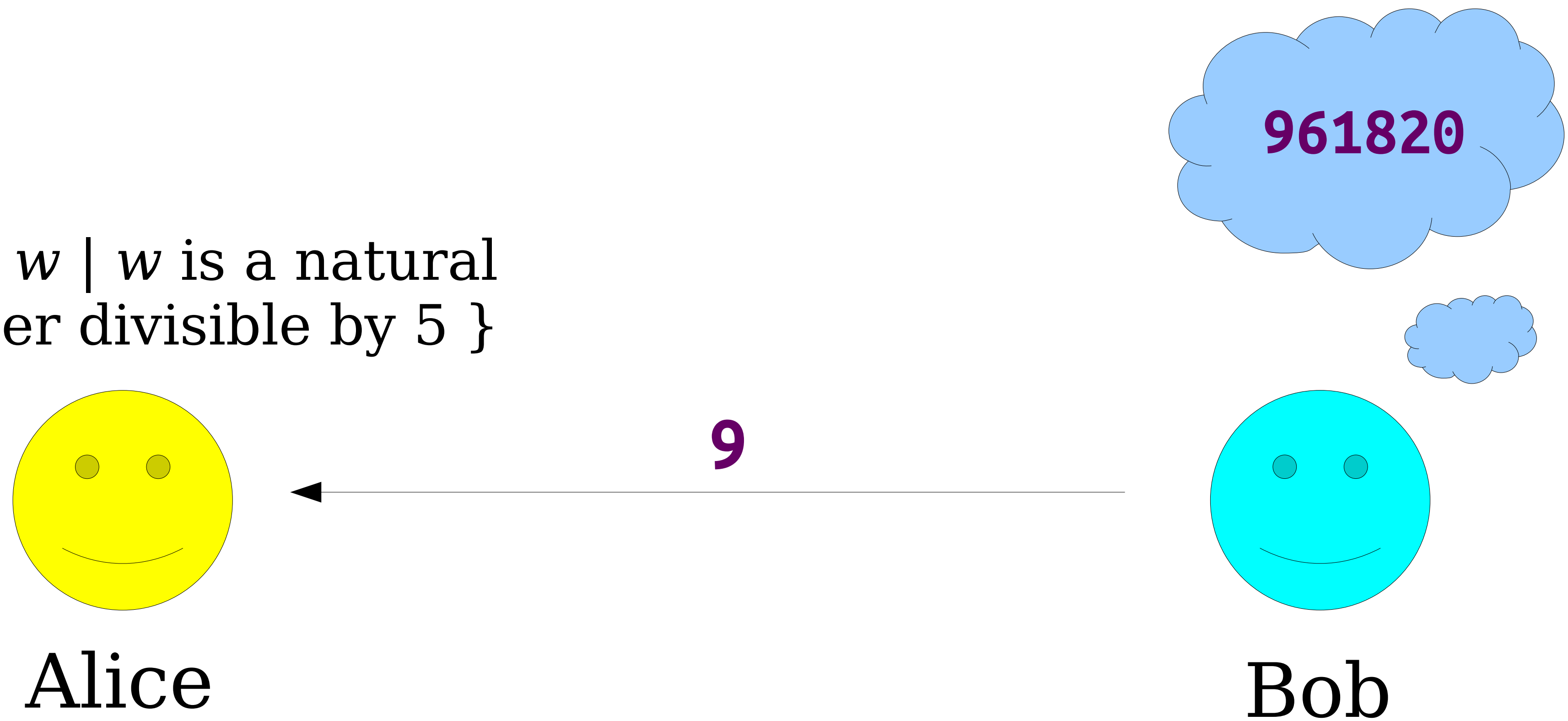
Alice

Bob

An Analogy

The catch: Bob can only send Alice one character at a time, and Alice doesn't know how long the string is until Bob tells her that he's done sending input.

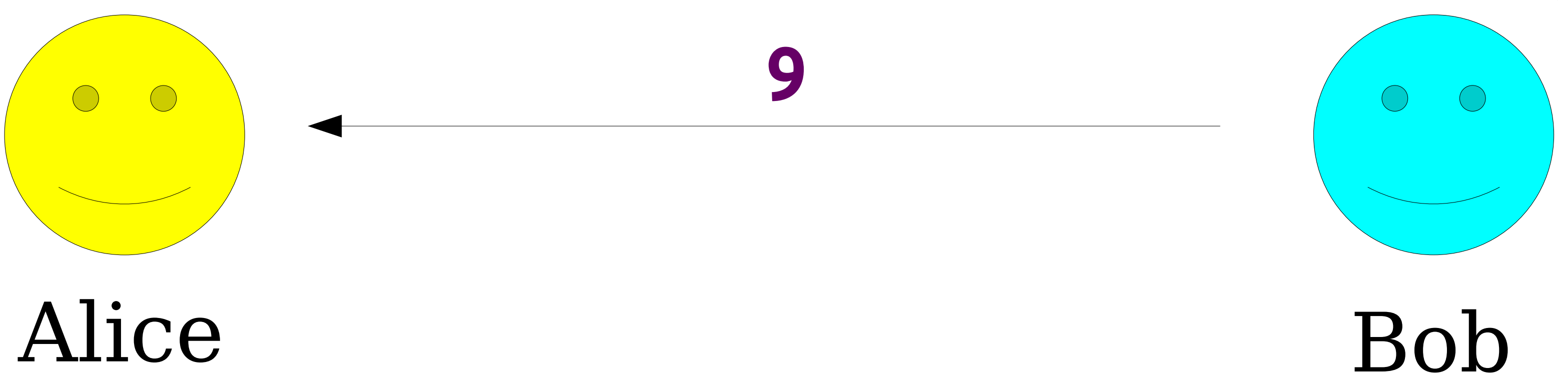
$L = \{ w \mid w \text{ is a natural number divisible by 5} \}$



An Analogy

What does Alice need to remember about the characters she's receiving from Bob?

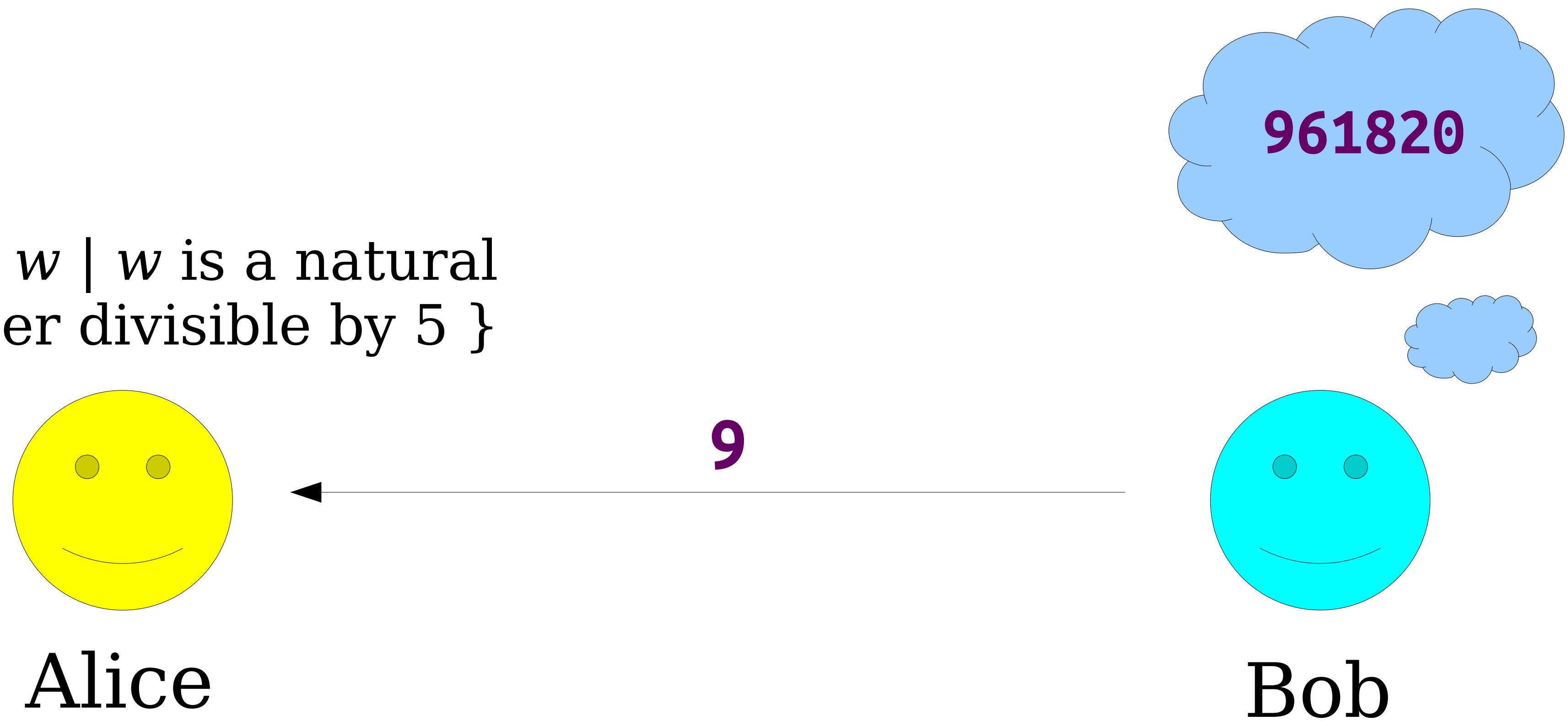
$L = \{\ w \mid w$ is a natural number divisible by 5 $\}$



An Analogy

Key insight: Alice only needs to remember the last character she received from Bob.

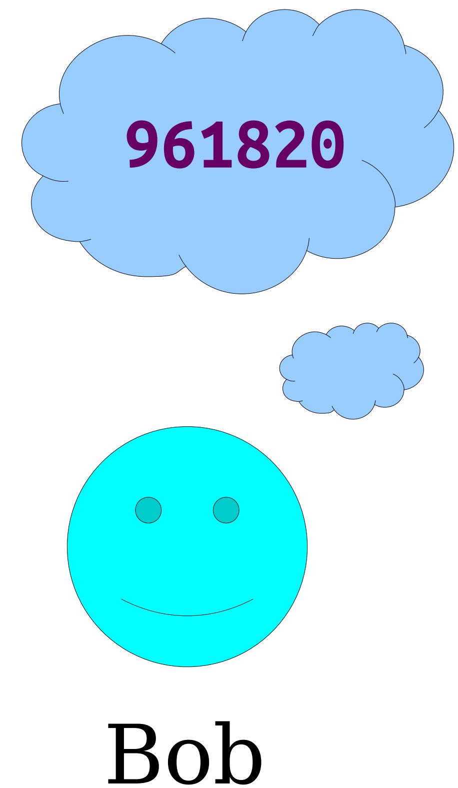
$L = \{\ w \mid w$ is a natural number divisible by 5 $\}$



An Analogy

Key insight: Alice only needs to remember the last character she received from Bob.

$L = \{\ w \mid w \text{ is a natural number divisible by 5 }\}$



An Analogy

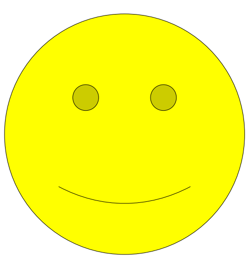
Key insight: Alice only needs to remember the last character she received from Bob.

$L = \{\ w \mid w$ is a natural number divisible by 5 $\}$

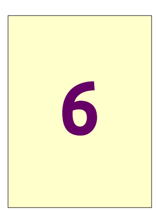
An Analogy

Key insight: Alice only needs to remember the last character she received from Bob.

L = { w | w is a natural number divisible by 5 }



Alice

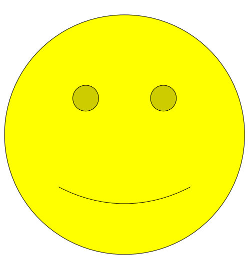


Bob

An Analogy

Key insight: Alice only needs to remember the last character she received from Bob.

$L = \{ w \mid w \text{ is a natural number divisible by 5} \}$



. . .

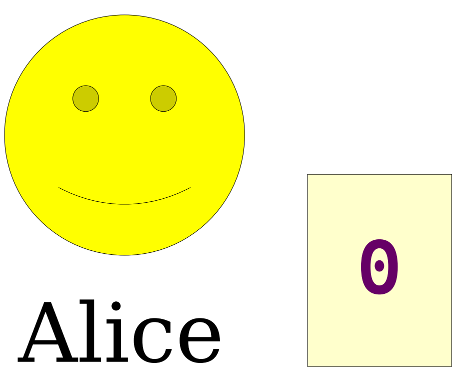
Alice

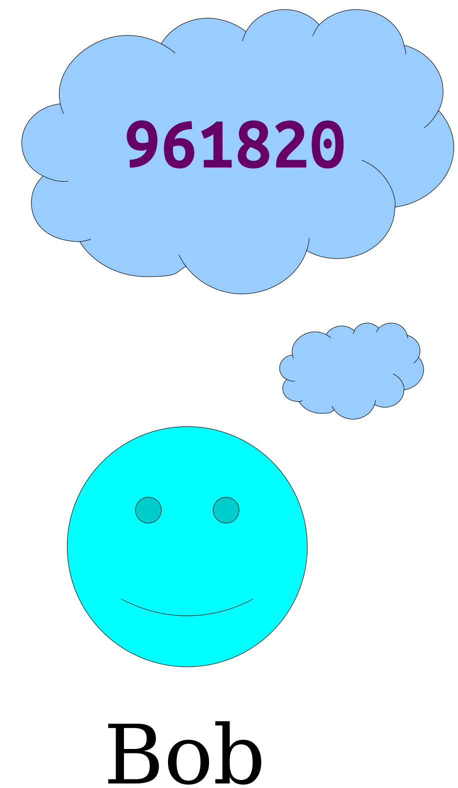
Bob

An Analogy

Eventually Bob gets to the end of his string and sends Alice a signal that he's done sending input.

