Regular Languages

Art by 白砂糖


OREO
O&REO
O&O
OREOREO
RERERERERE
OOOOO
OREOO
OREOREREREORE
OREOREORE
REREO
REORE
OREREREREREREREREO
OOOREREREREREREOOO
OREREREREOOOOOOOOO

Typeset by Ree
Translated by Mysterious Uploader (danbooru)

Oreo Sandwiches

- Let Σ = { **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** $\in L$
- **ROOOR** $\in L$
- **OROORORRO** $\in L$

- **OR** $\notin L$
- **OOOOOR** $\notin L$
- **ROROROROO** $\notin L$

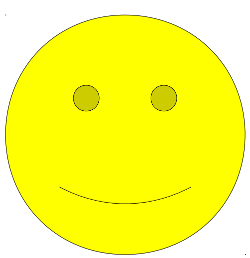
Designing DFAs

- ***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

$L = \{ w \text{ is divisible by } 5 \}$



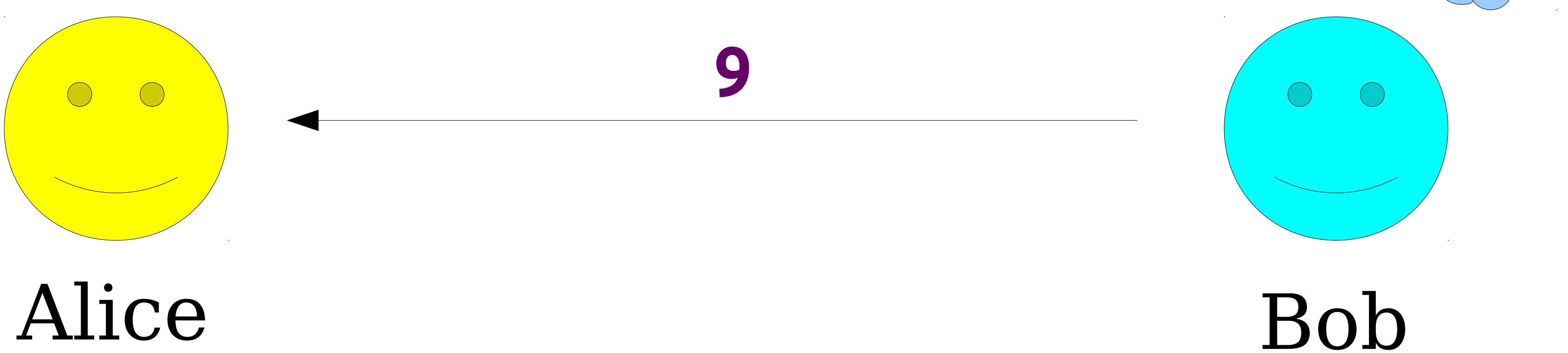
Alice

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

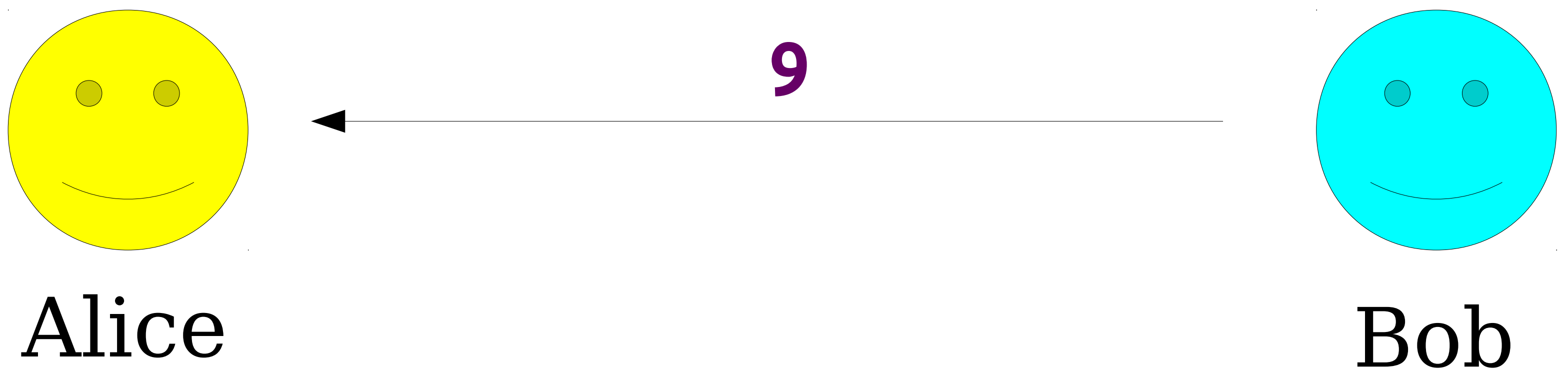
$L = \{ \ w \text{ is divisible by } 5 \ \}$



An Analogy

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

$L = \{ w \text{ is divisible by } 5 \}$



An Analogy

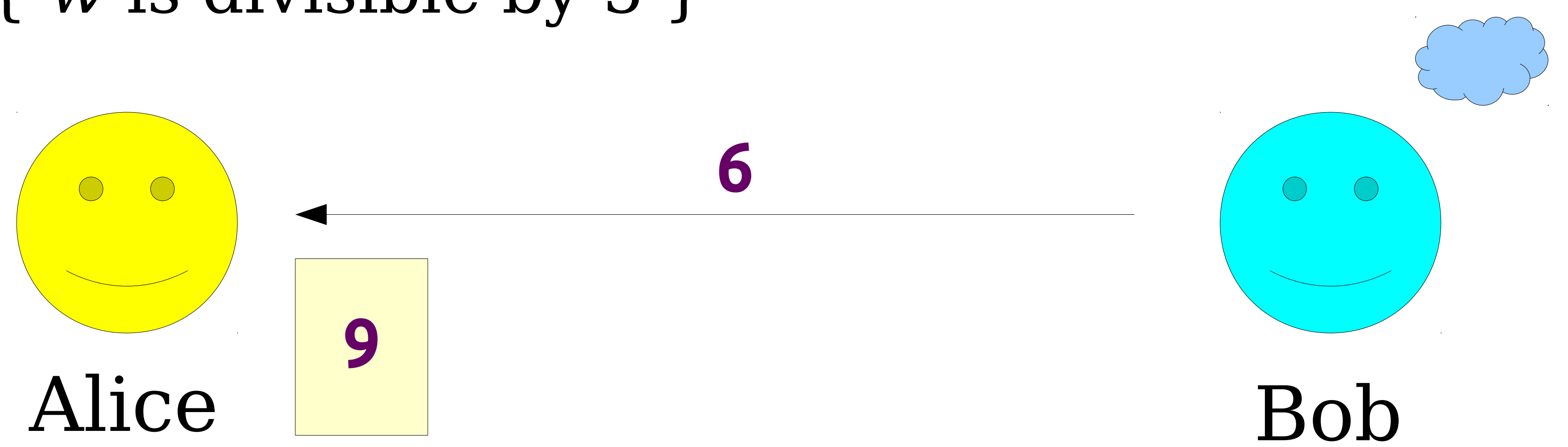
Key insight: Alice only needs to remember the last character she received from Bob

$L = \{\ w \text{ is divisible by } 5\ \}$

An Analogy

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

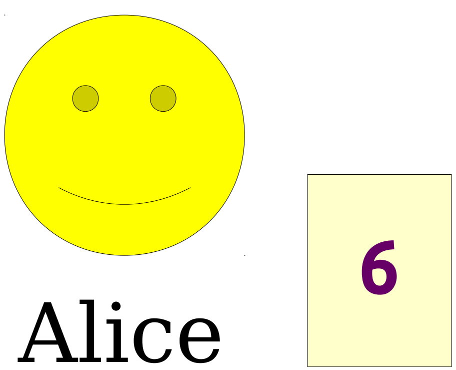
$L = \{\ w$ is divisible by 5 $\}$

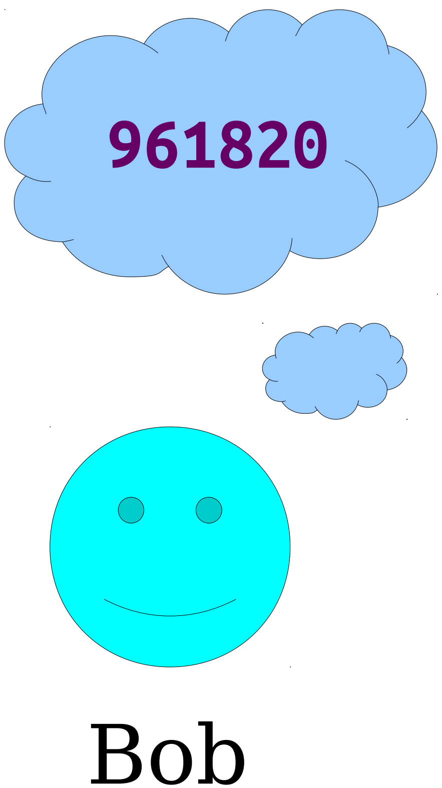


An Analogy

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

$L = \{ w \text{ is divisible by } 5 \}$

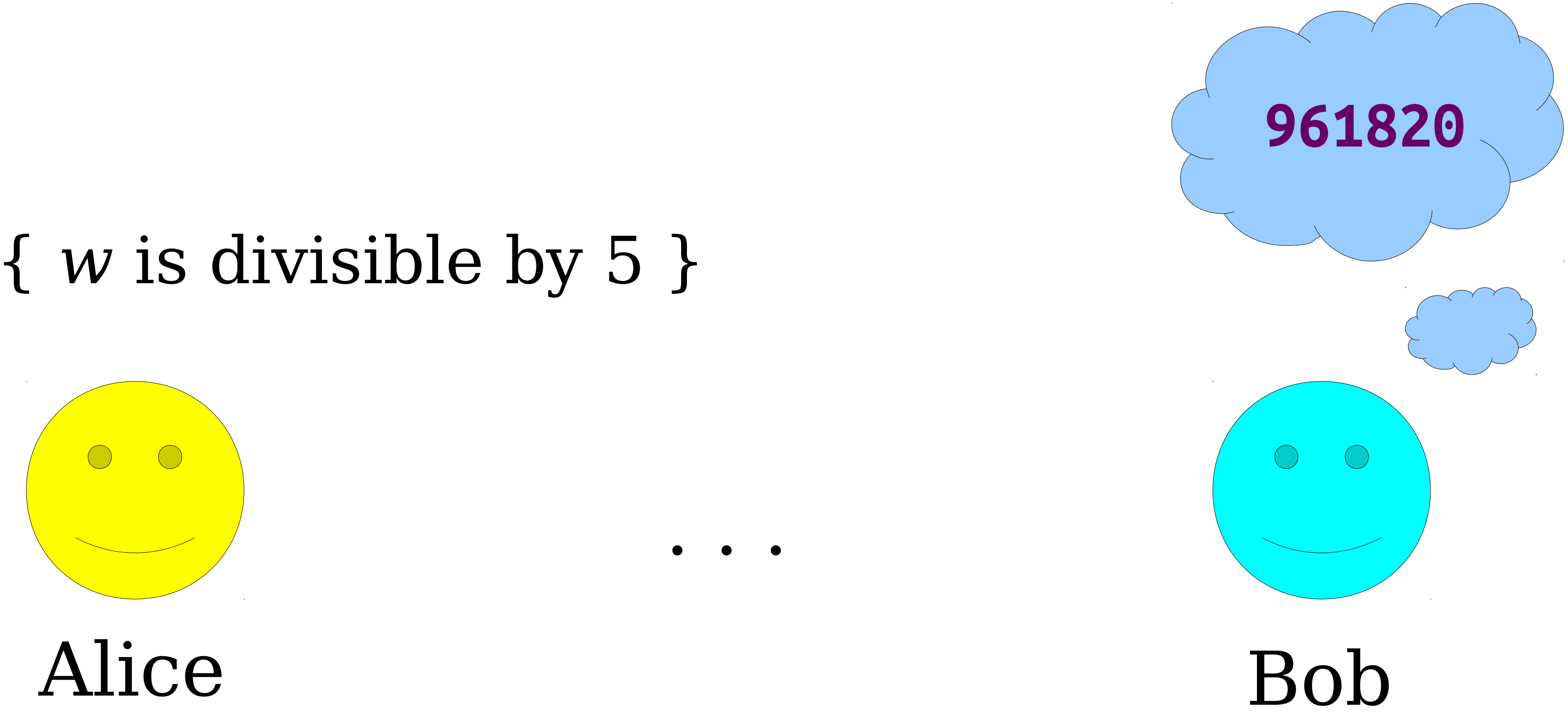




An Analogy

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

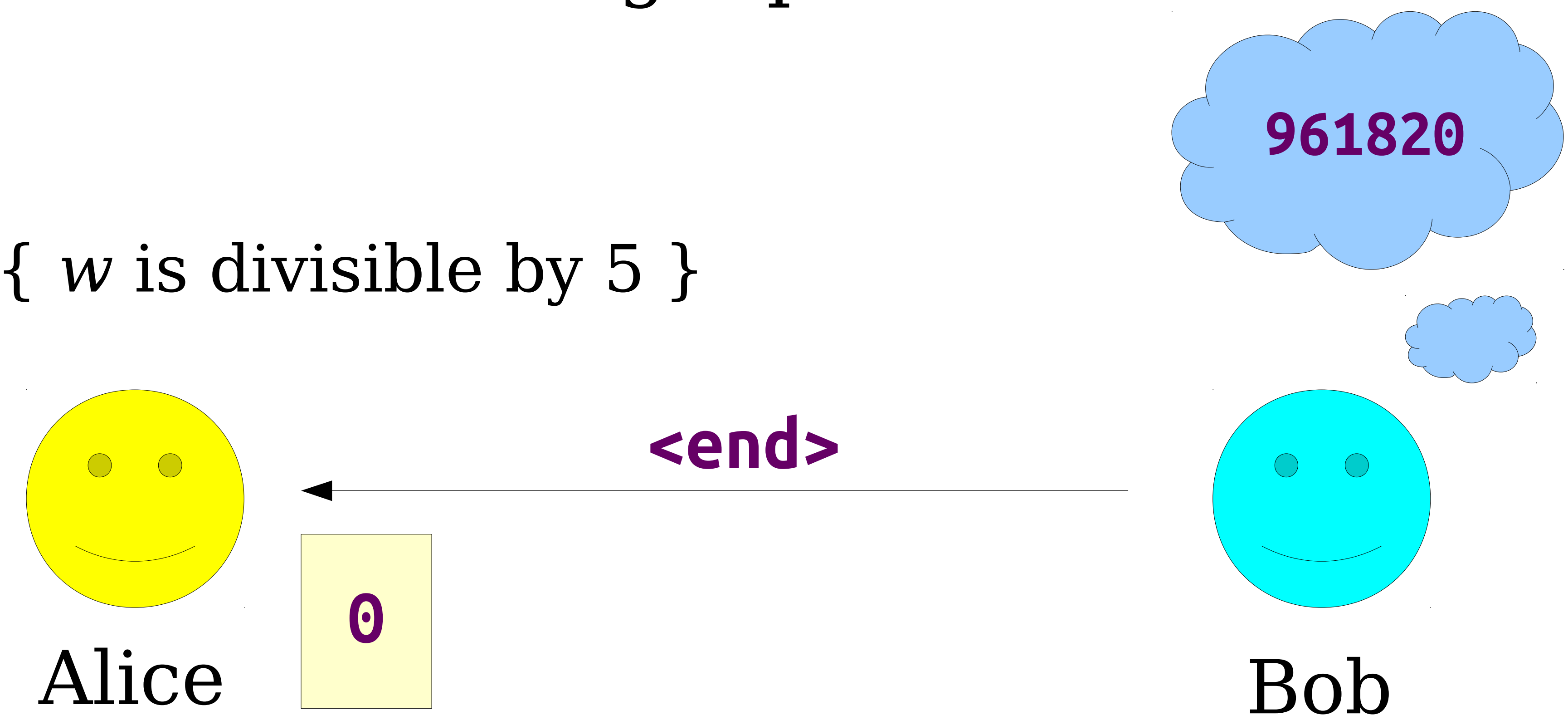
$L = \{ w \text{ is divisible by } 5 \}$



An Analogy

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

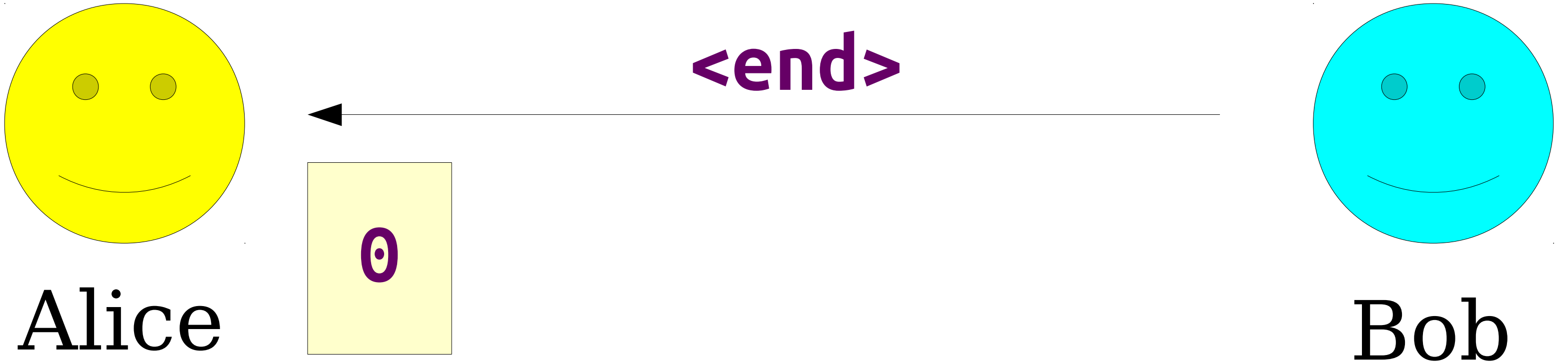
$L = \{\ w \text{ is divisible by } 5\ \}$



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

$L = \{\ w \text{ is divisible by } 5\ \}$



DFA Design Strategy

1. Answer the question “What do I have to keep track of in the course of figuring out whether a string is in this language?”

2. Create a state that represents each possible answer to that question.

3. 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.

DFA Design Strategy

$L = \{ w \text{ is divisible by } 5 \}$

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

We need to keep track of the last character.

2. Create a state that represents each possible answer to that question.

The last character could be any digit 0–9. The states for 0 and 5 are accepting states.

3. 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.

Reading a character *d* should transition to the state representing "the last character of the string is *d*".

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?

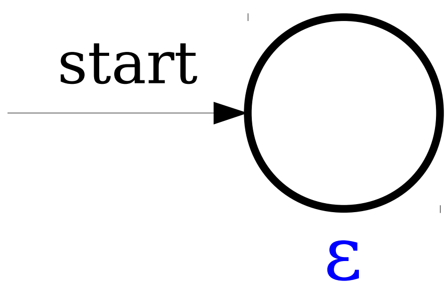
Oreo Sandwiches

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

- 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

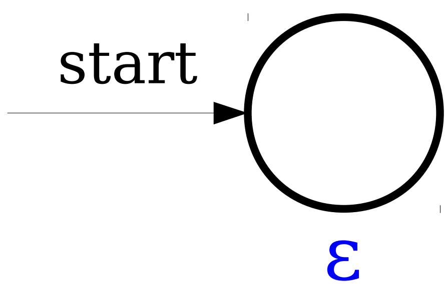
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

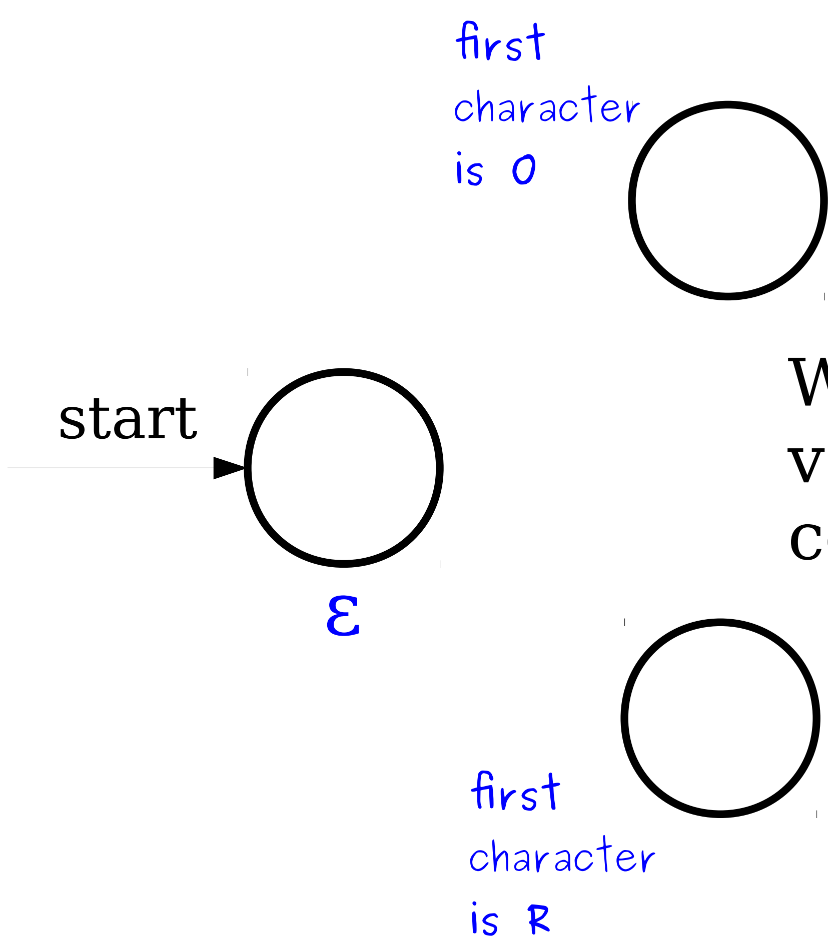
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

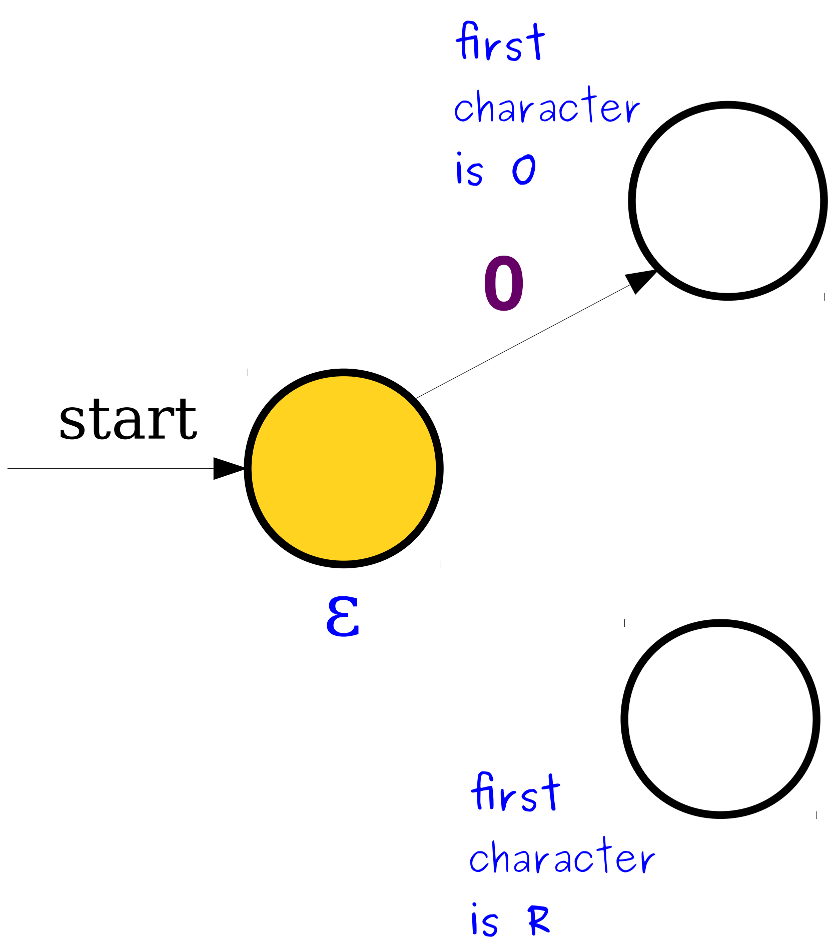
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