$L = \{\ w \mid w$ is a natural number divisible by 5 $\}$





An Analogy

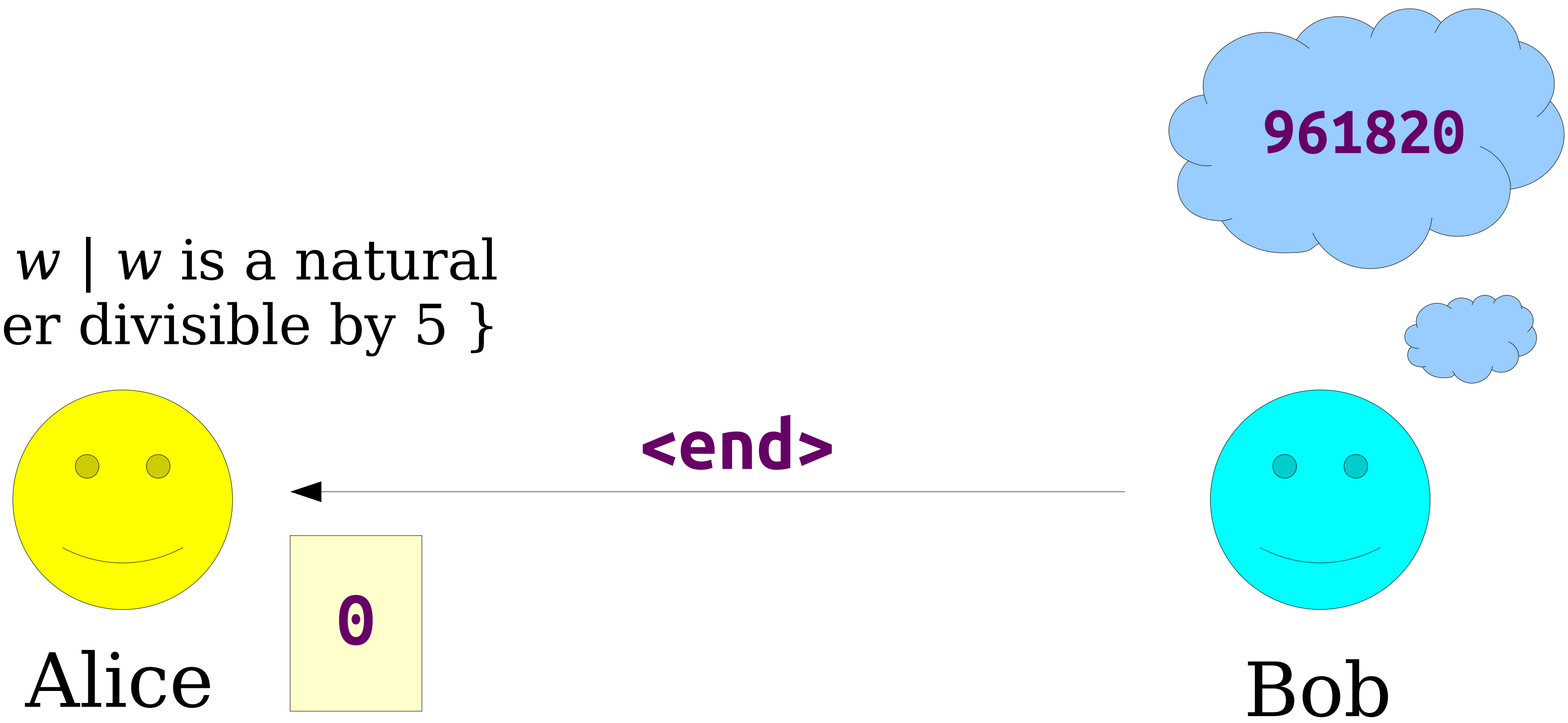
Eventually Bob gets to the end of his string and sends Alice a signal that he’s done sending input.

L = { w | w is a natural number divisible by 5 }

An Analogy

At this point, Alice just has to look at the last digit she wrote down and if it’s a 5 or 0, Bob’s string belongs in the language.

$L = \{ w \mid w$ is a natural number divisible by 5 $\}$



DFA Design Strategy

- ***Identify Core Information***
  - Answer the question “What do I have to keep track of in the course of figuring out whether a string is in this language?”
- ***Create Your States***
  - Create a state that represents each possible answer to that question.
- ***Add Transitions***
  - From each state, go through all of the characters and answer the question “How would reading this character change what I know about my string?” and draw transitions to the appropriate states.


Art by 白砂糖
OREO
O&REO
O&O
OREOREO
RERERERERE
OOOOO
OREOO
OREOREREREORE
OREOREORE
REREO
REORE
OREREREREREREREO
OOOREREREREREREOOO
OREREREREOOOOOOOOO
Typeset by Ree
Translated by Mysterious Uploader (danbooru)

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$

For simplicity, let's just use a single character for the "cream" part of the Oreo :)

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

$$L = \{ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}.$$

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

$$L = \{ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}.$$

ORO $\in L$	**OR** $\notin L$
ROOOR $\in L$	**OOOOOR** $\notin L$
OROORORRO $\in L$	**ROROROROO** $\notin L$

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

$$L = \{ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}.$$

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

$$L = \{ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}.$$

What do I have to keep track of in the course of figuring out whether a string is in this language?

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

$$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same }\}.$$

- We need to keep track of the very first character.
- And we need to keep track of the last character we've read so that when we reach the end, we can check whether the first and last characters were the same.

Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design a DFA for the language

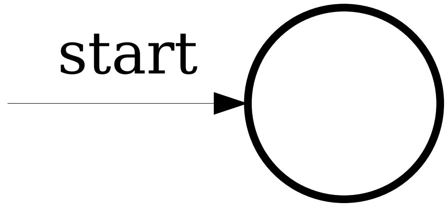
$$L = \{ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}.$$

1) Draw a DFA for L using the Automaton Editor and save it as `res/TutorialWeek7.Q1.automaton`

   Then, submit that file to Gradescope.

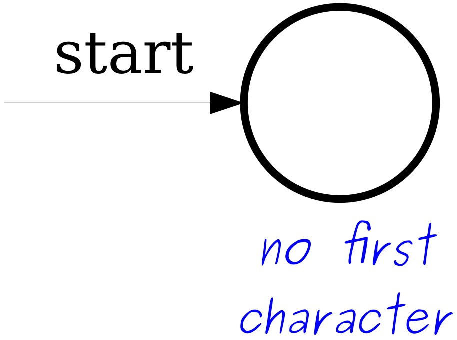
Oreo Sandwiches

$$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}$$



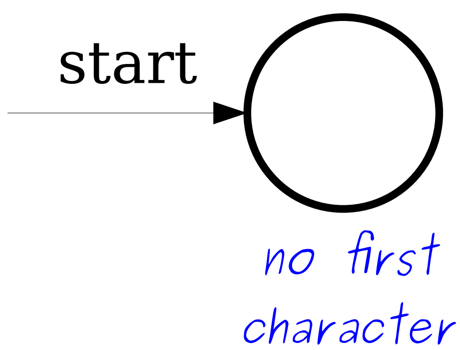
Oreo Sandwiches

$$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same }\}$$



Oreo Sandwiches

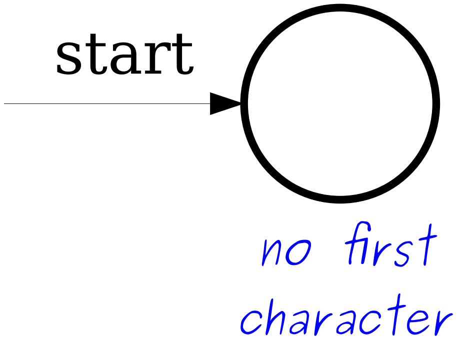
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Remember that each state should represent a piece of information. We'll annotate what each state represents in blue.

Oreo Sandwiches

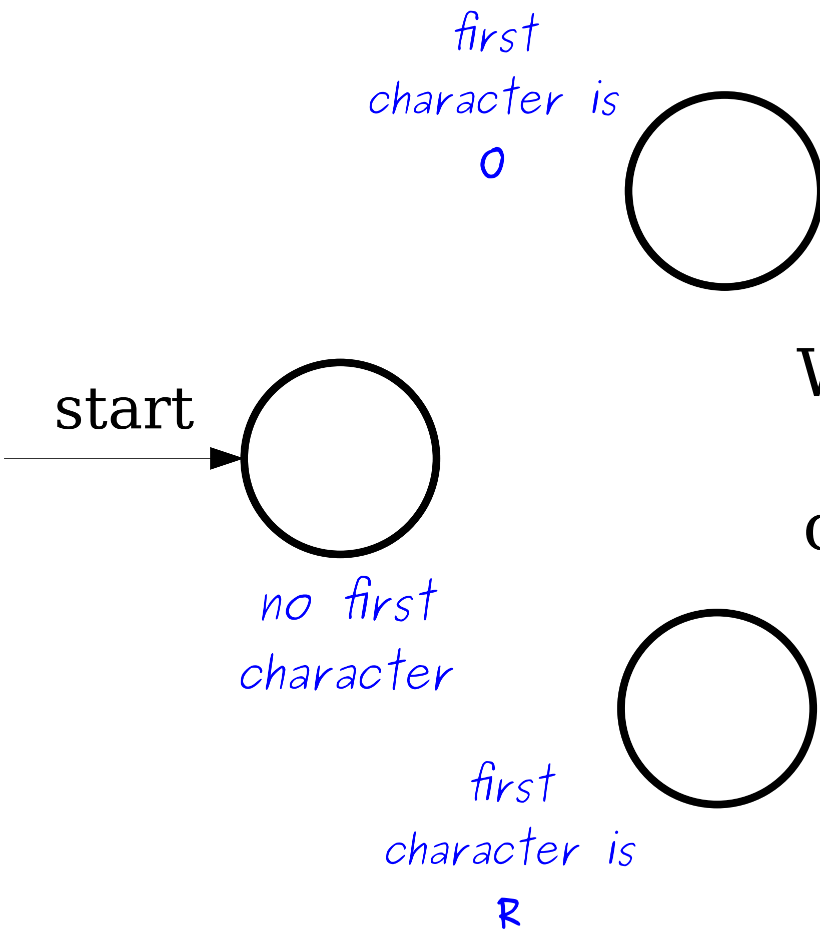
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



We need to keep track of the very first character, which could either be an **O** or an **R**.

Oreo Sandwiches

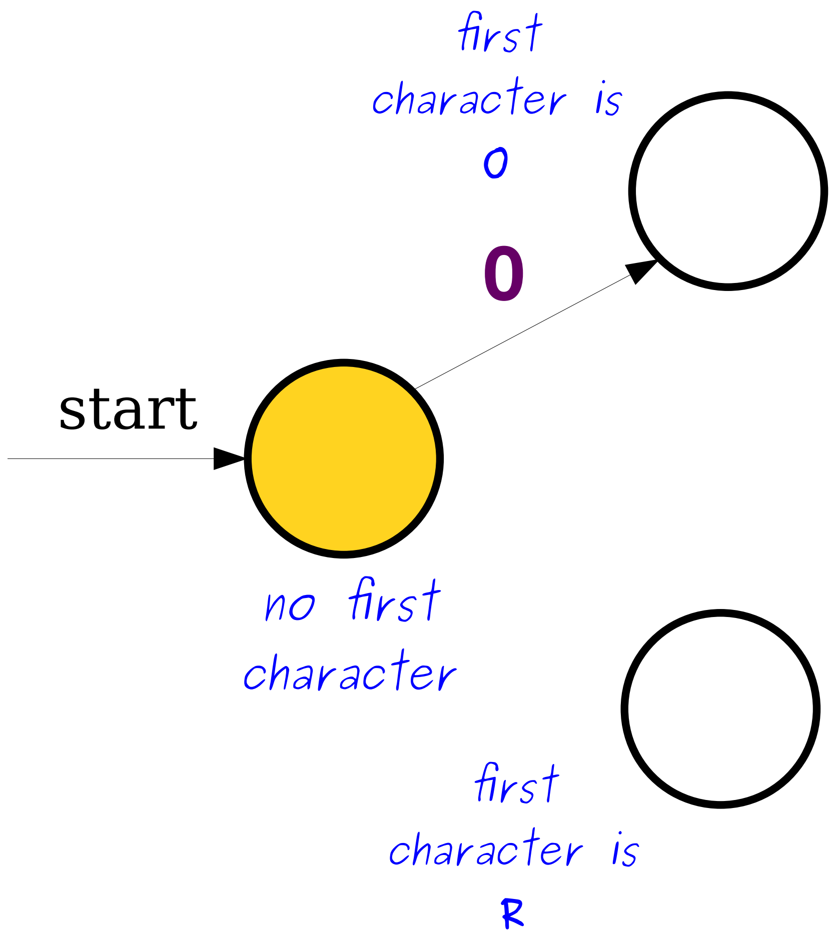
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



We need to keep track of the very first character, which could either be an **O** or an **R**.

Oreo Sandwiches

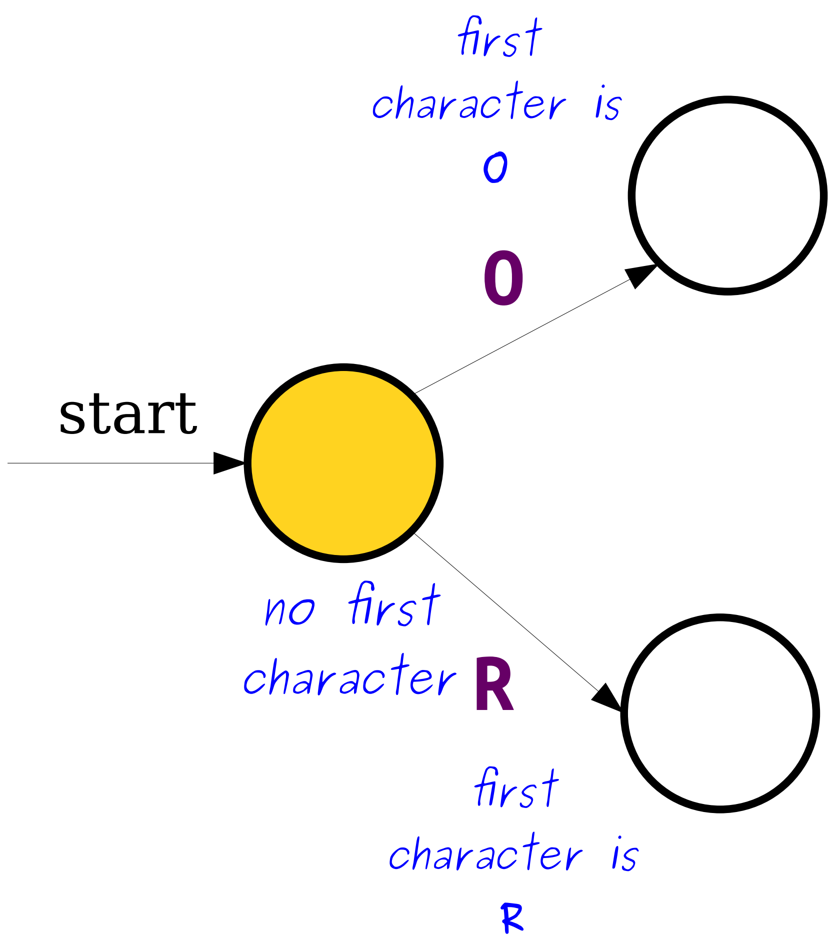
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



If I'm in the start state and I read an **0**, I should transition to this state

Oreo Sandwiches

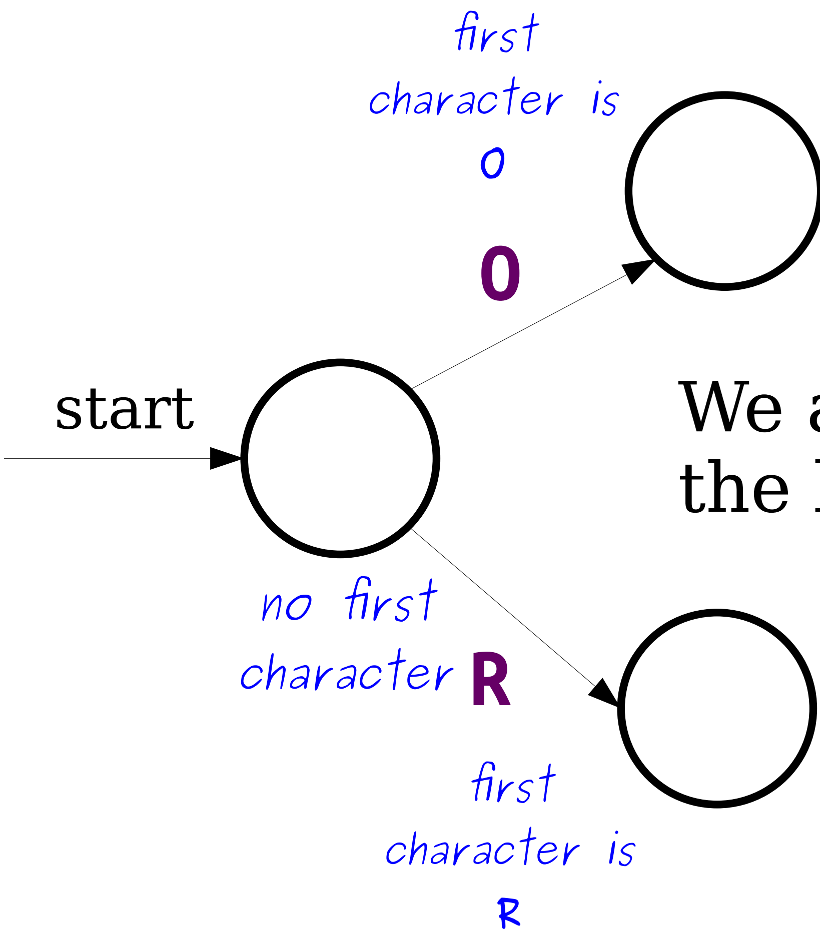
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Likewise if I'm in the start state and I read an **R**, I should transition to this state

Oreo Sandwiches

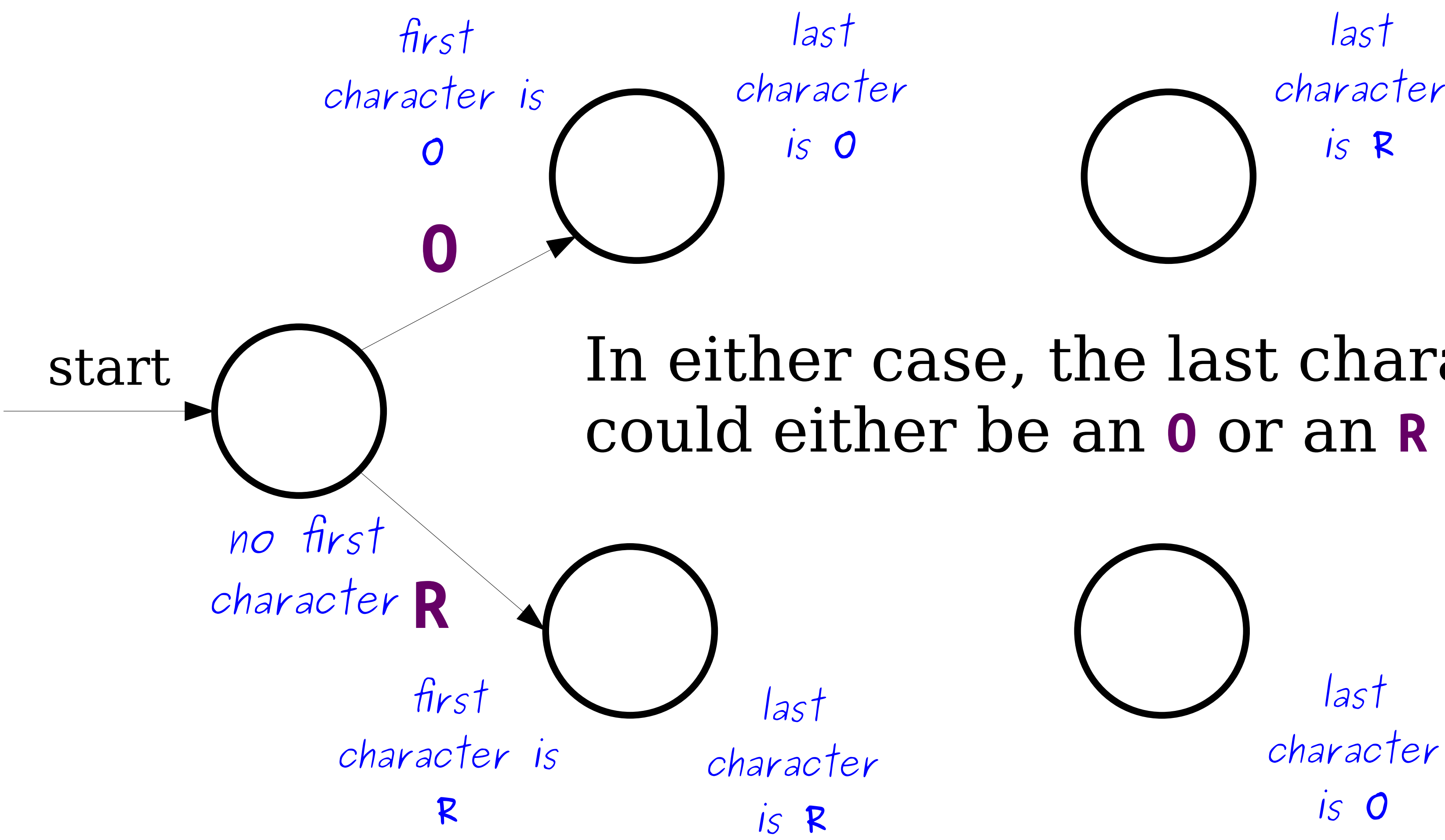
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



We also need to keep track of the last character we've read

Oreo Sandwiches

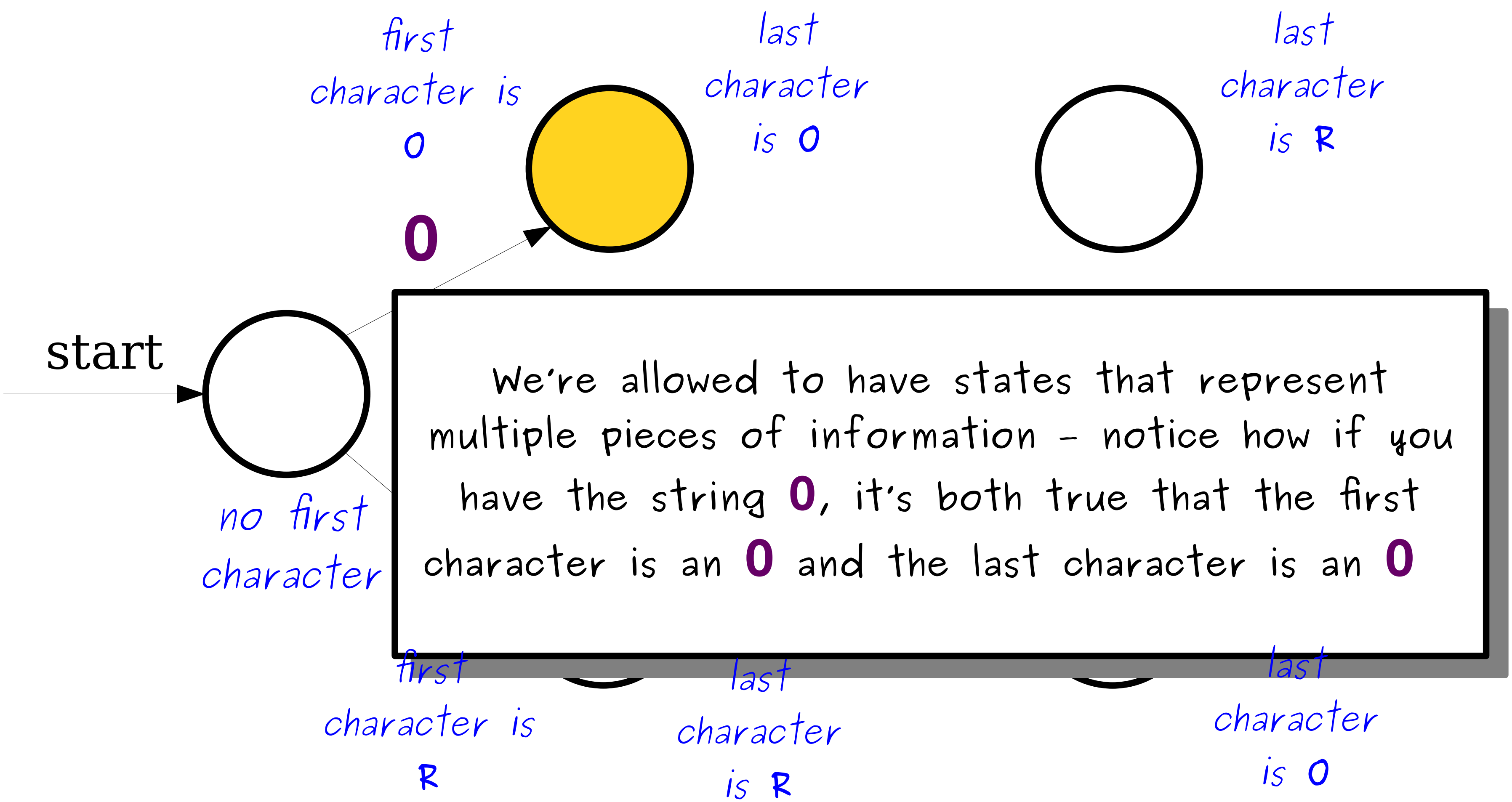
$$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same }\}$$



In either case, the last character could either be an **O** or an **R**

Oreo Sandwiches

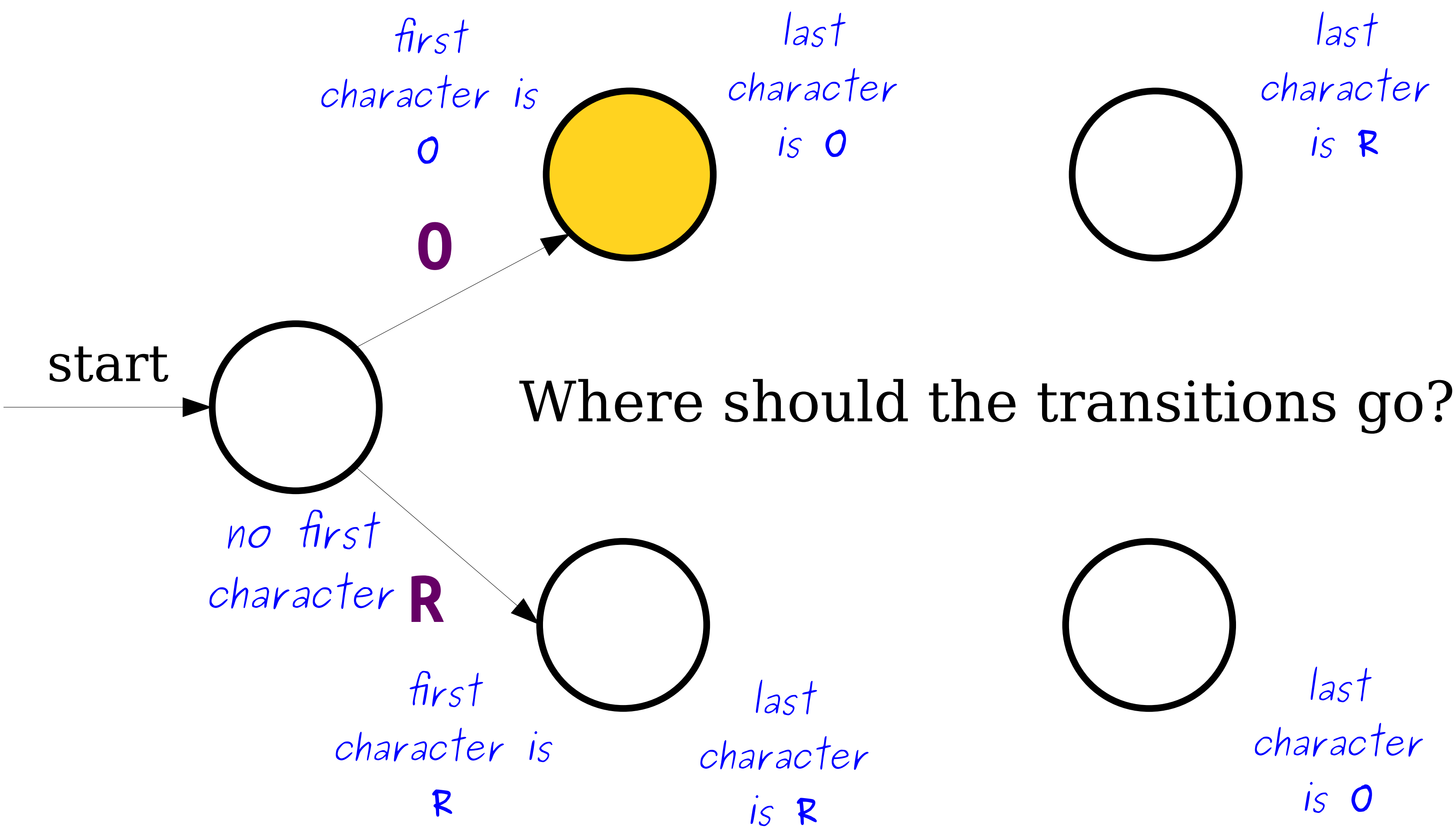
$L = \{ \; w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



We're allowed to have states that represent multiple pieces of information – notice how if you have the string **0**, it's both true that the first character is an **0** and the last character is an **0**

Oreo Sandwiches

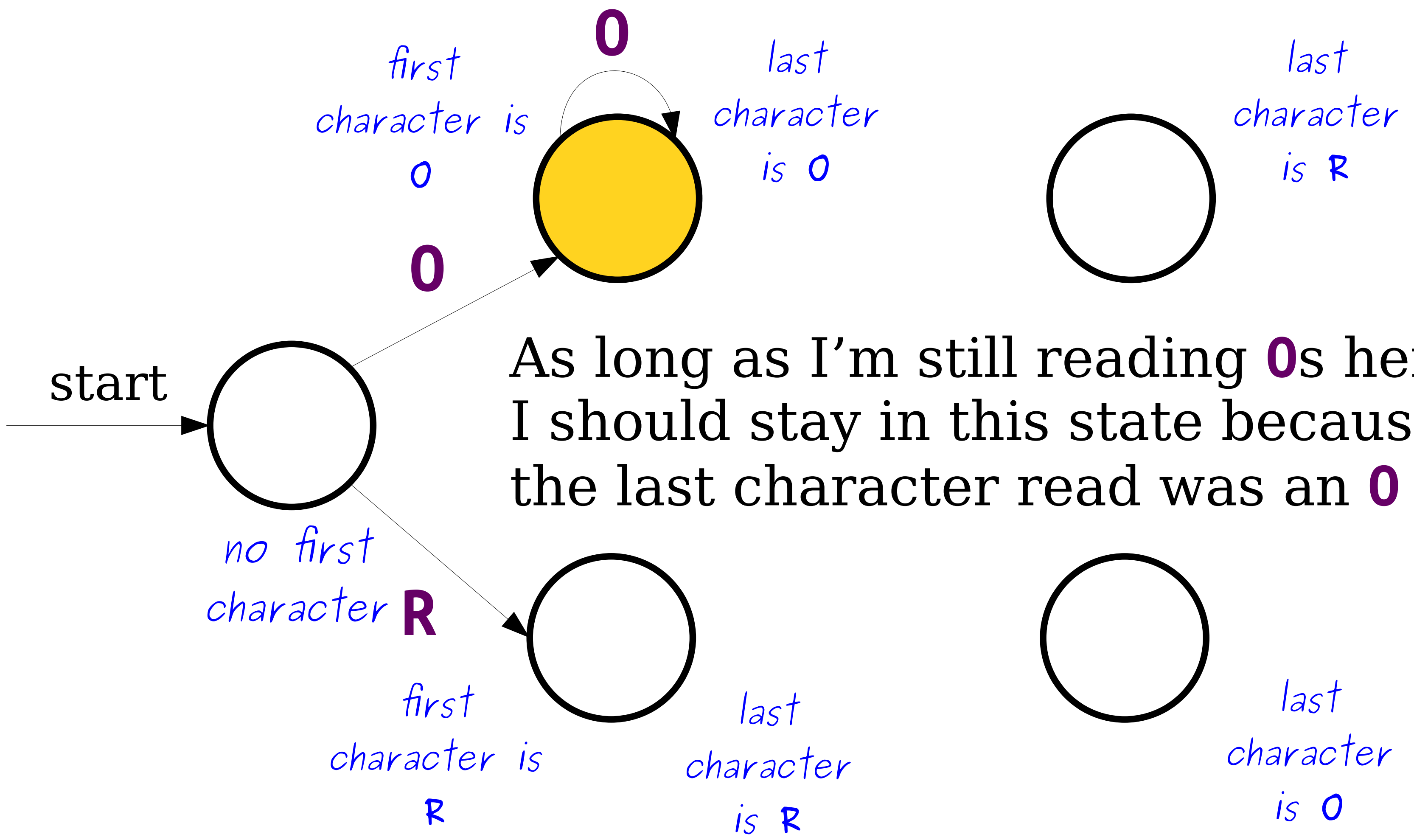
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Where should the transitions go?