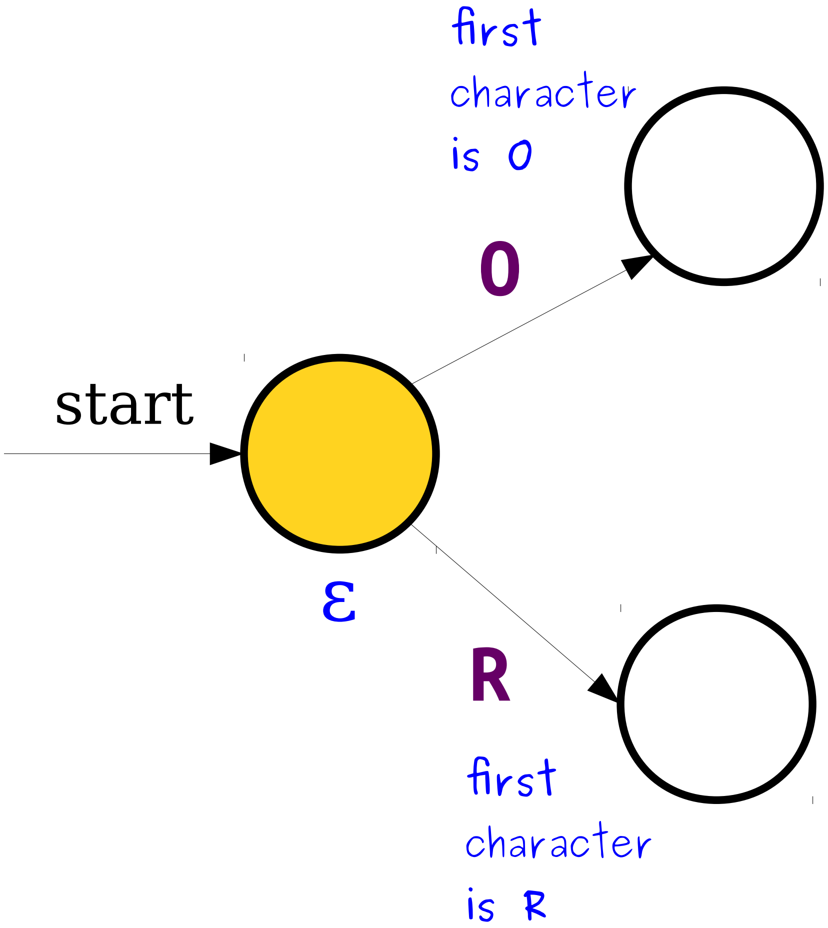
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

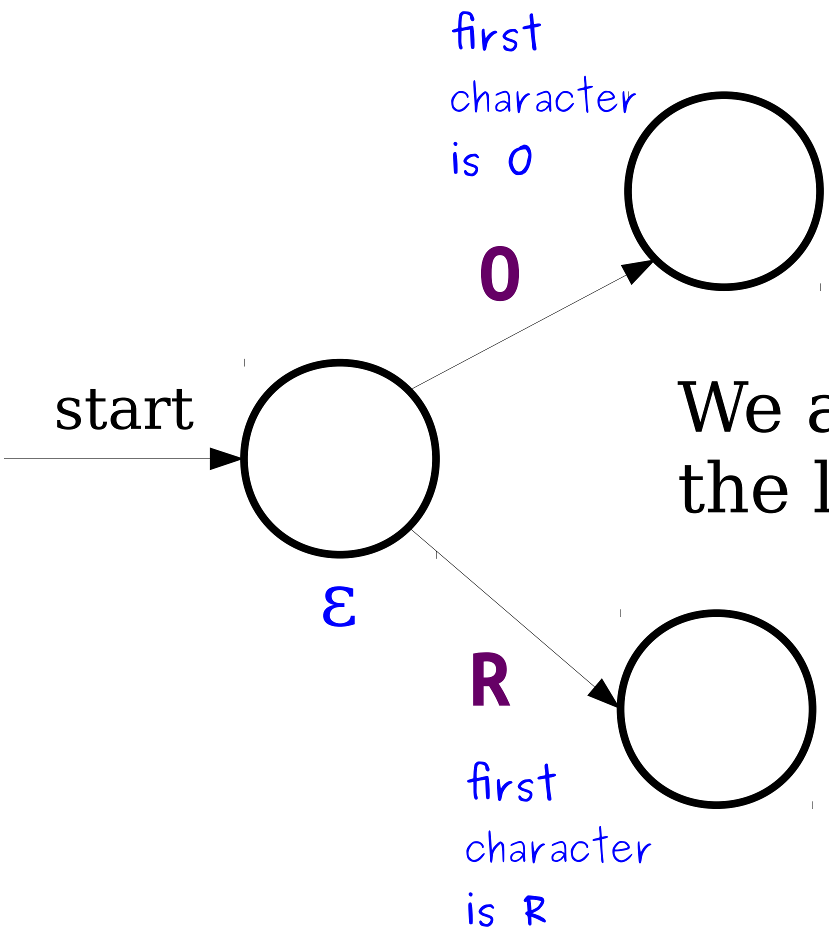
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

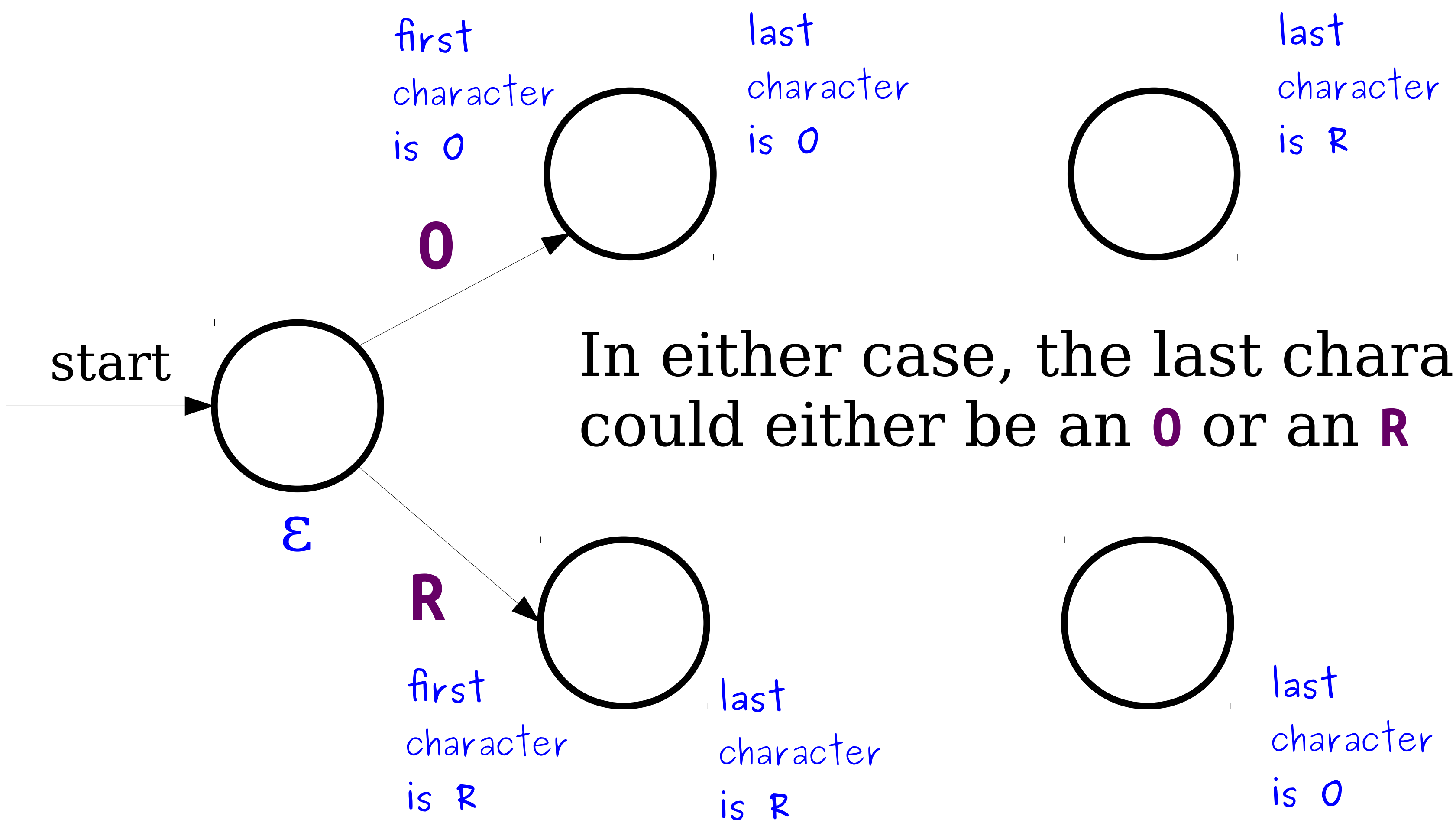
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

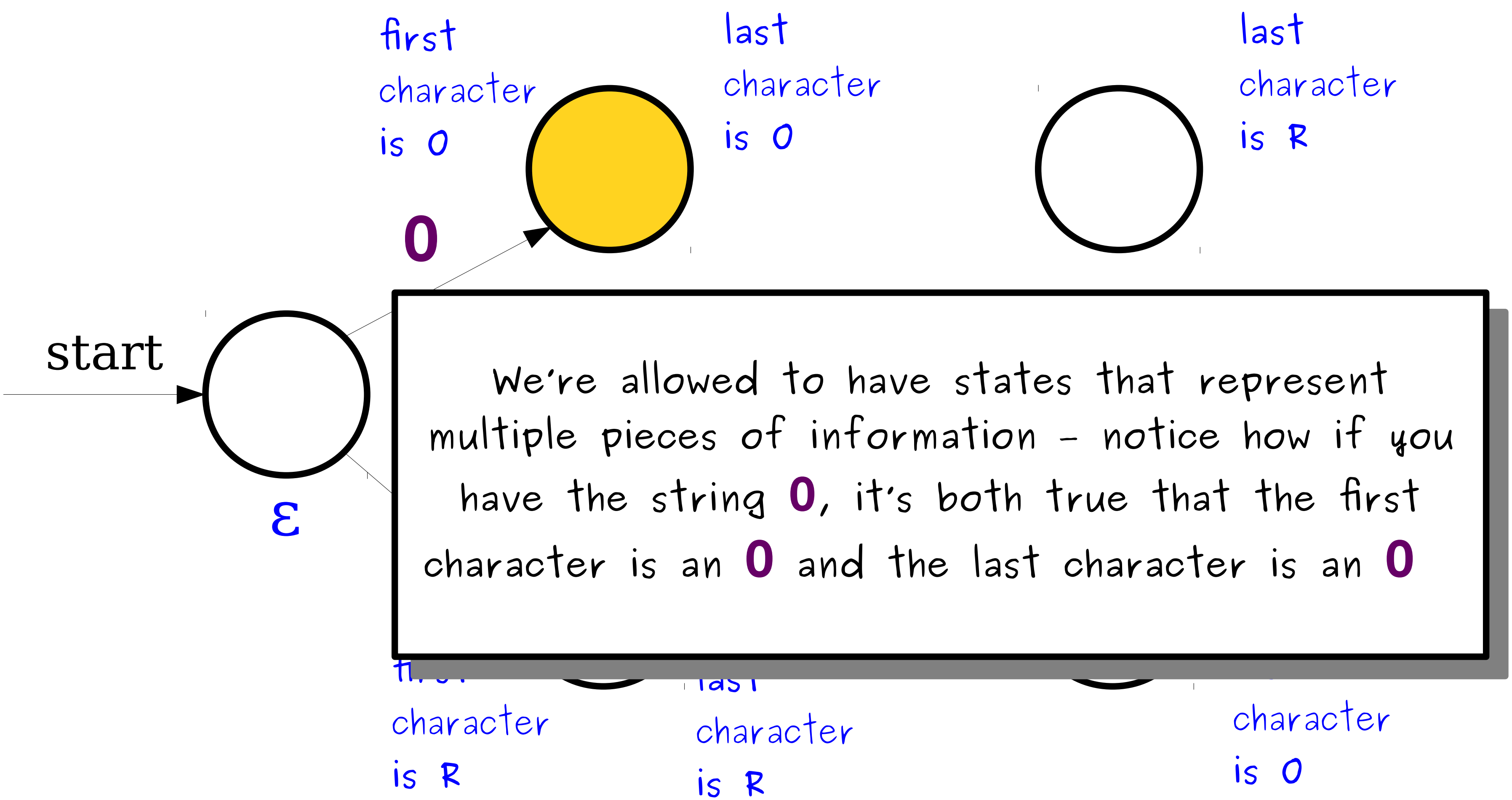
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