Oreo Sandwiches

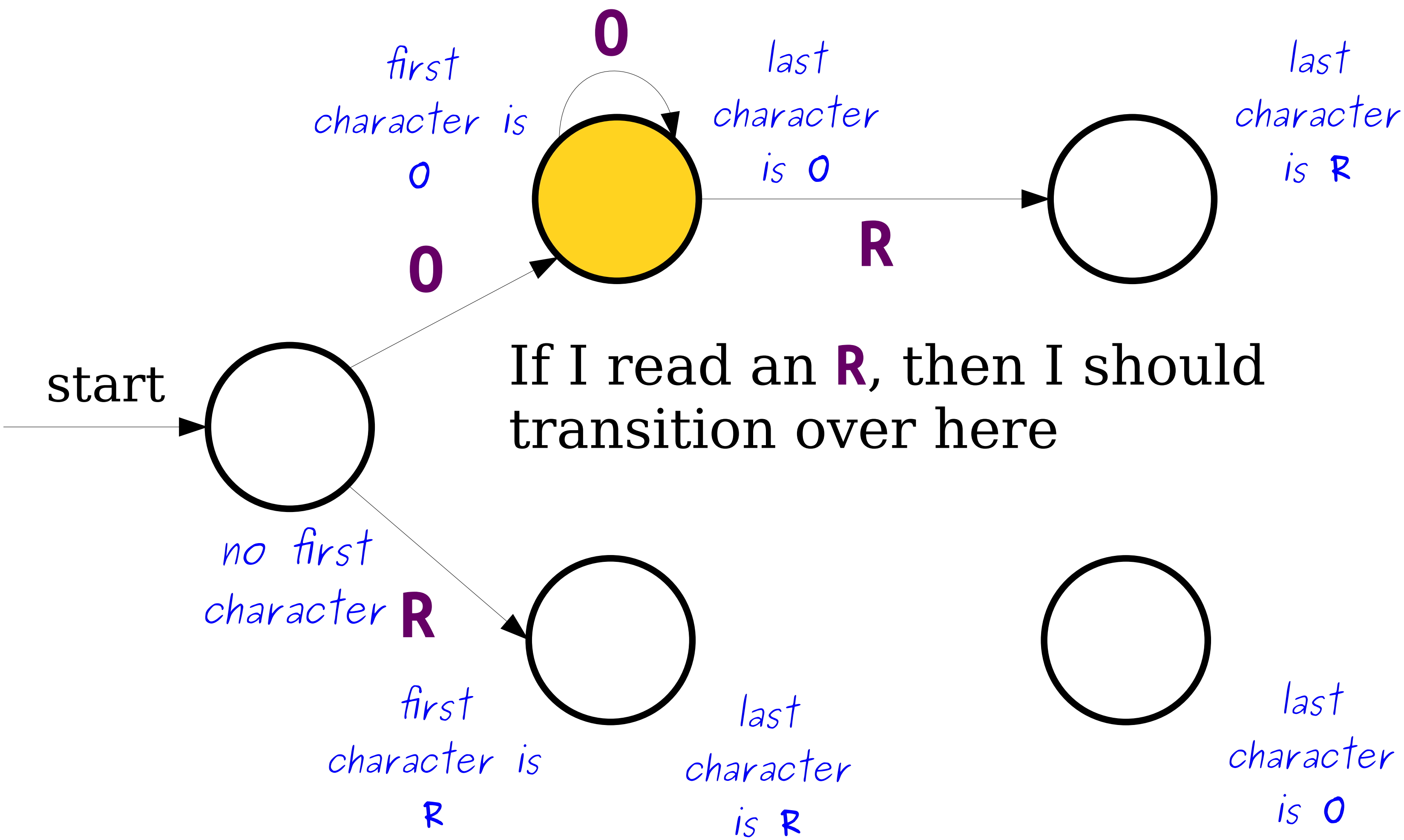
$L = \{ \, w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



As long as I'm still reading **0**s here, I should stay in this state because the last character read was an **0**

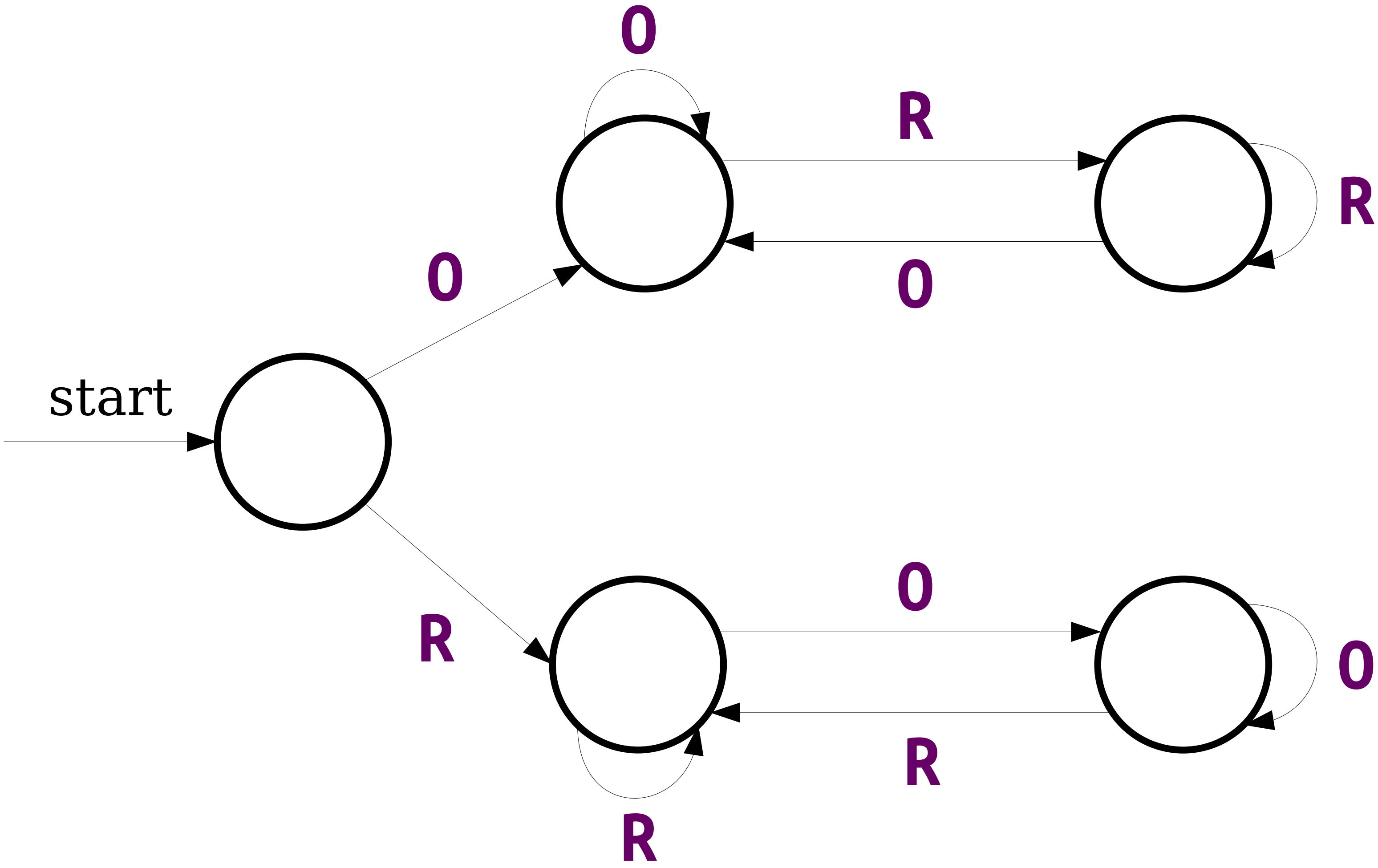
Oreo Sandwiches

$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



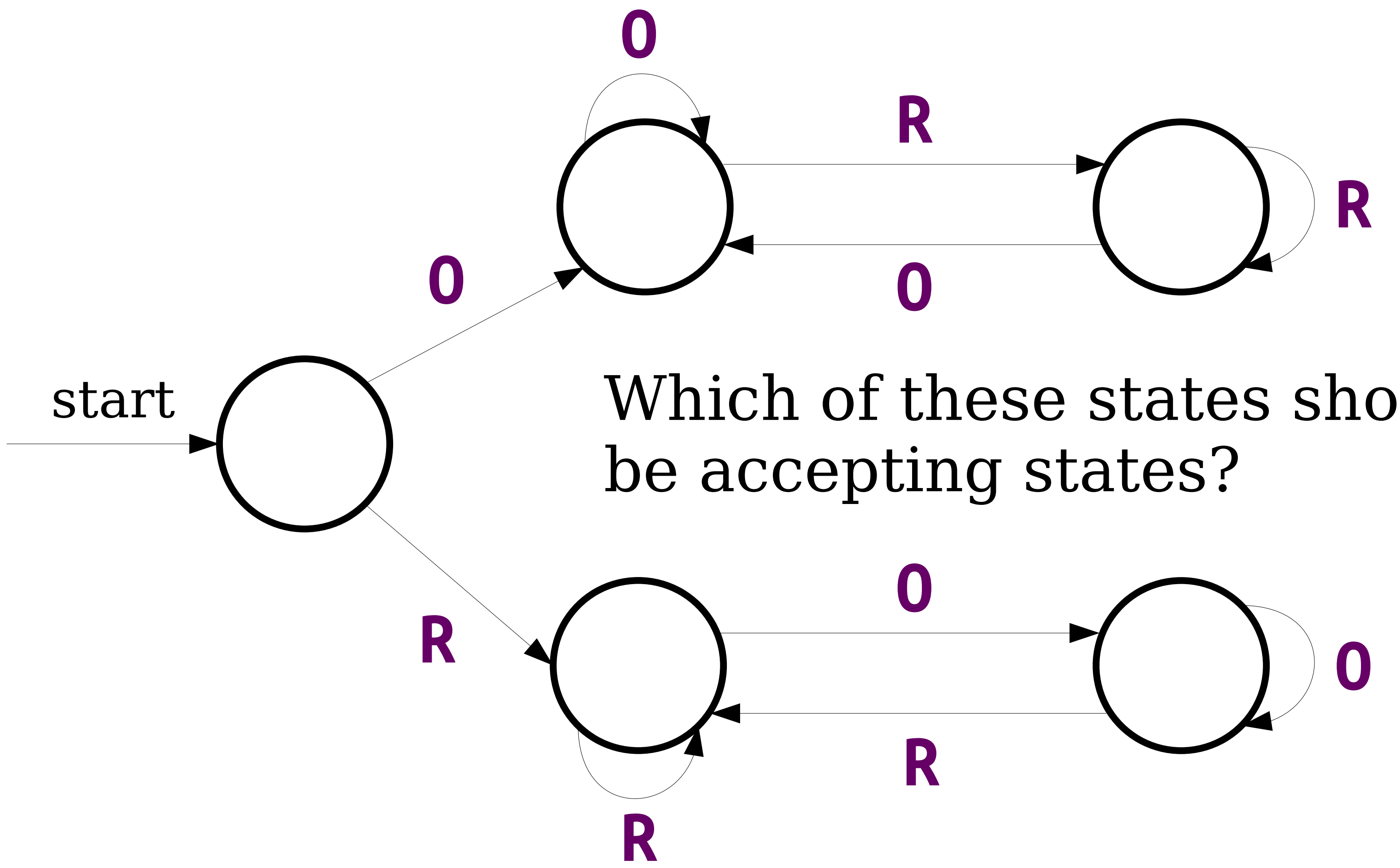
Oreo Sandwiches

$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Oreo Sandwiches

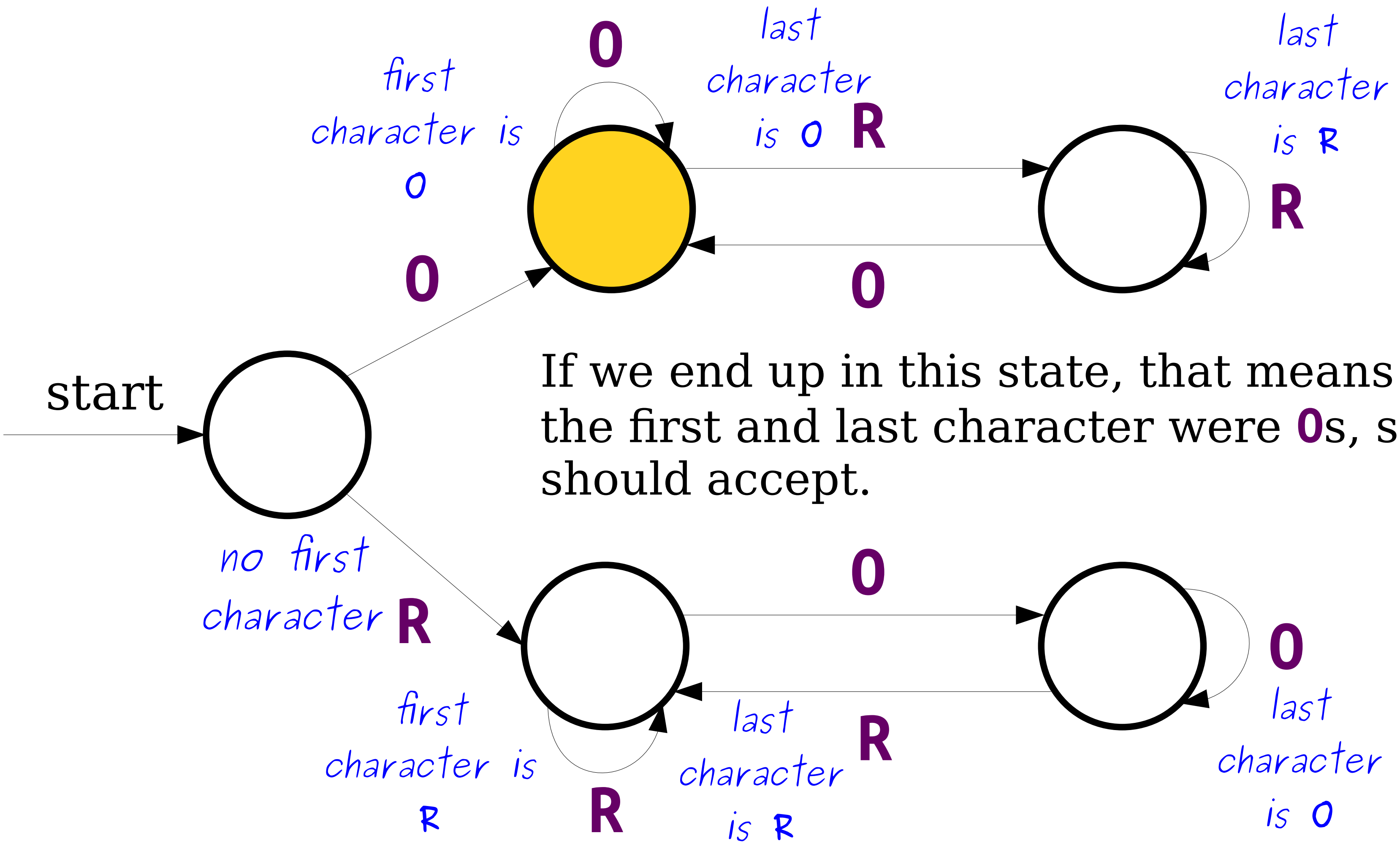
$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Which of these states should be accepting states?

Oreo Sandwiches

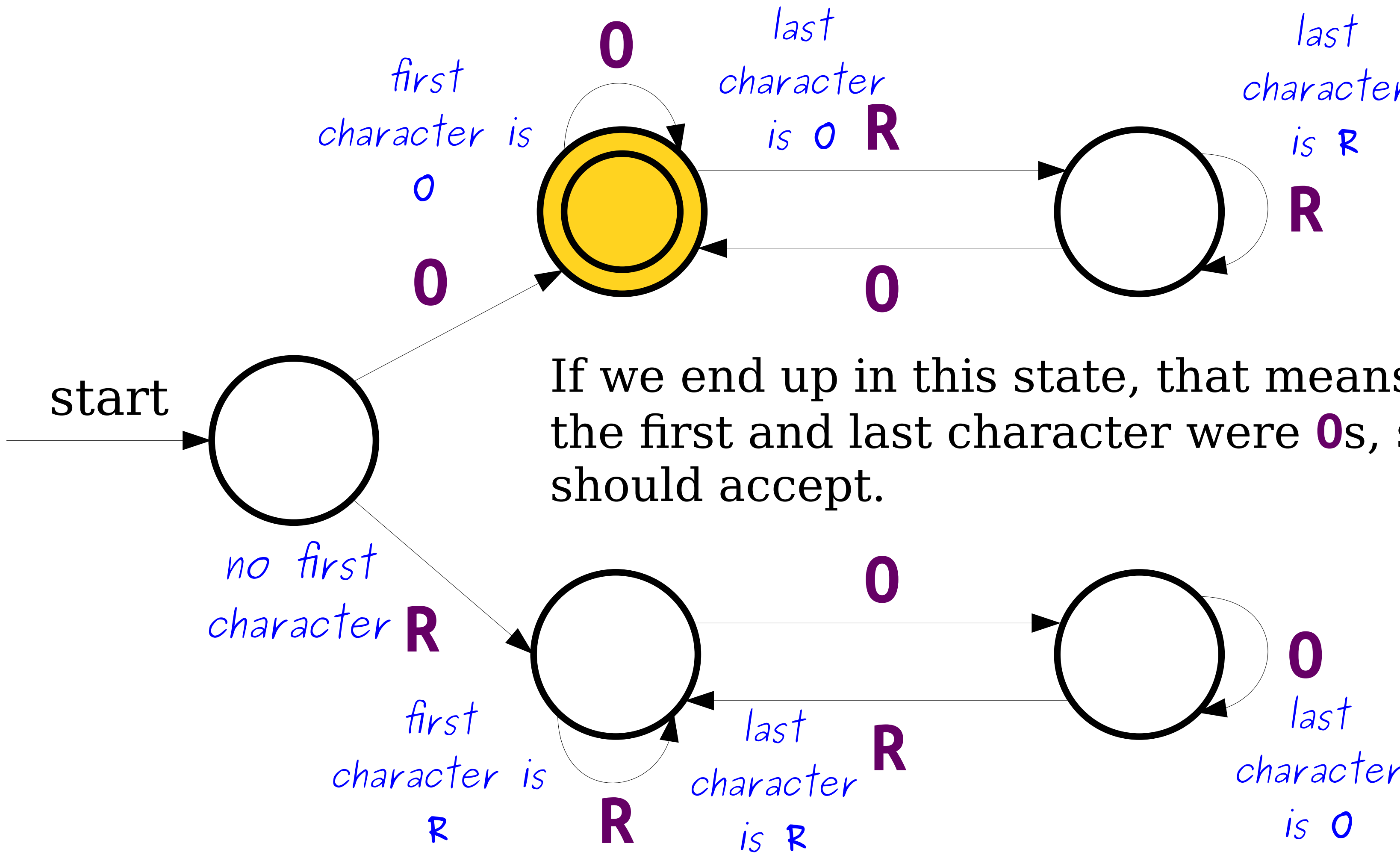
$L = \{ \, w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



If we end up in this state, that means both the first and last character were **0**s, so we should accept.

Oreo Sandwiches

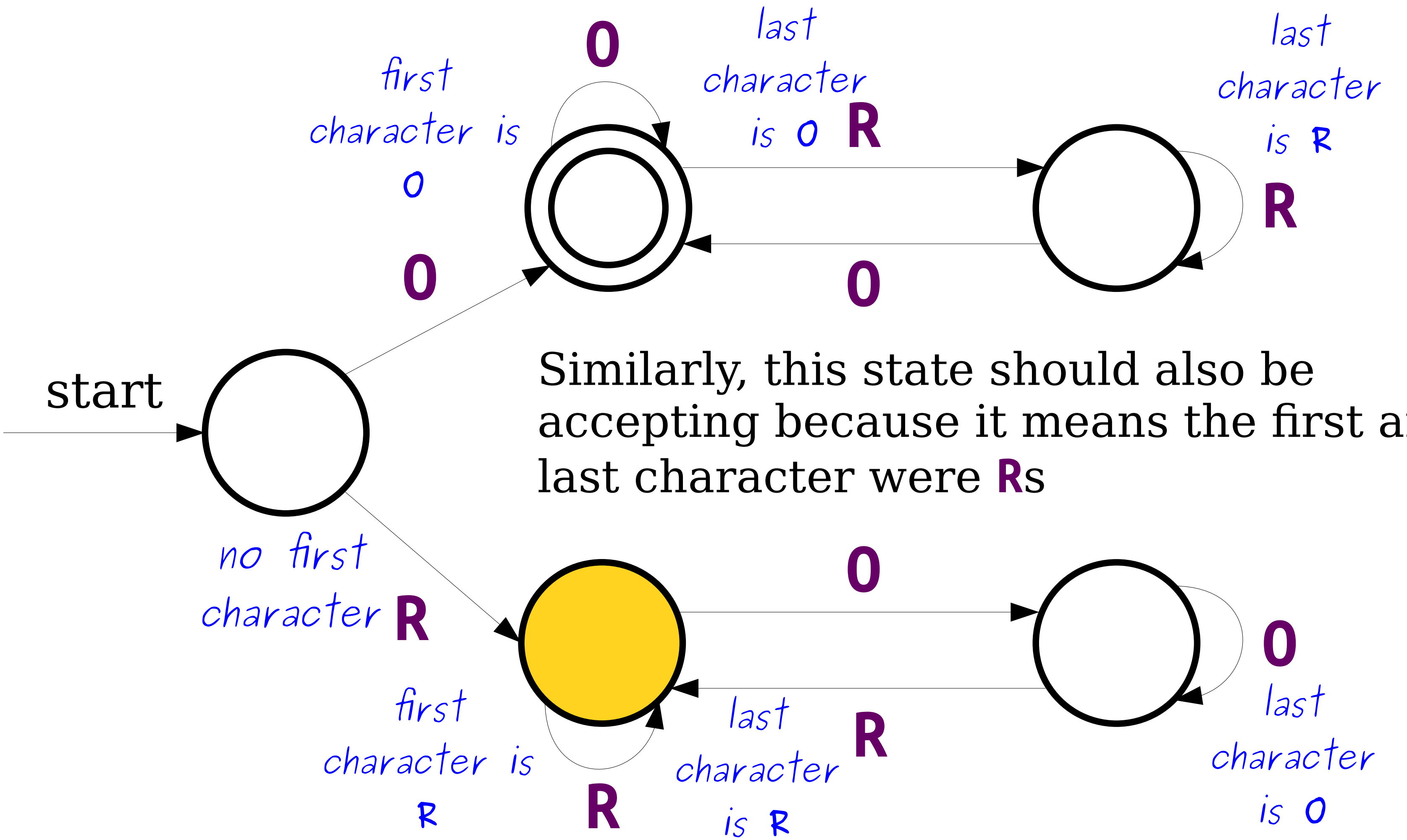
$$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same }\}$$



If we end up in this state, that means both the first and last character were **0**s, so we should accept.

Oreo Sandwiches

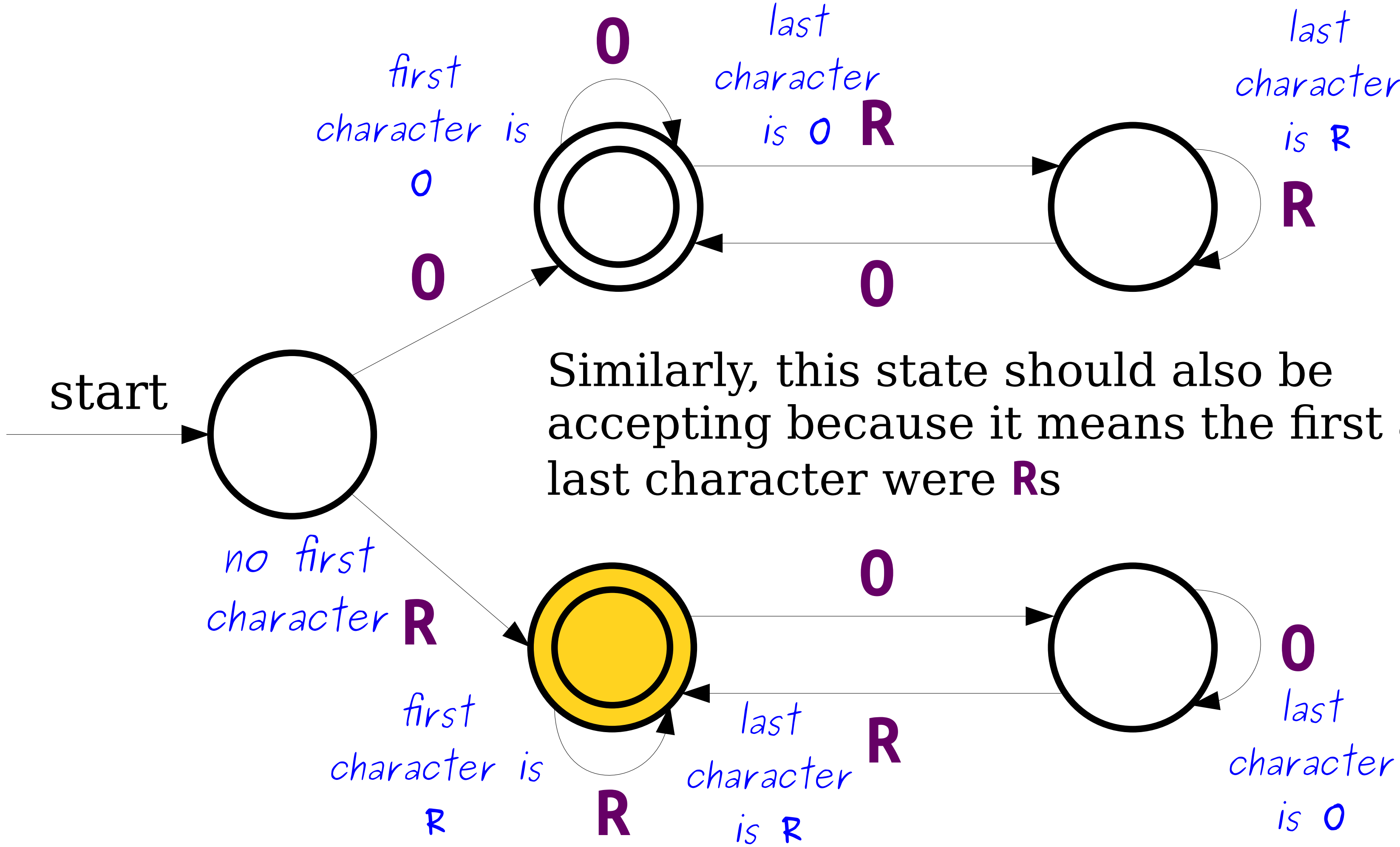
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Similarly, this state should also be accepting because it means the first and last character were **R**s