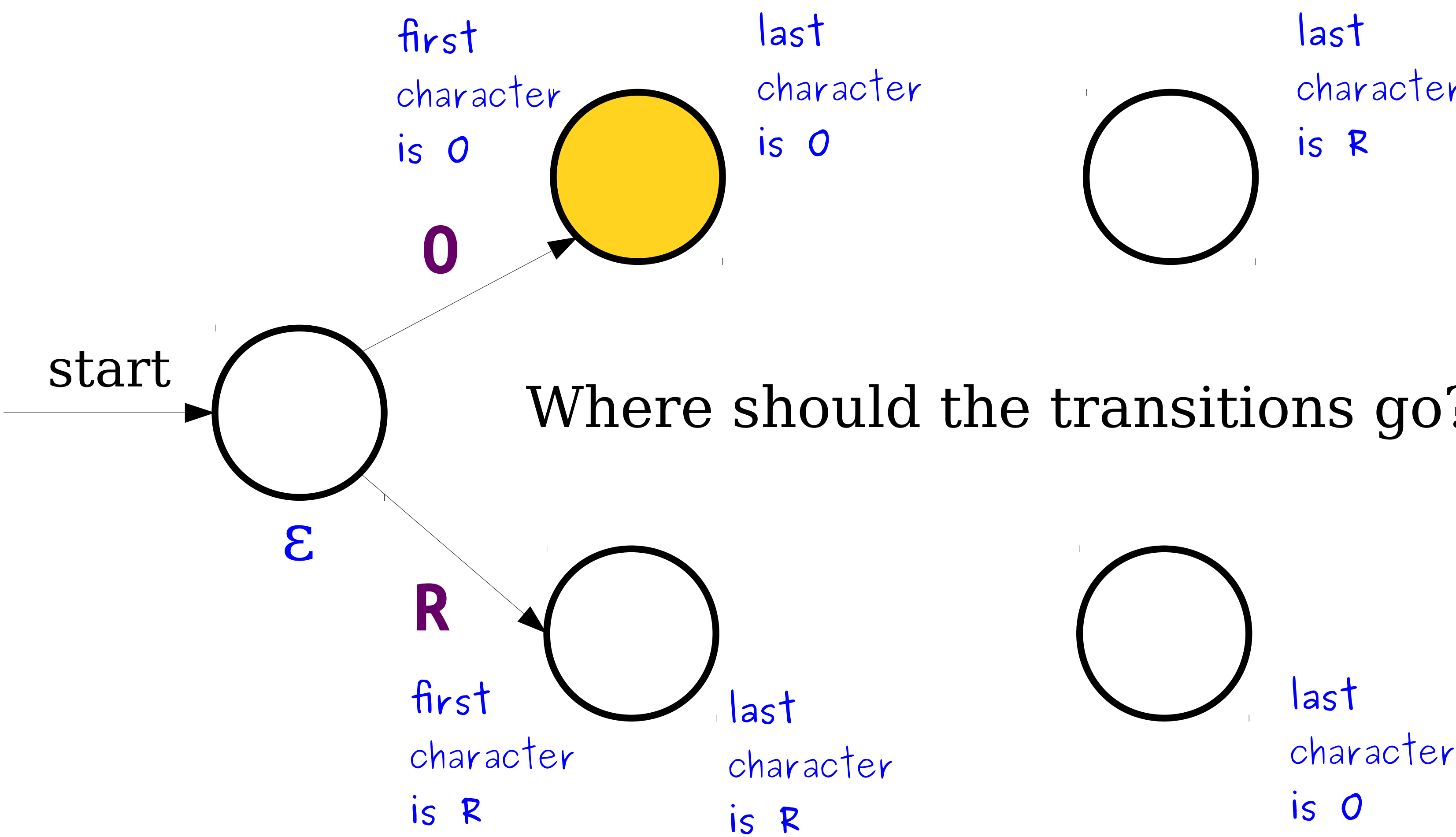
Oreo Sandwiches

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



Oreo Sandwiches

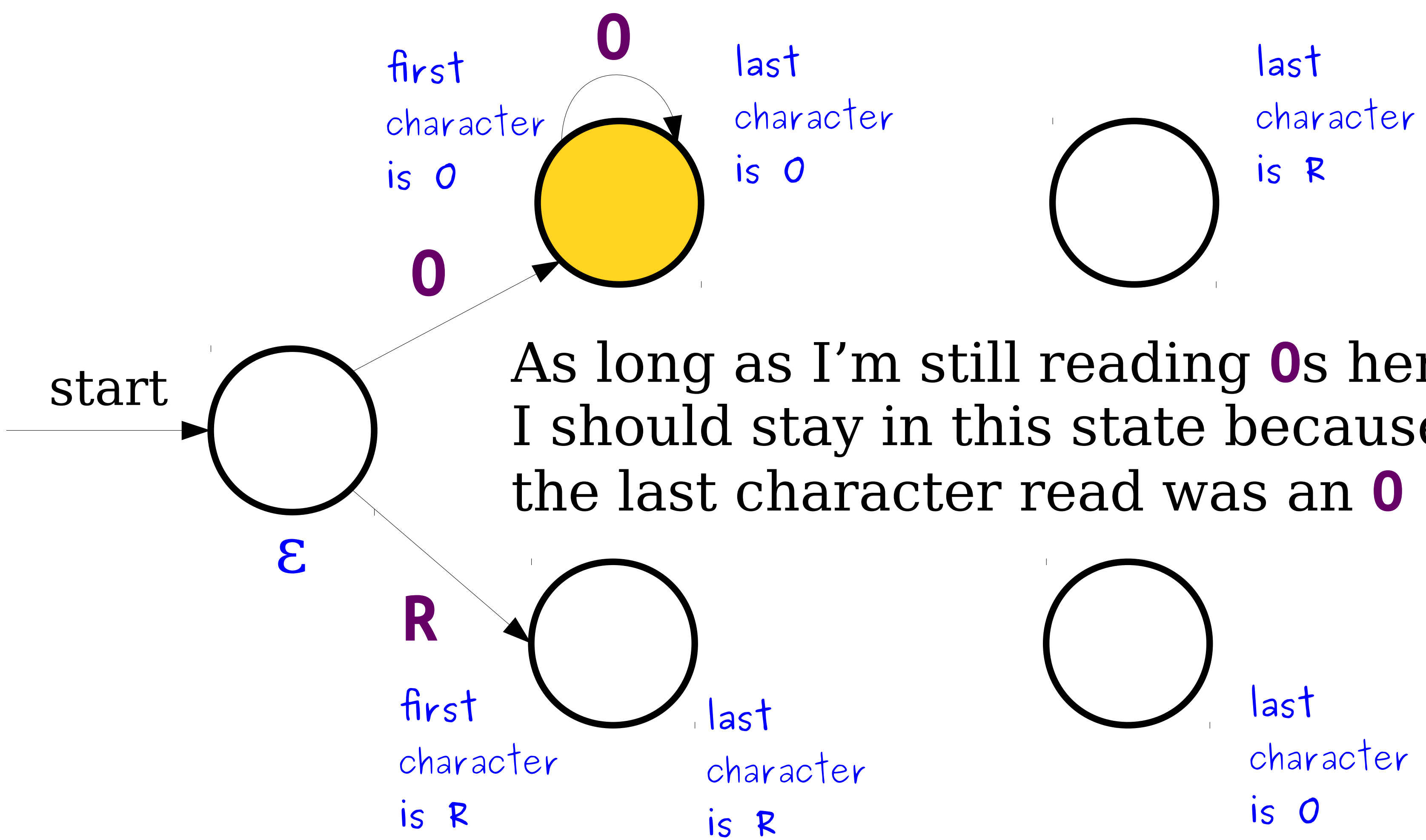
$L = \{\ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Where should the transitions go?

Oreo Sandwiches

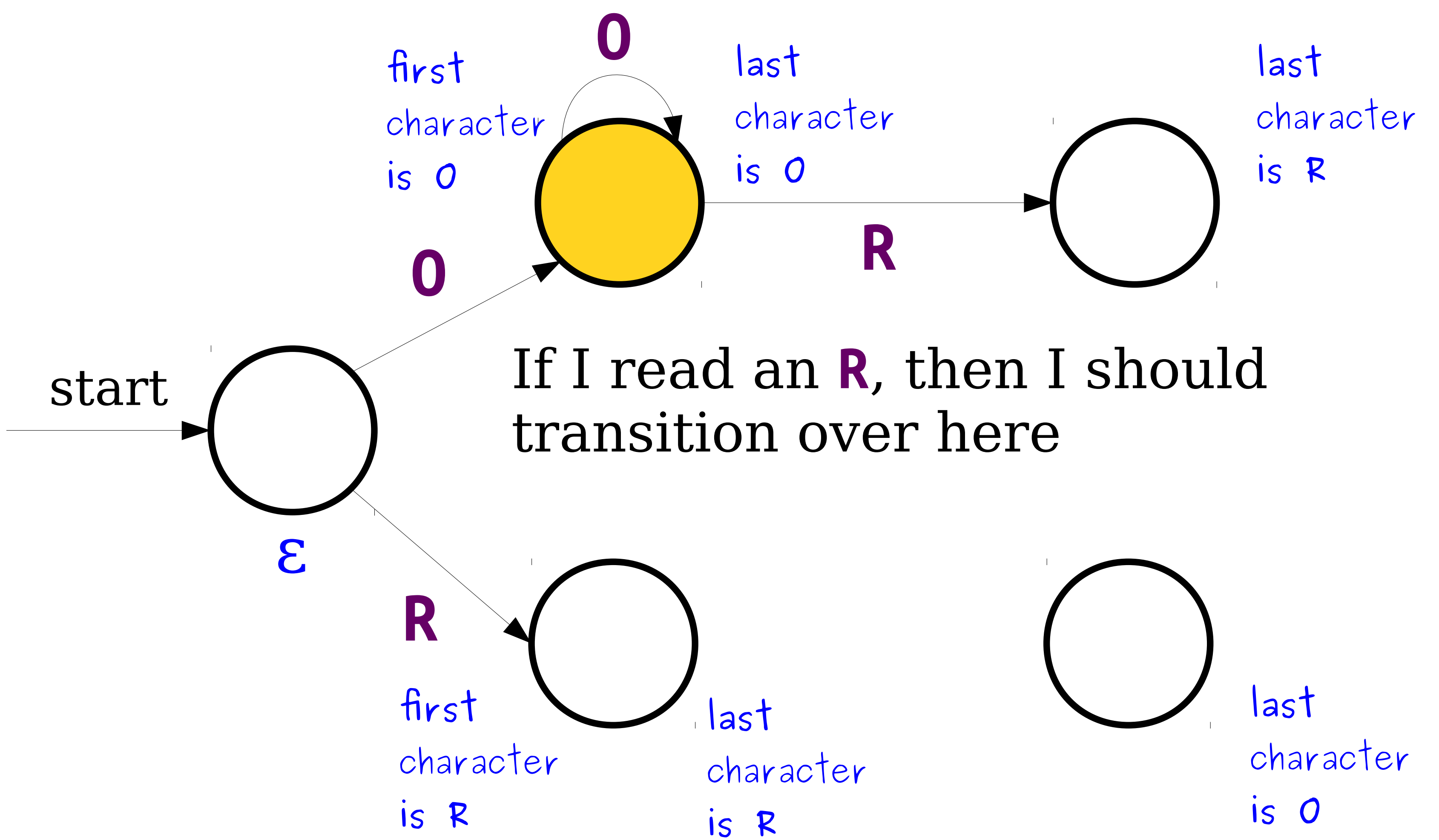
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