Oreo Sandwiches

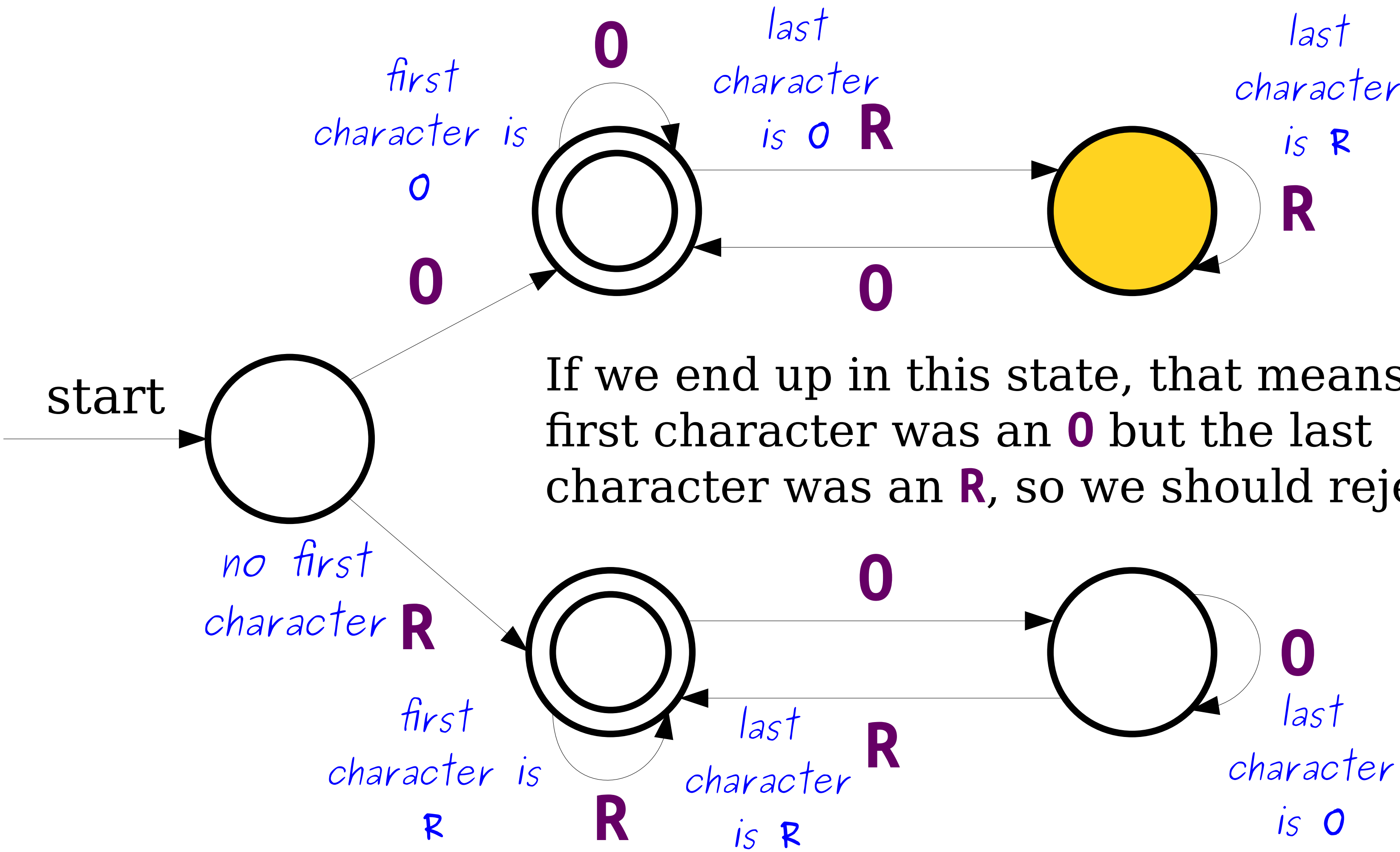
$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Similarly, this state should also be accepting because it means the first and last character were **R**s

Oreo Sandwiches

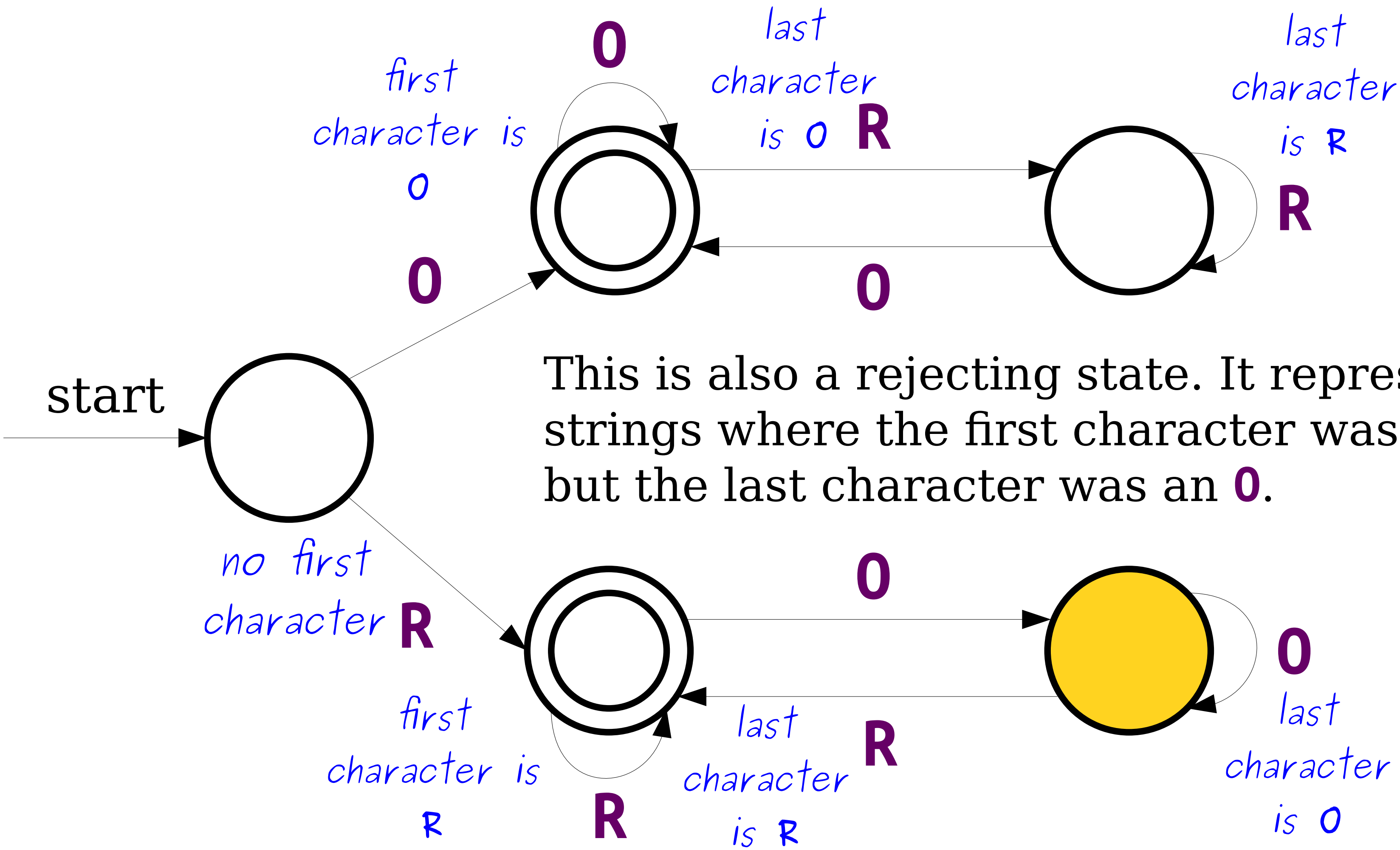
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



If we end up in this state, that means the first character was an **0** but the last character was an **R**, so we should reject.

Oreo Sandwiches

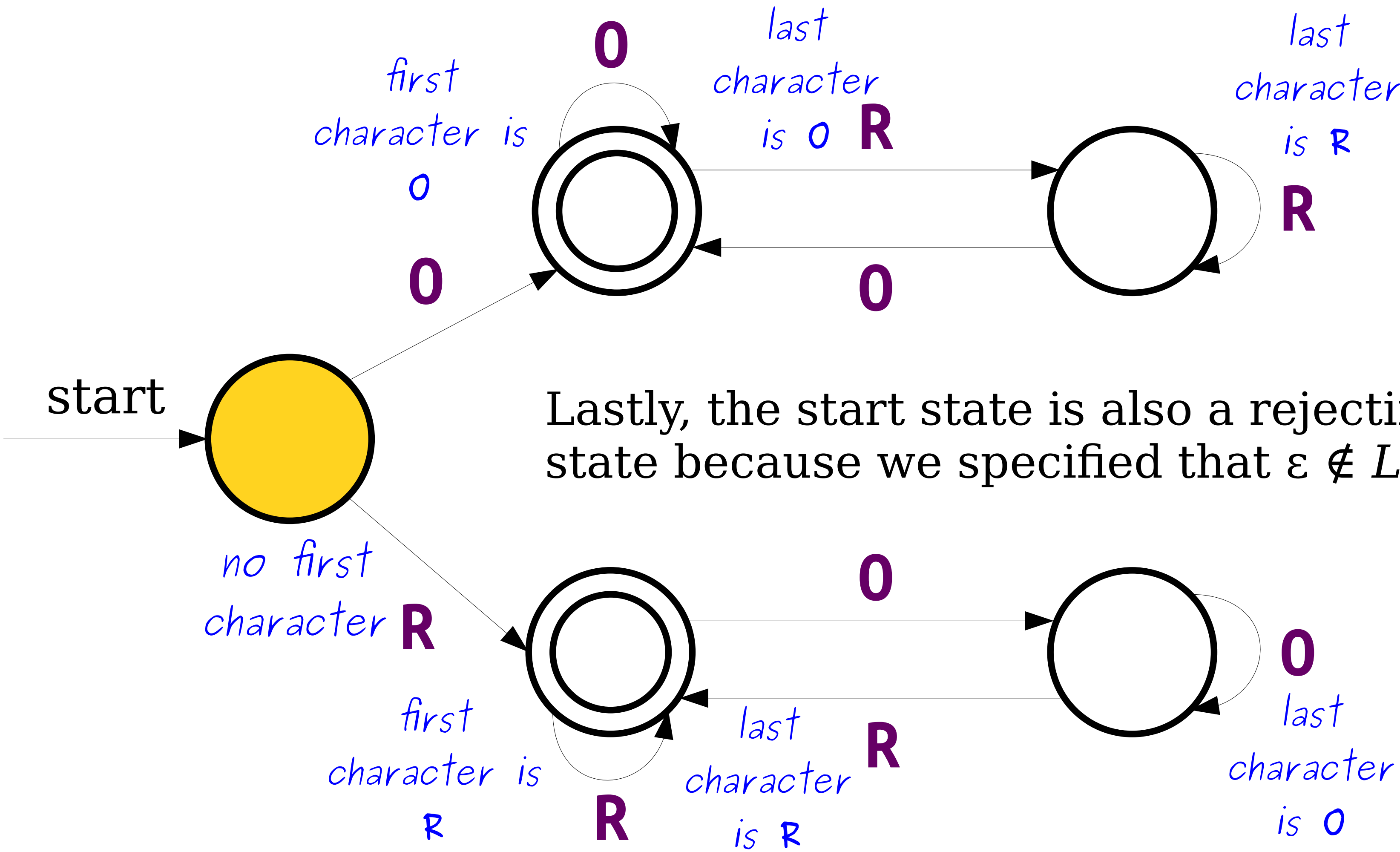
$$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}$$



This is also a rejecting state. It represents strings where the first character was an **R** but the last character was an **0**.

Oreo Sandwiches

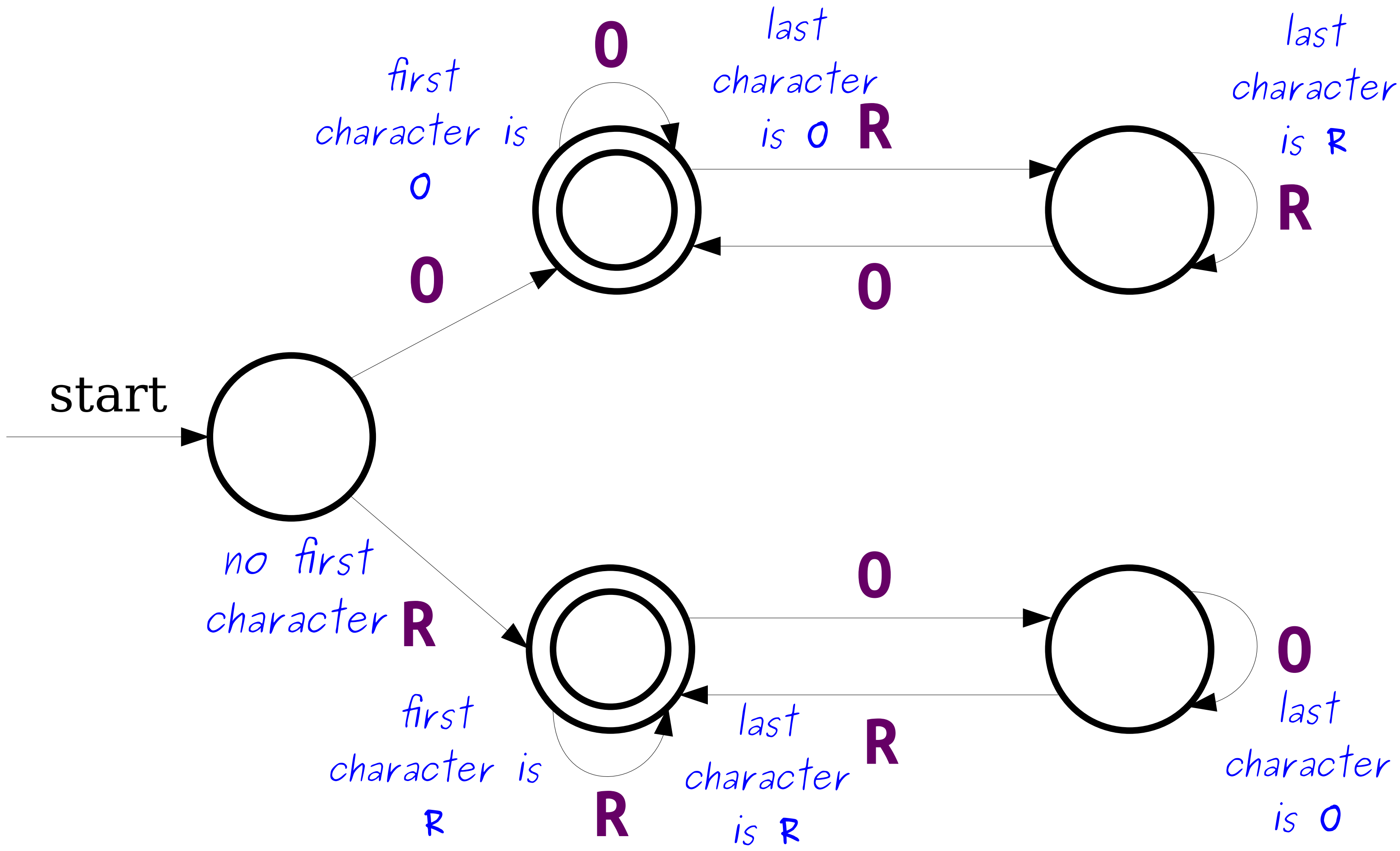
$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Lastly, the start state is also a rejecting state because we specified that $\varepsilon \notin L$

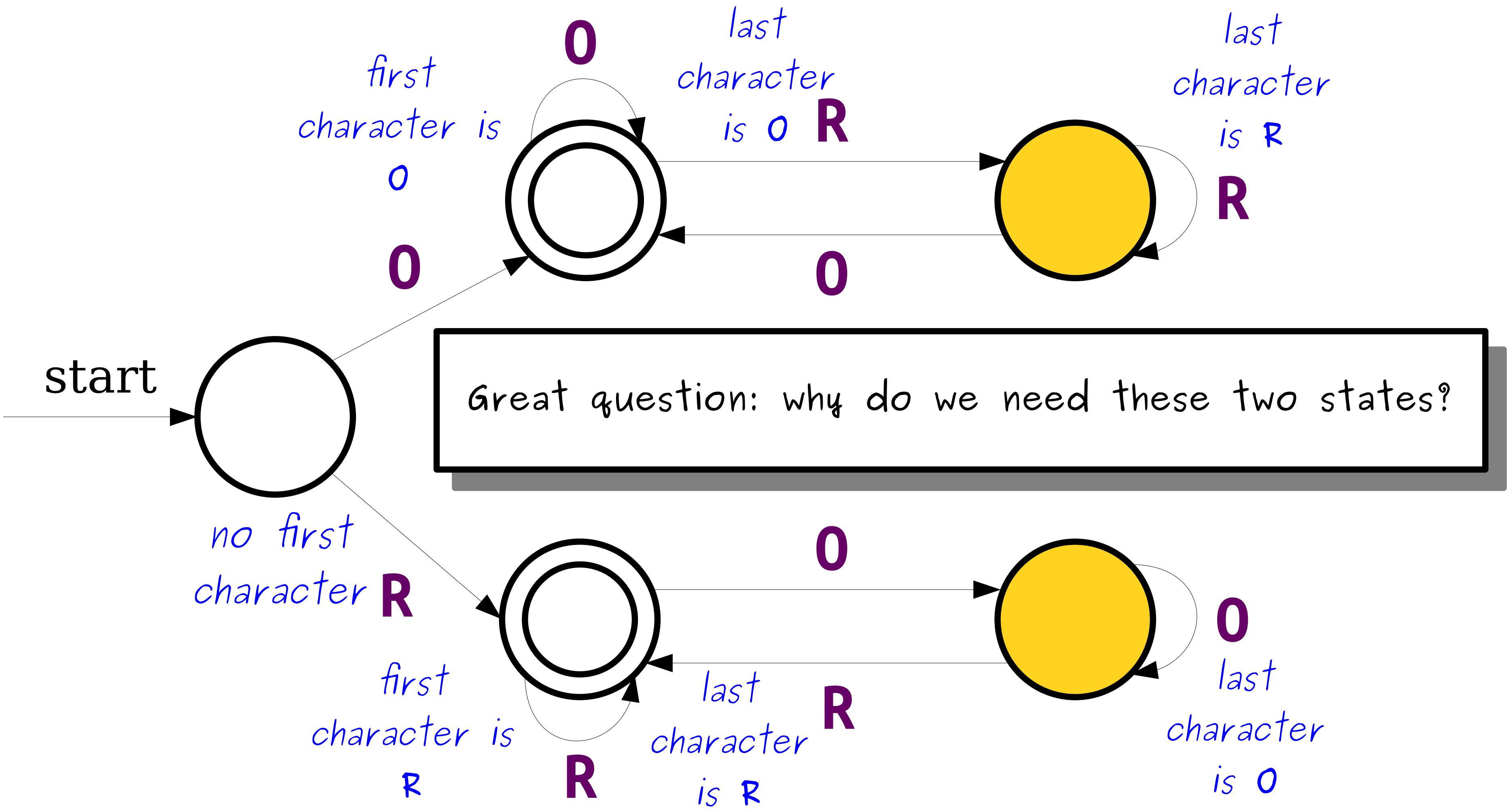
Oreo Sandwiches

$$L = \{ \ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same } \}$$



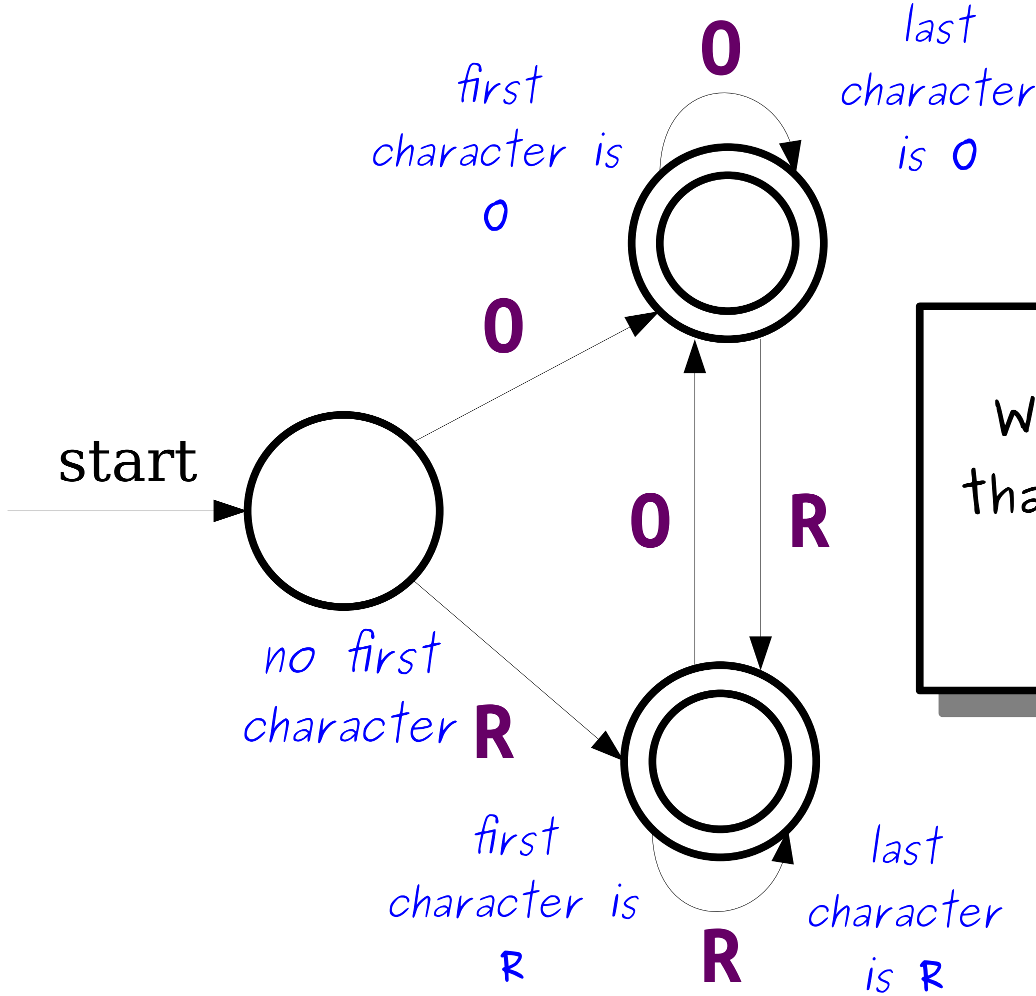
Oreo Sandwiches

$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Oreo Sandwiches

$$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon \text{ and the first and last character of } w \text{ are the same }\}$$



Why can't we have a DFA that looks like this for this language?

Part 2: ***Designing NFAs***

Designing NFAs

- Is there some information that you'd really like to have?
  - Have the machine *nondeterministically guess* that information.
  - Then, have the machine *deterministically check* that the choice was correct.

More Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design an NFA for the language

$$L = \{\, w \in \Sigma^* \mid \text{Some character of } \Sigma \text{ appears at most twice in } w \,\}$$

More Oreo Sandwiches

Let $\Sigma = \{$ **O**, **R** $\}$. Design an NFA for the language

$$L = \{\ w \in \Sigma^* \mid \text{Some character of } \Sigma \text{ appears at most twice in } w\ \}$$

$\varepsilon \in L$

R $\in L$

ORO $\in L$

RRORR $\in L$

RRROOO $\notin L$

OROORRO $\notin L$

ROROROOO $\notin L$

More Oreo Sandwiches

Let Σ = { **O**, **R** }. Design an NFA for the language

L = { $w \in \Sigma^*$ | Some character of Σ appears at most twice in w }

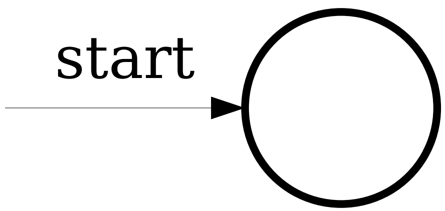
1) Draw a NFA for L using the Automaton Editor and save it as `res/TutorialWeek7.Q2.automaton`

(Hint: What would you do if you knew which character was going to appear at most twice?)

Then, submit `res/TutorialWeek7.Q1.automaton` and `res/TutorialWeek7.Q2.automaton` to Gradescope.

More Oreo Sandwiches

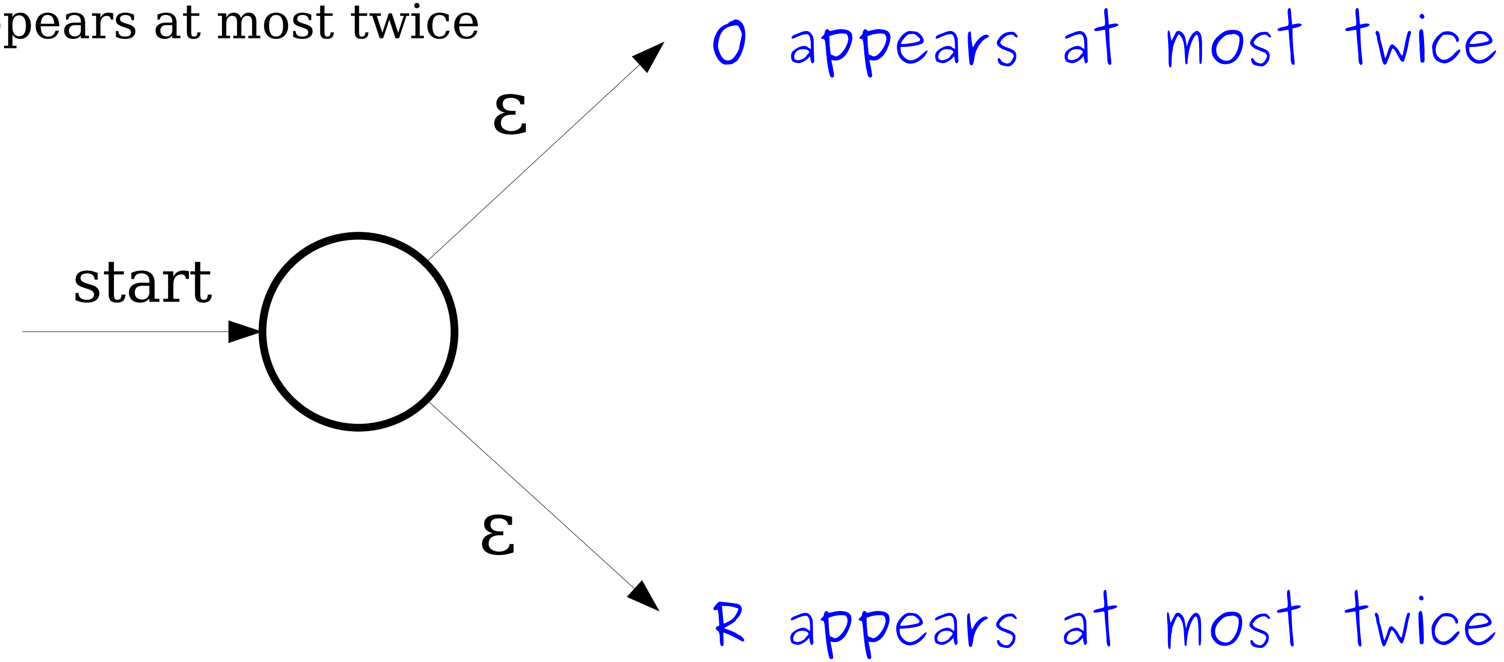
$L = \{ w \in \Sigma^* \mid$ Some character of Σ appears at most twice in $w \}$



More Oreo Sandwiches

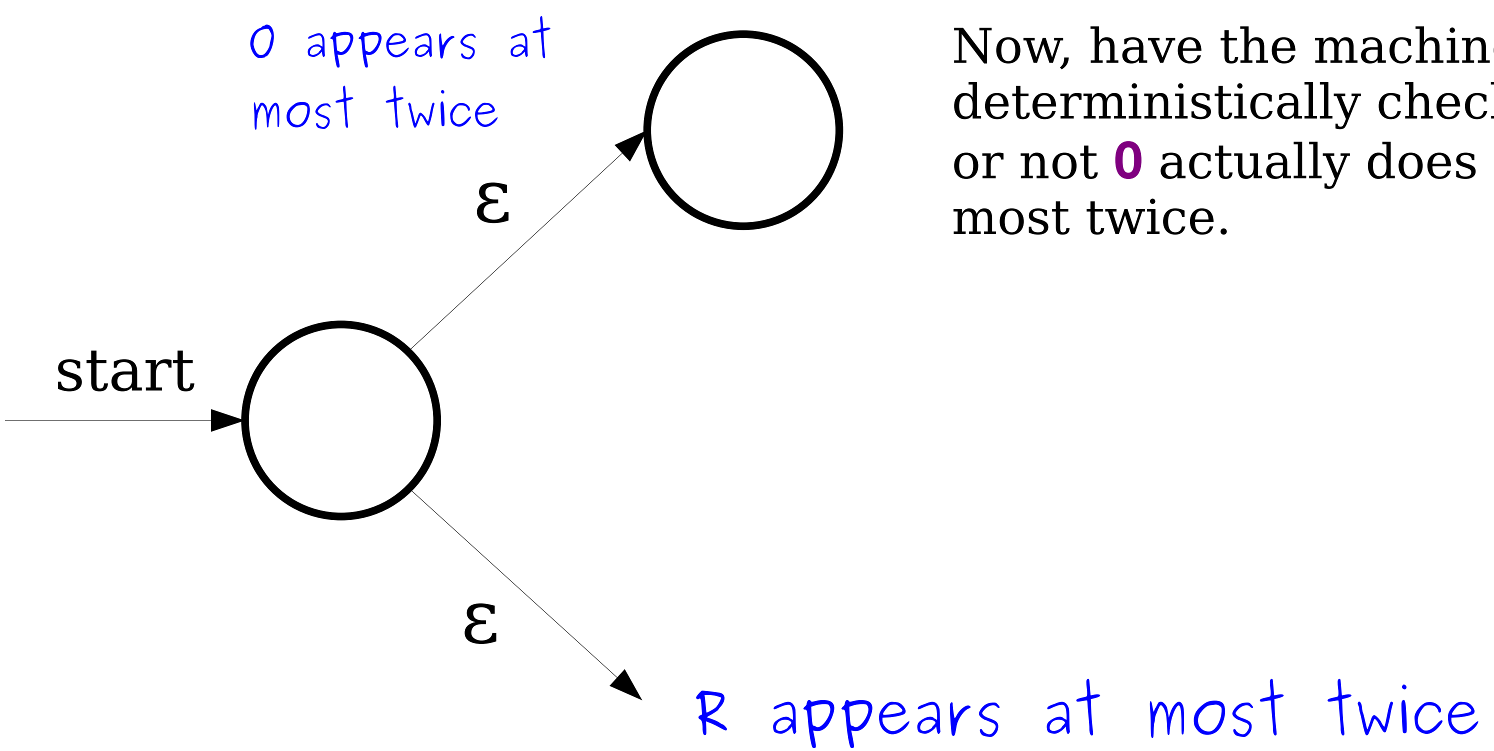
$$L = \{ w \in \Sigma^* \mid \text{Some character of } \Sigma \text{ appears at most twice in } w \}$$

Have the machine nondeterministically guess which character appears at most twice