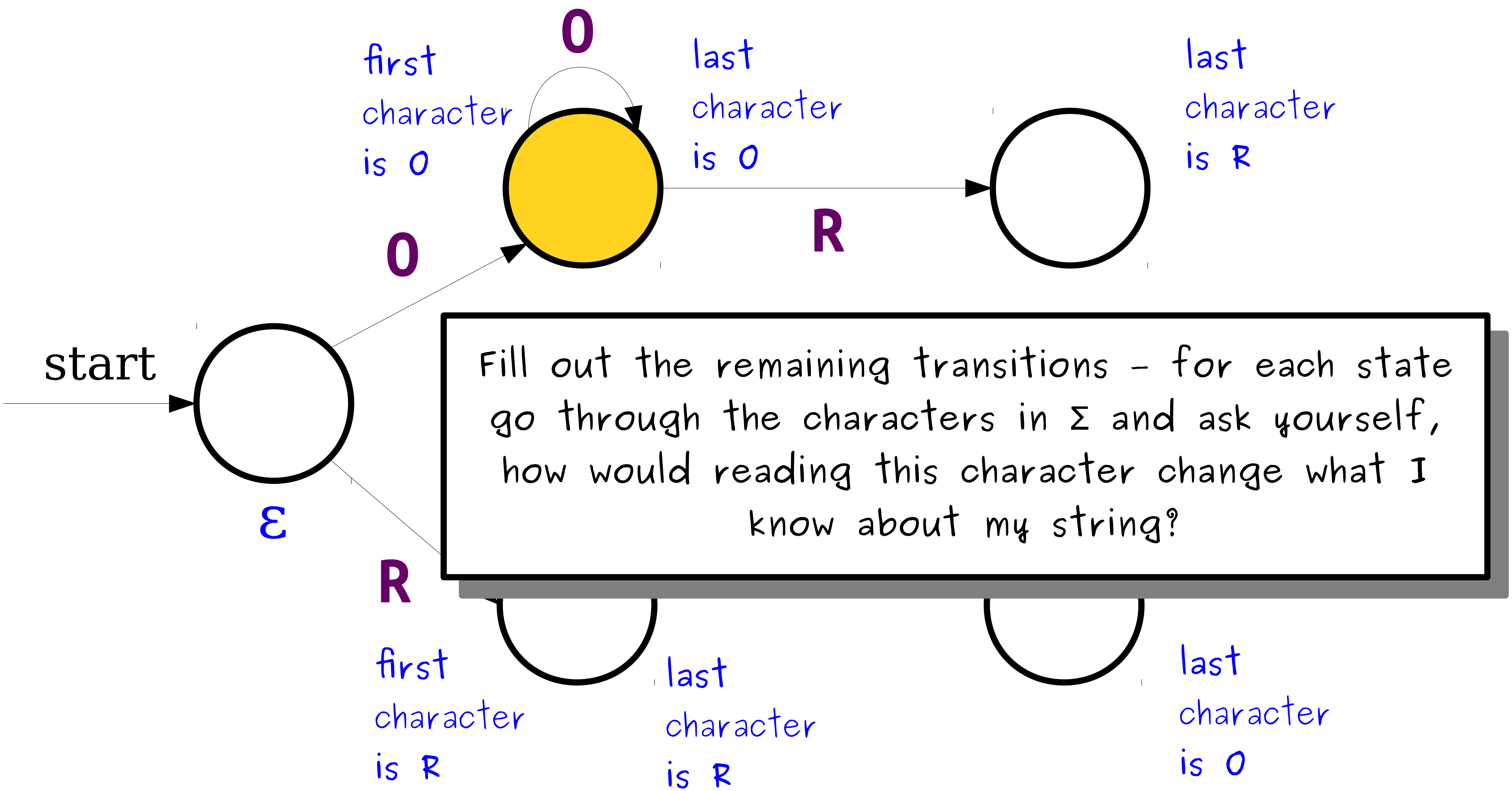
Oreo Sandwiches

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



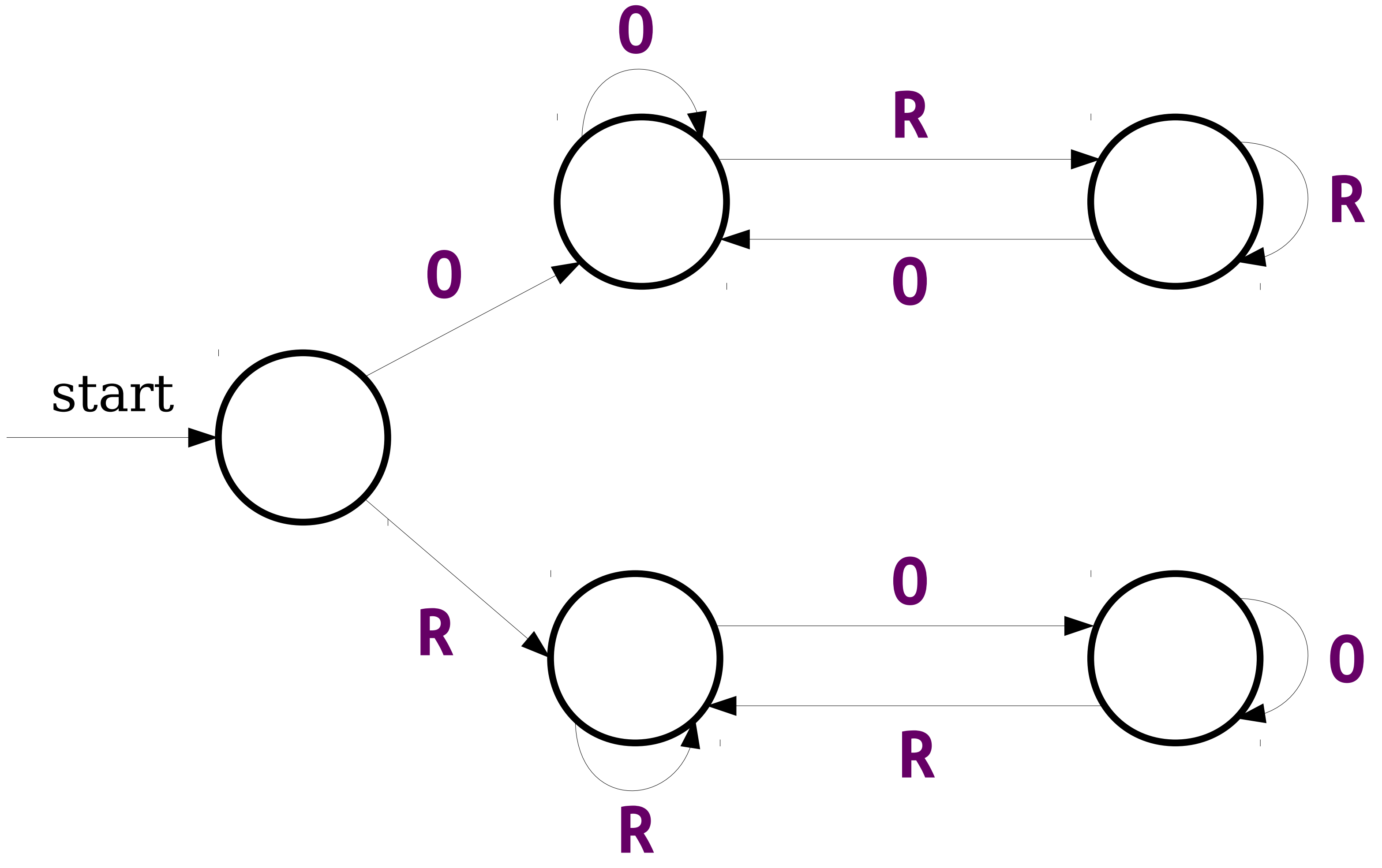
Oreo Sandwiches

L = { *w* ∈ Σ* | *w* ≠ ε and the first and last character of *w* are the same }



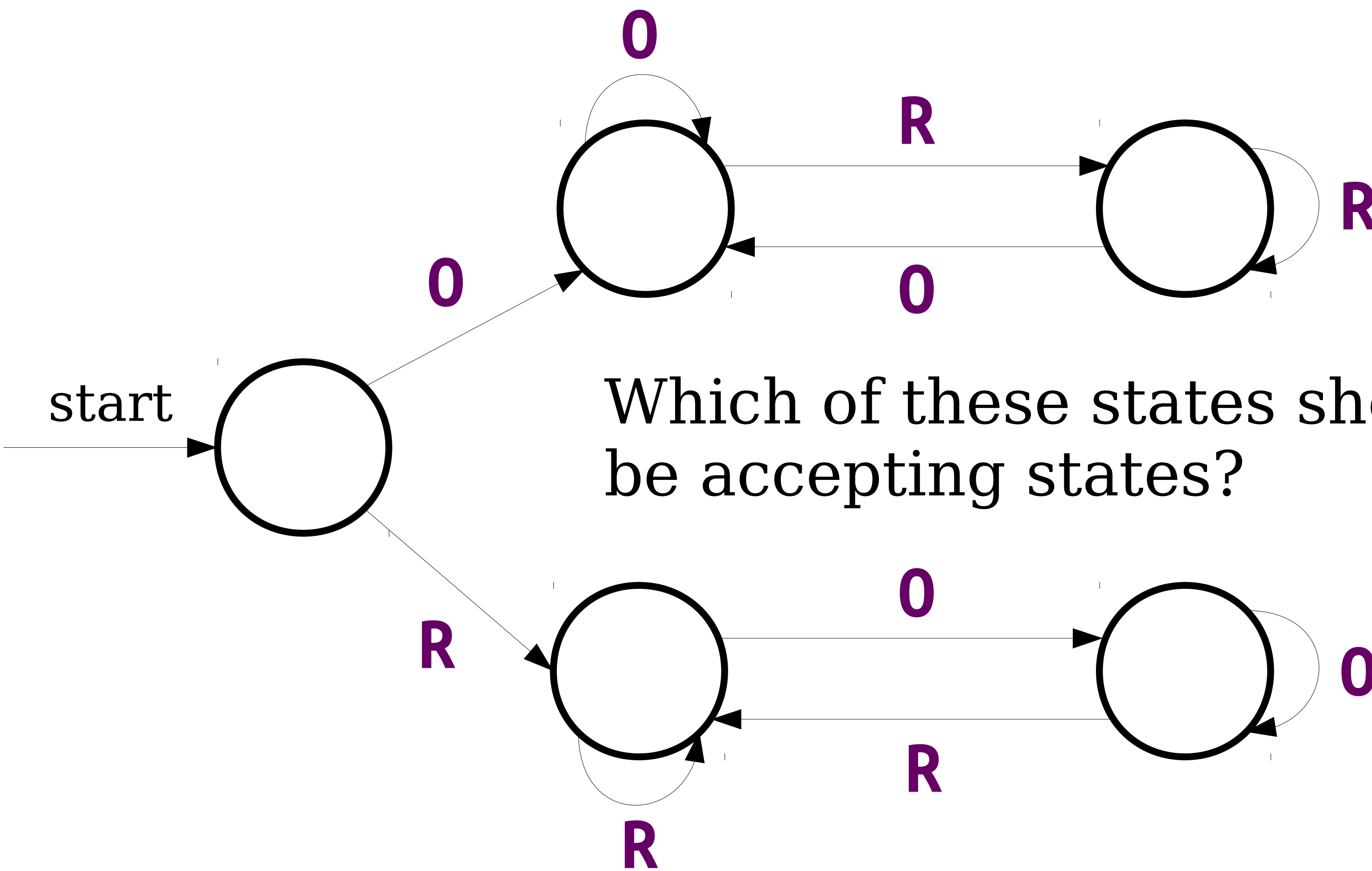
Oreo Sandwiches

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



Oreo Sandwiches