More Oreo Sandwiches

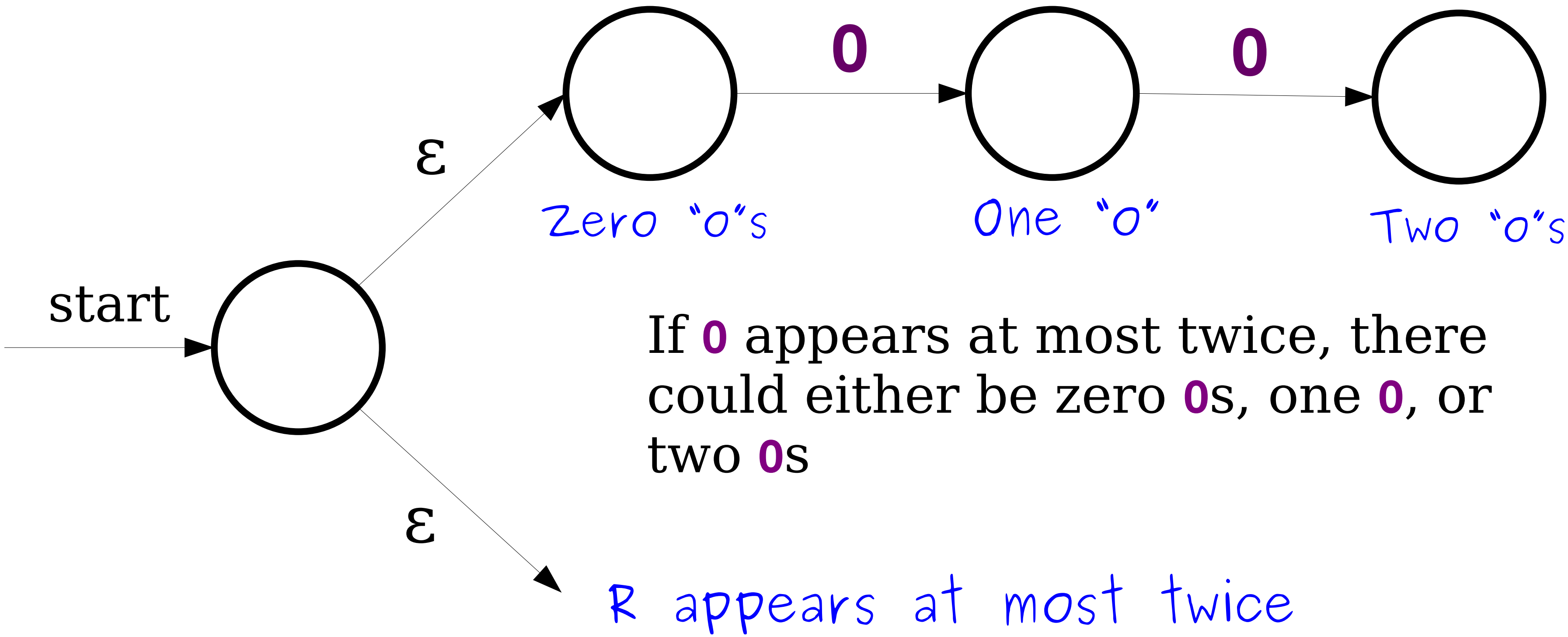
$$L = \{\ w \in \Sigma^* \mid \text{Some character of } \Sigma \text{ appears at most twice in } w\ \}$$



Now, have the machine deterministically check whether or not **0** actually does appear at most twice.

More Oreo Sandwiches

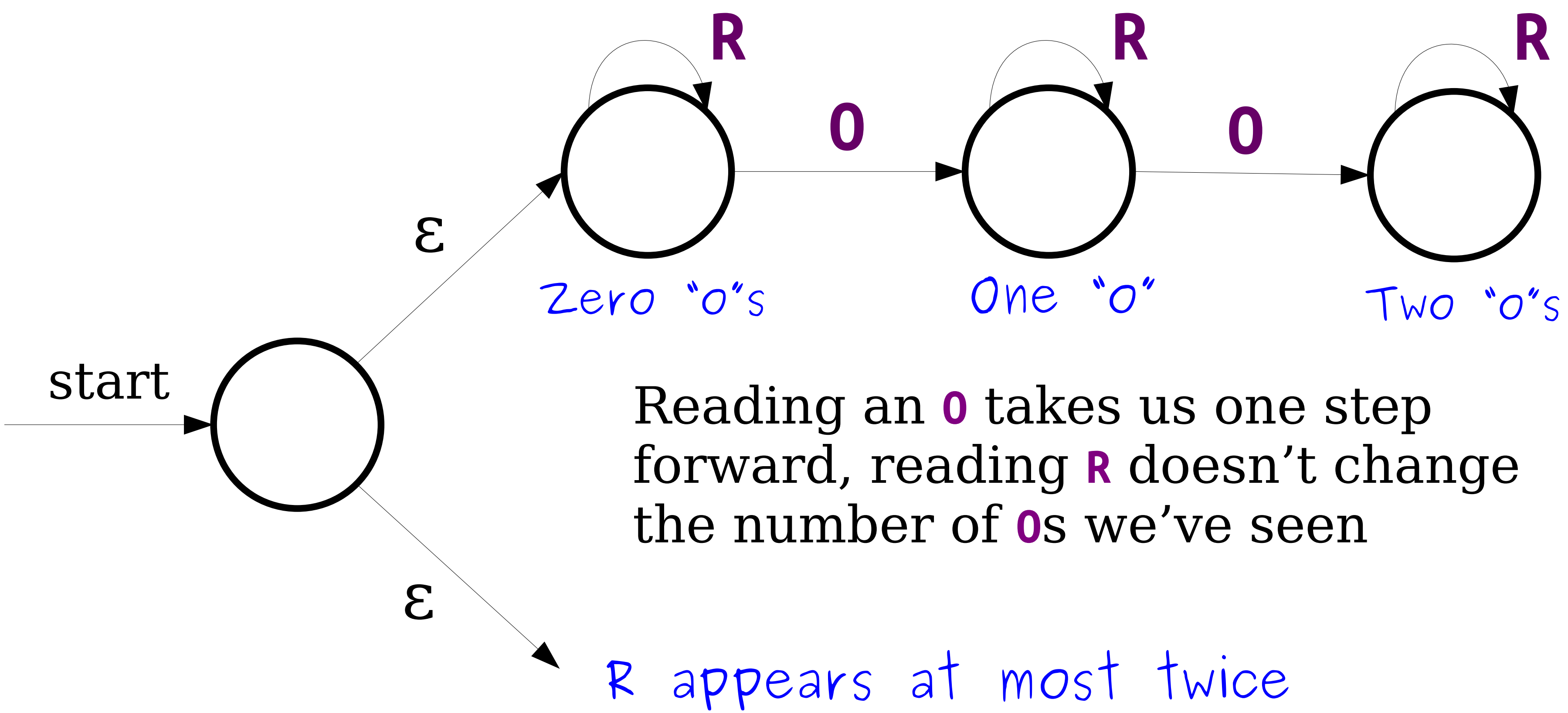
$L = \{ w \in \Sigma^* \mid$ Some character of Σ appears at most twice in $w \}$



If **0** appears at most twice, there could either be zero **0**s, one **0**, or two **0**s

More Oreo Sandwiches

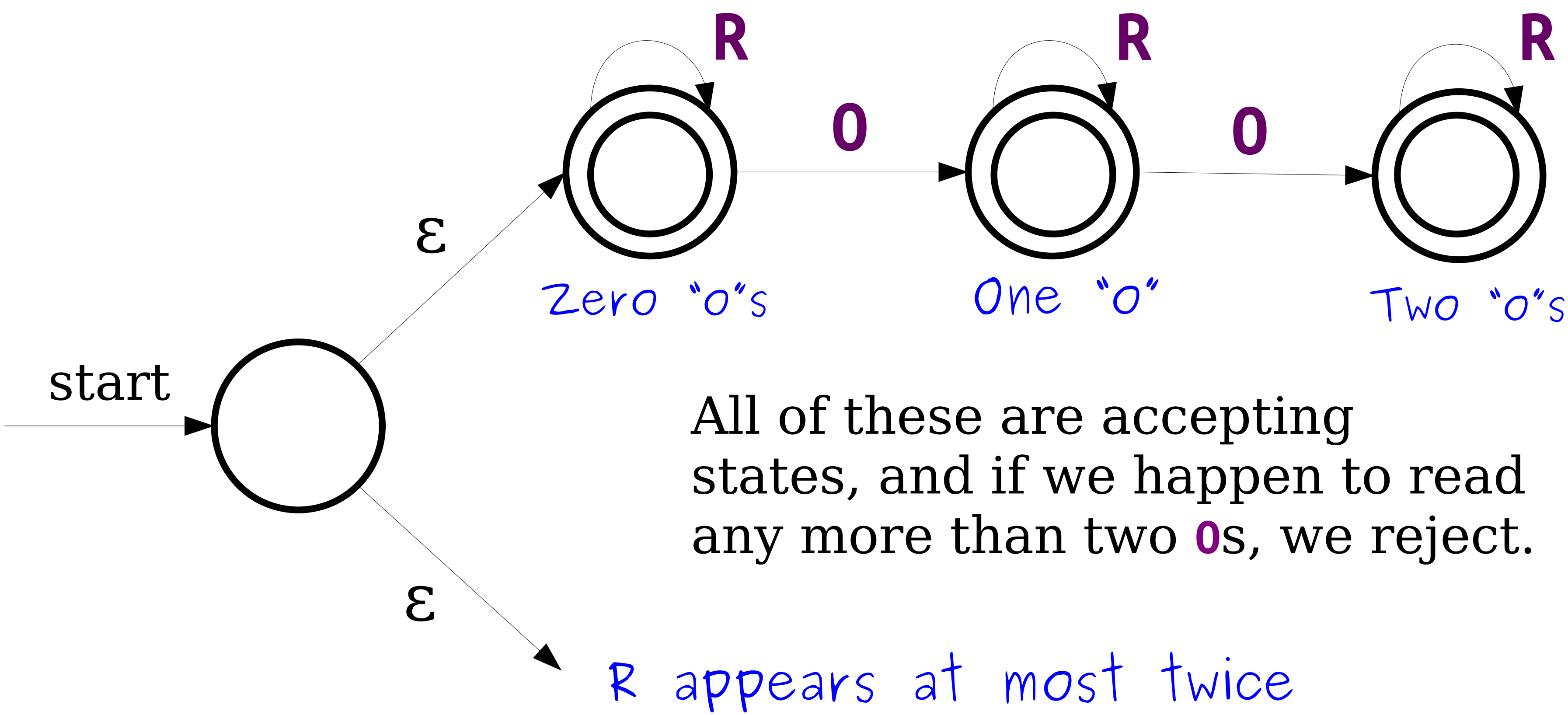
$L = \{ w \in \Sigma^* \mid \text{Some character of } \Sigma \text{ appears at most twice in } w \}$



Reading an **0** takes us one step forward, reading **R** doesn't change the number of **0**s we've seen

More Oreo Sandwiches

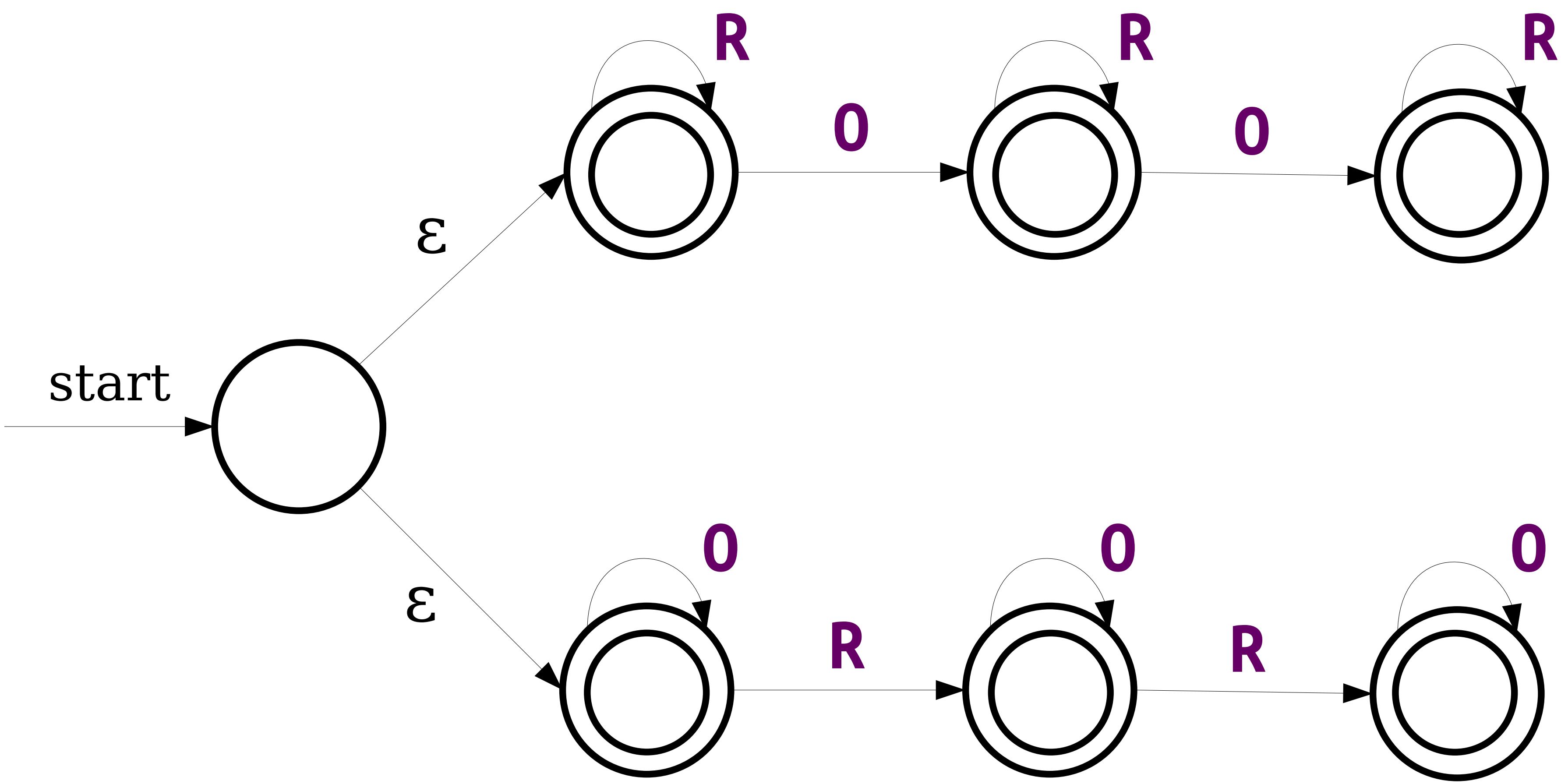
$L = \{ w \in \Sigma^* \mid$ Some character of Σ appears at most twice in w $\}$



All of these are accepting states, and if we happen to read any more than two **0**s, we reject.

More Oreo Sandwiches

$L = \{ w \in \Sigma^* \mid$ Some character of Σ appears at most twice in w $\}$



Thanks for Calling In!

Stay safe, stay healthy,
and have a good week!

See you next time.