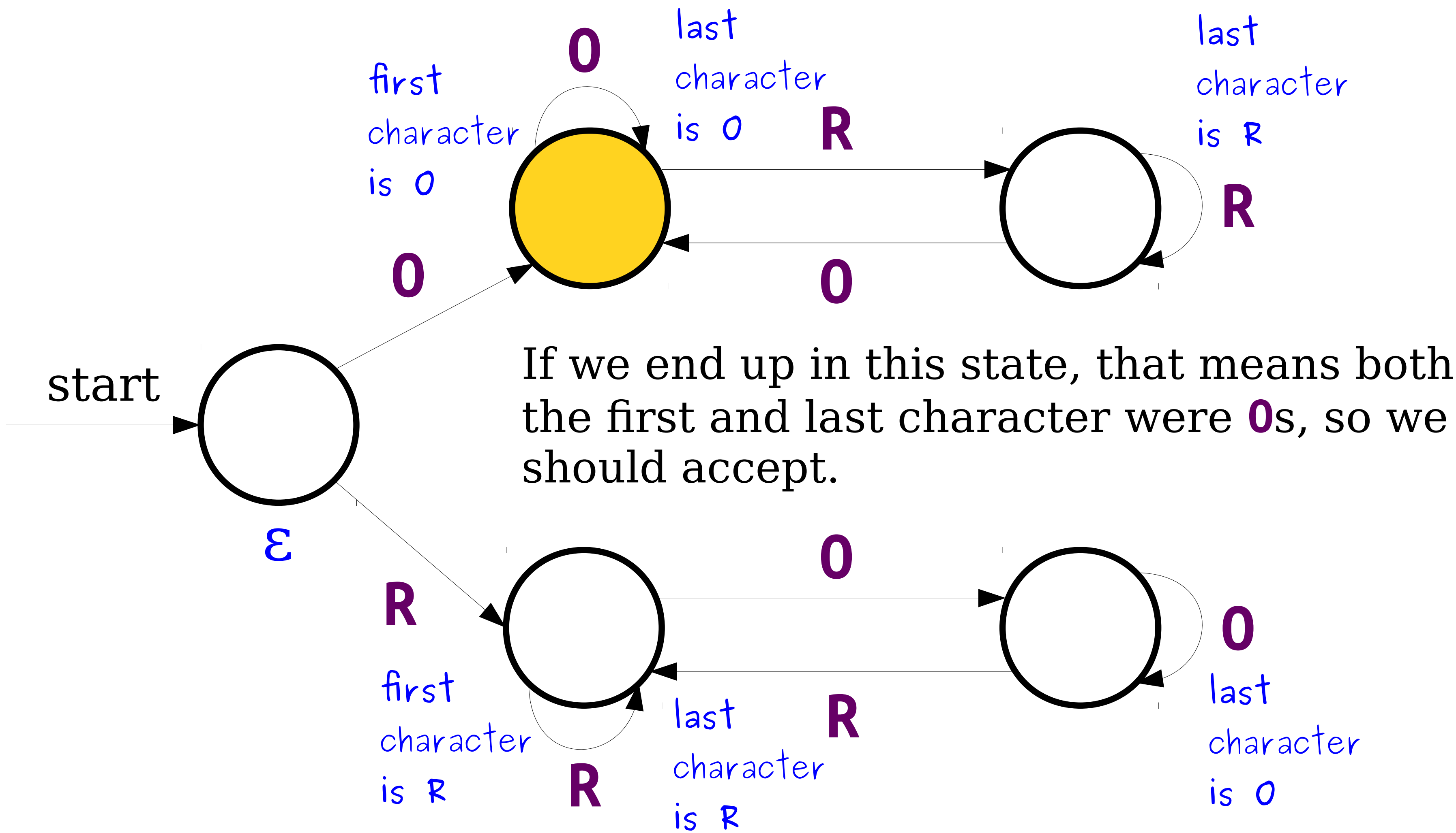
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



Which of these states should be accepting states?

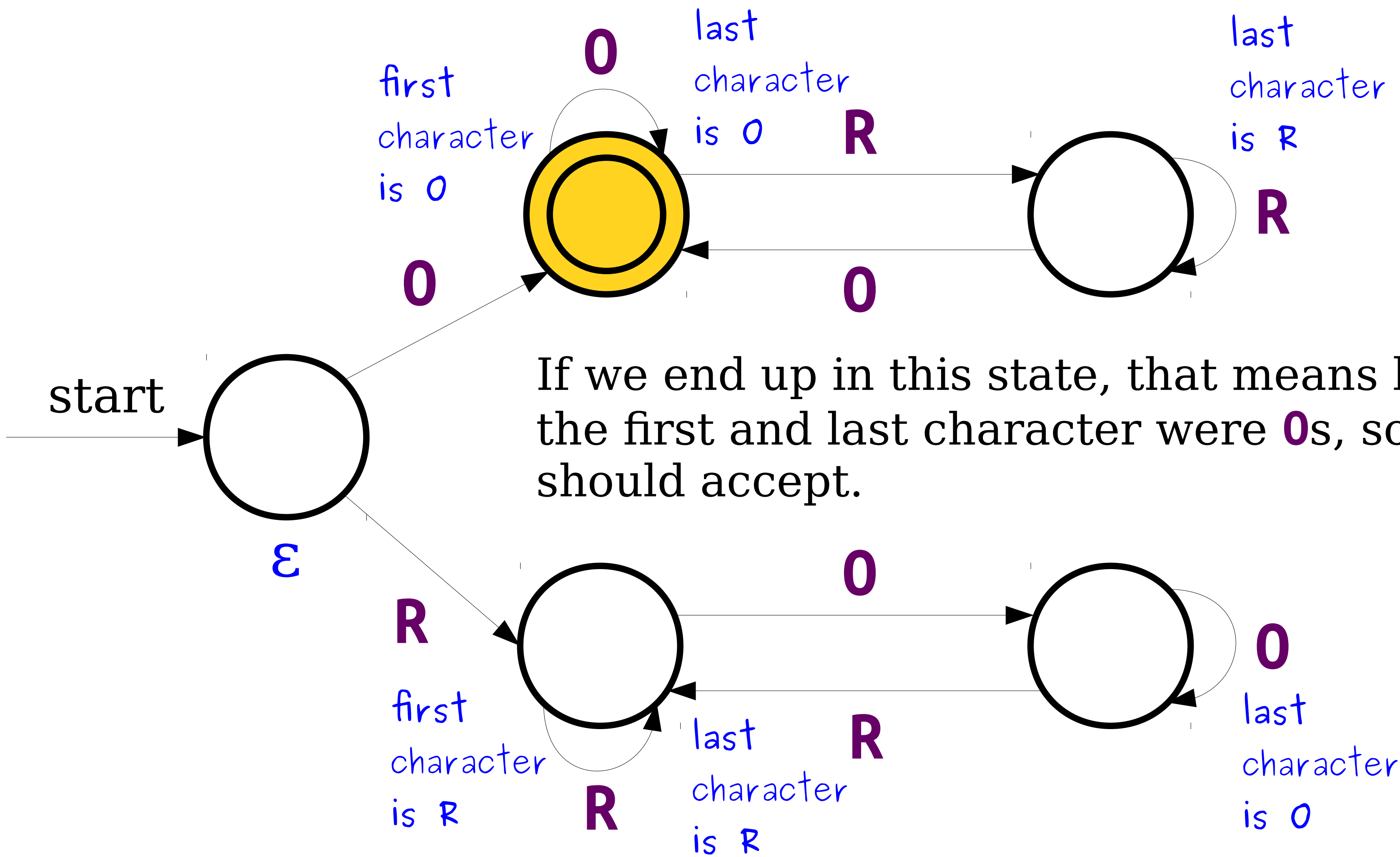
Oreo Sandwiches

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



Oreo Sandwiches

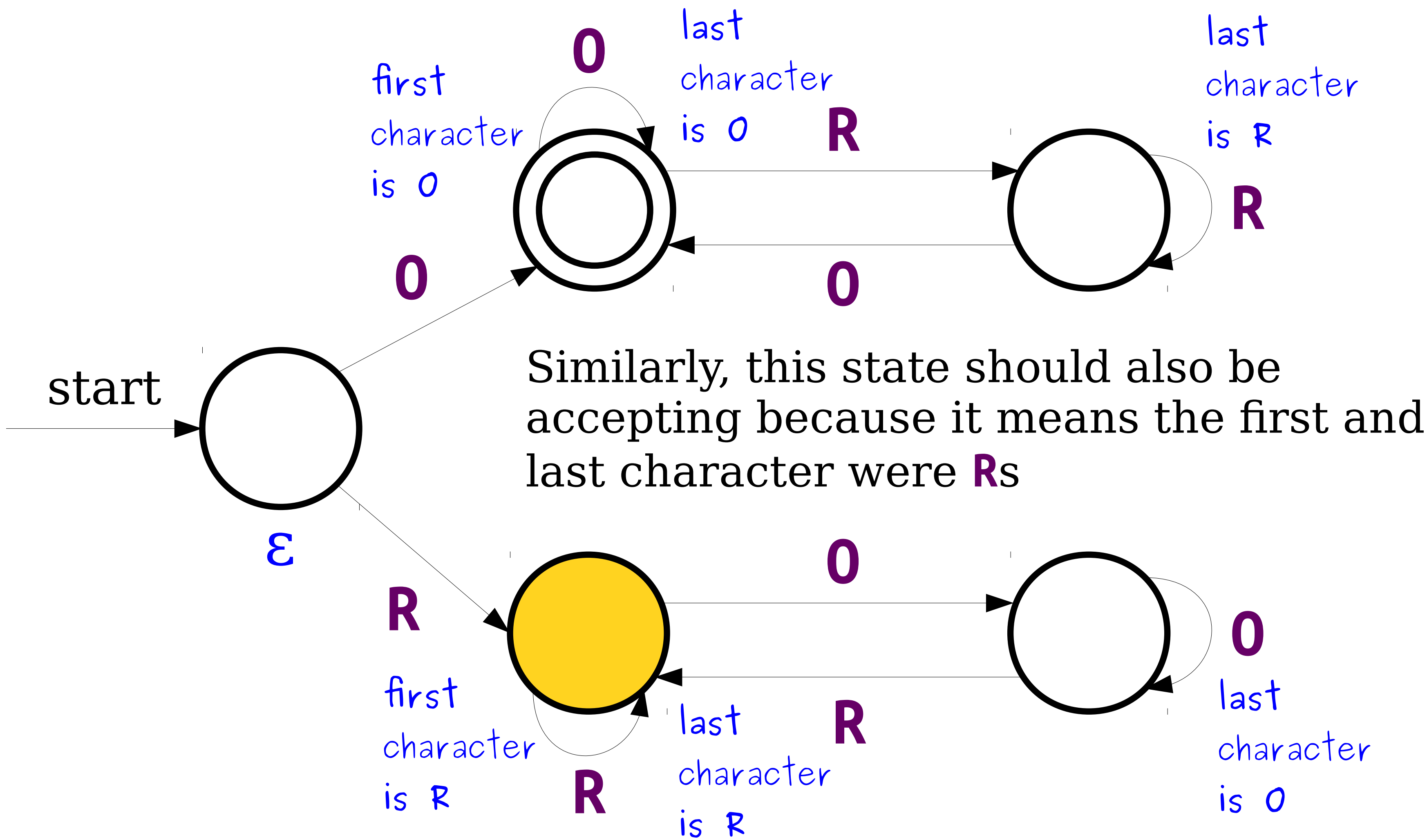
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

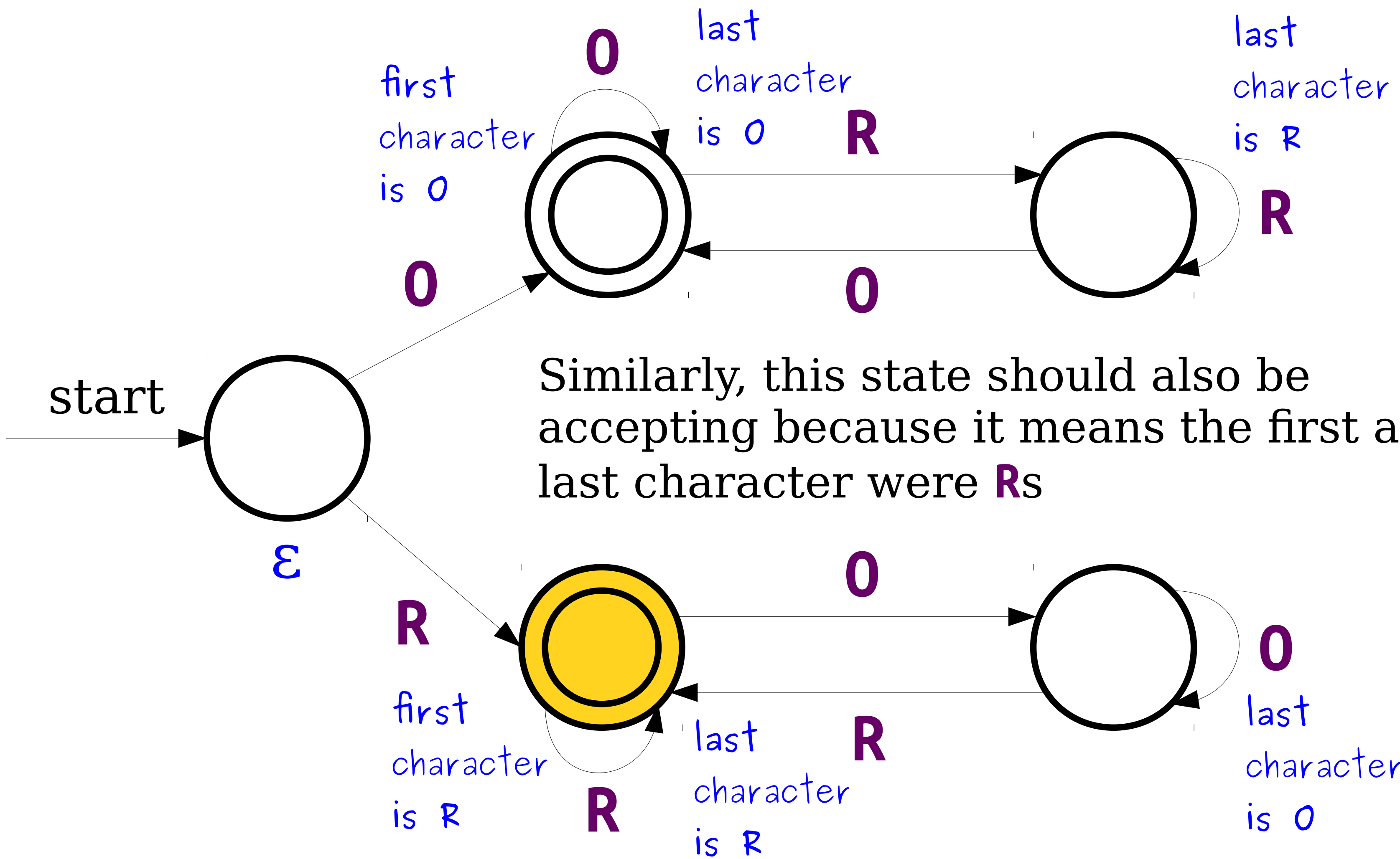
Oreo Sandwiches

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



Oreo Sandwiches

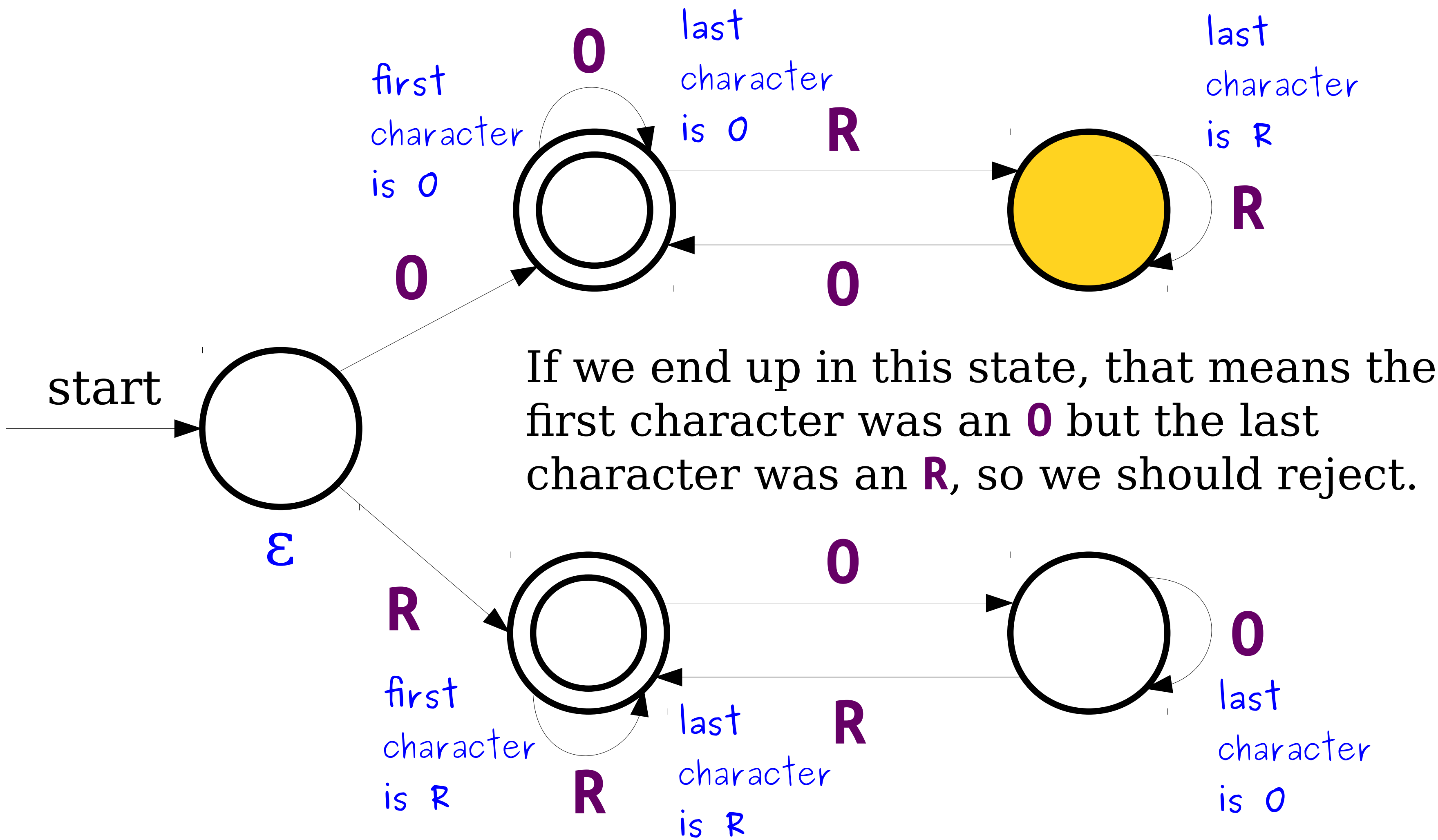
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

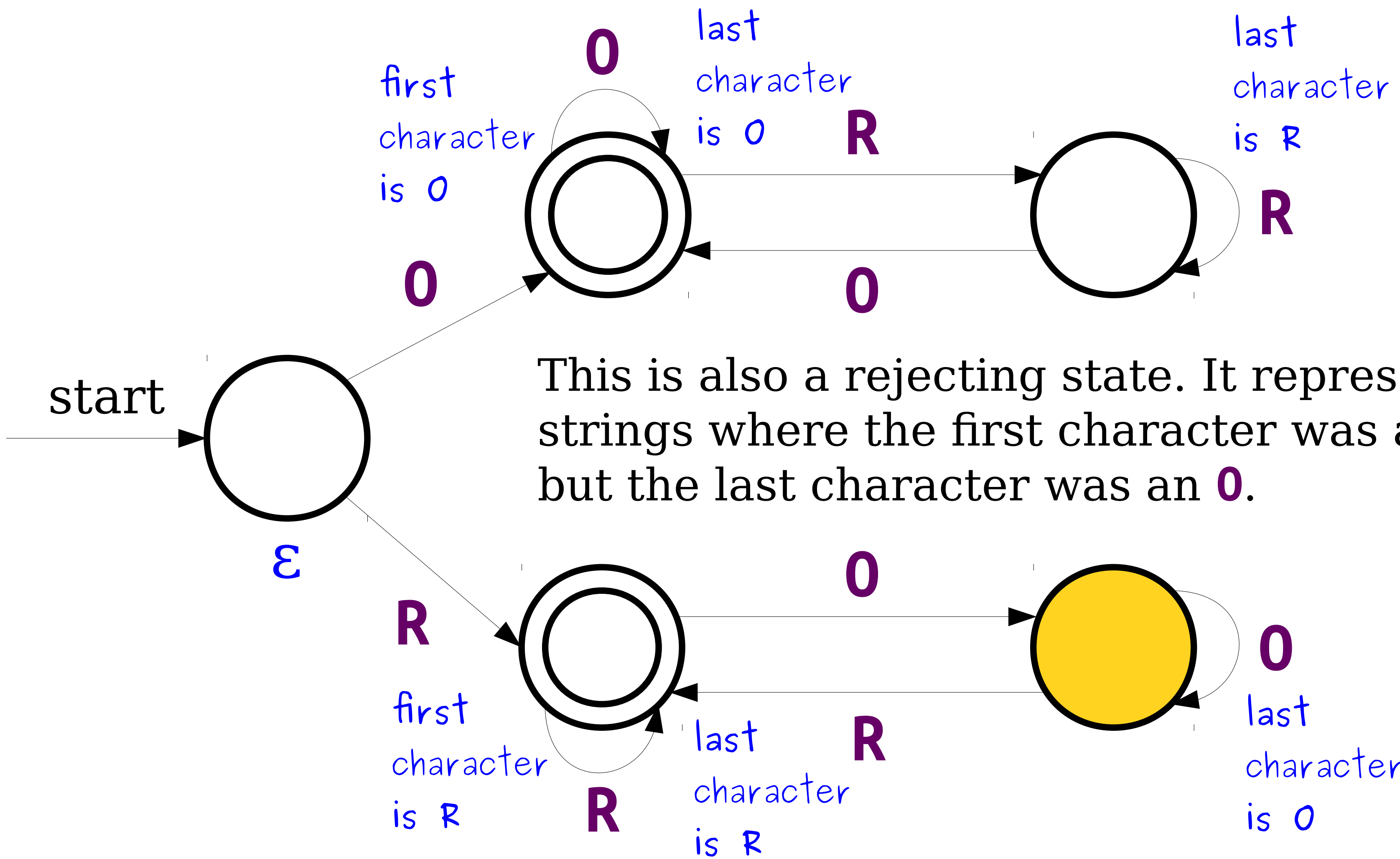
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

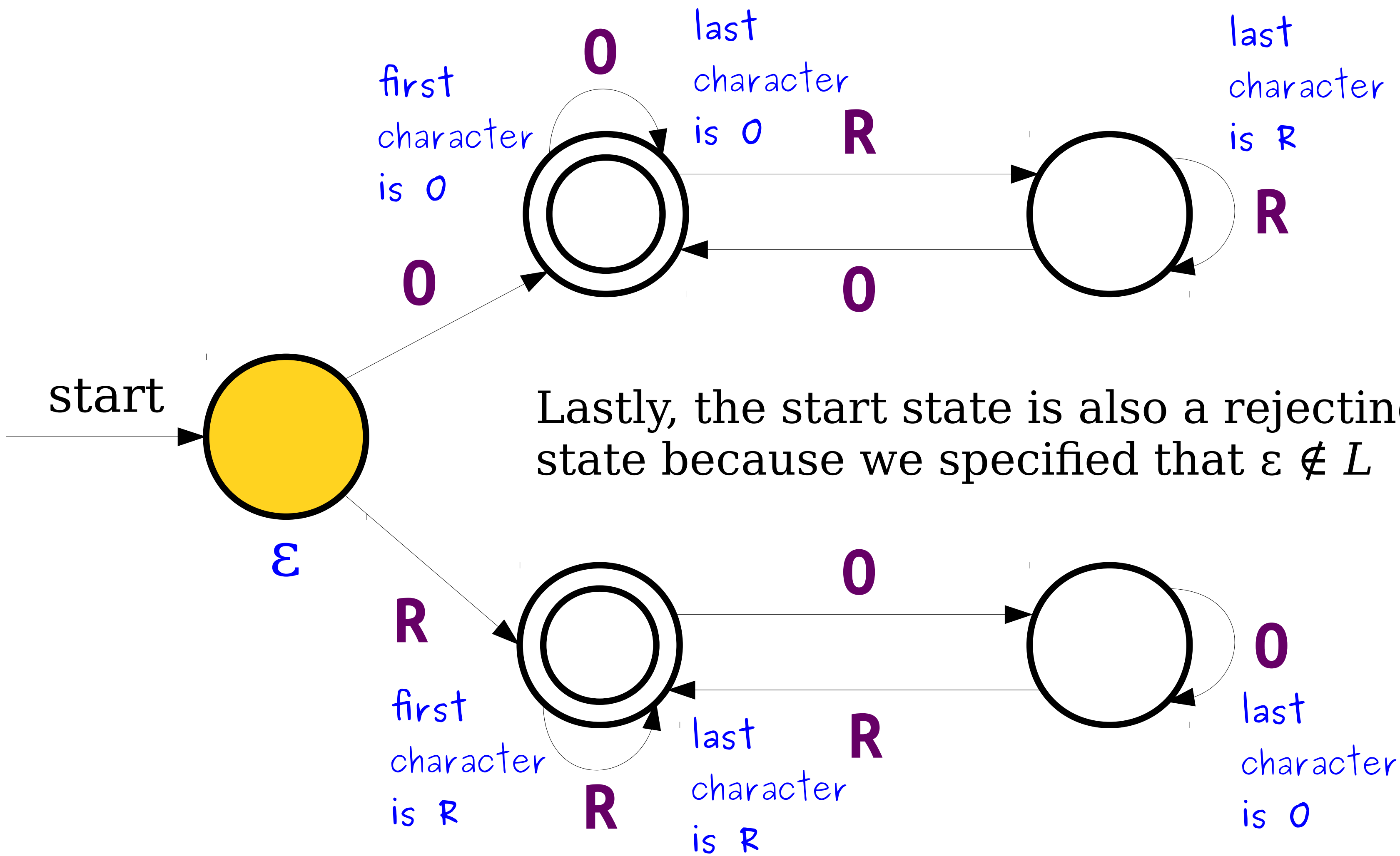
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

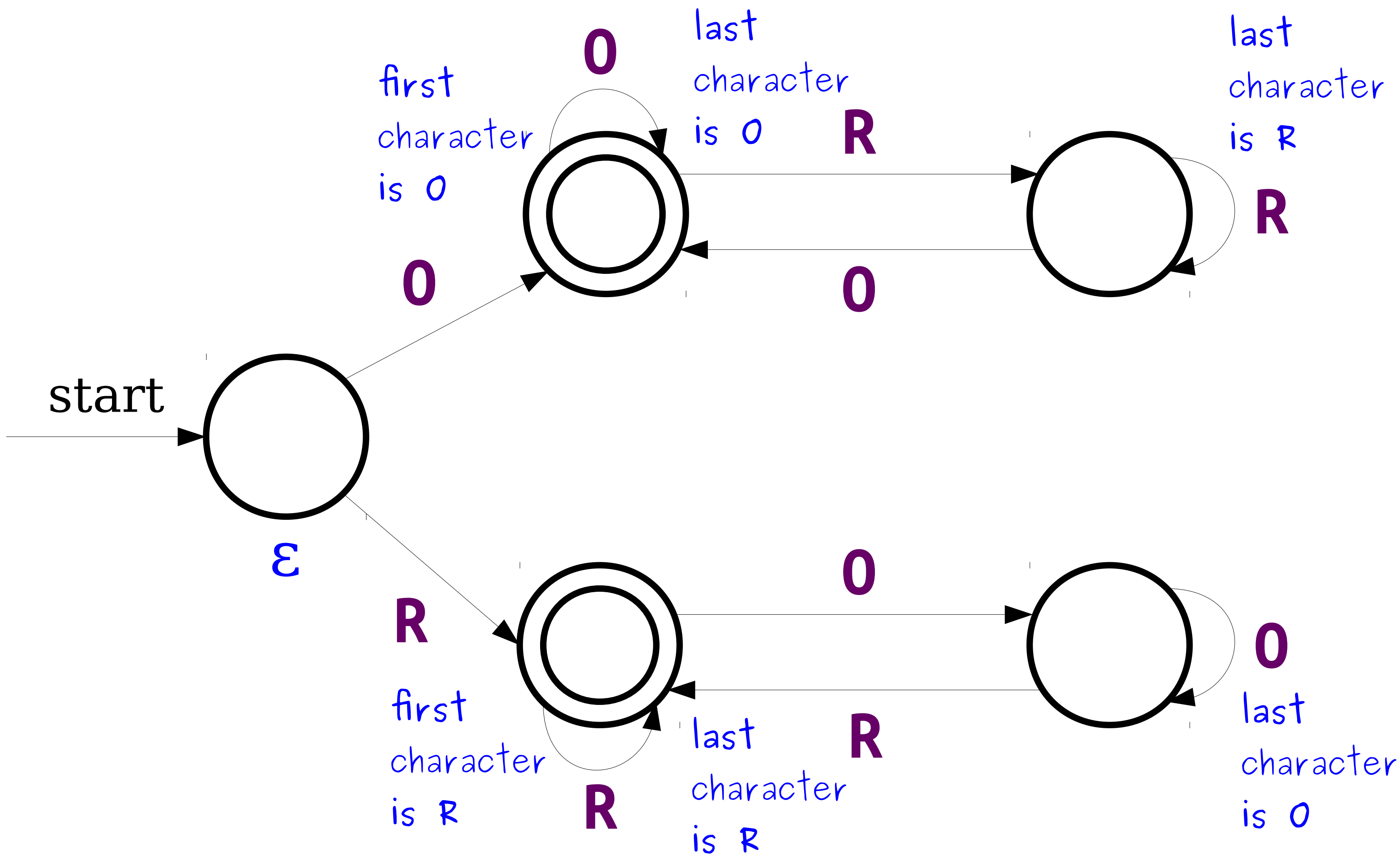
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

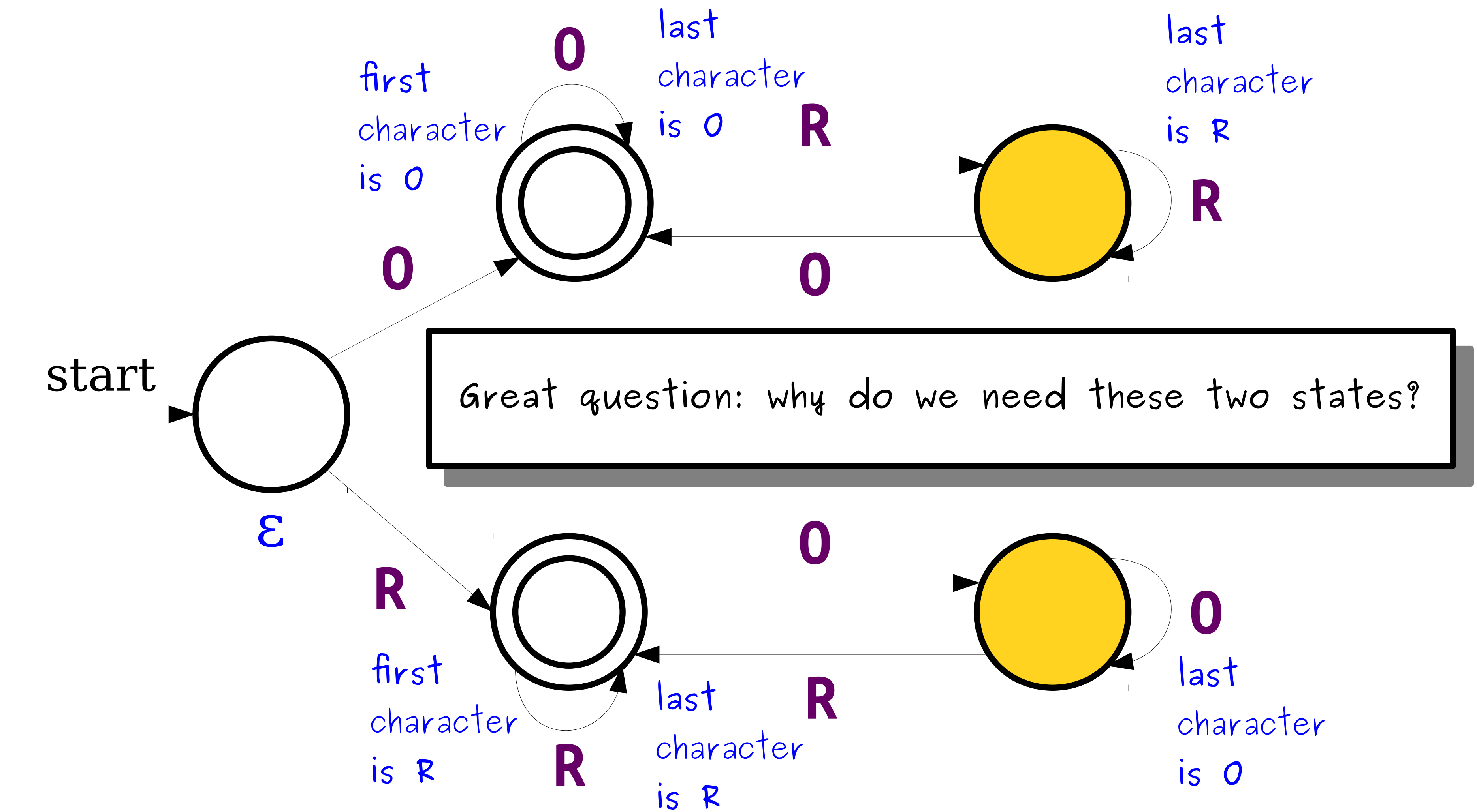
Oreo Sandwiches

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



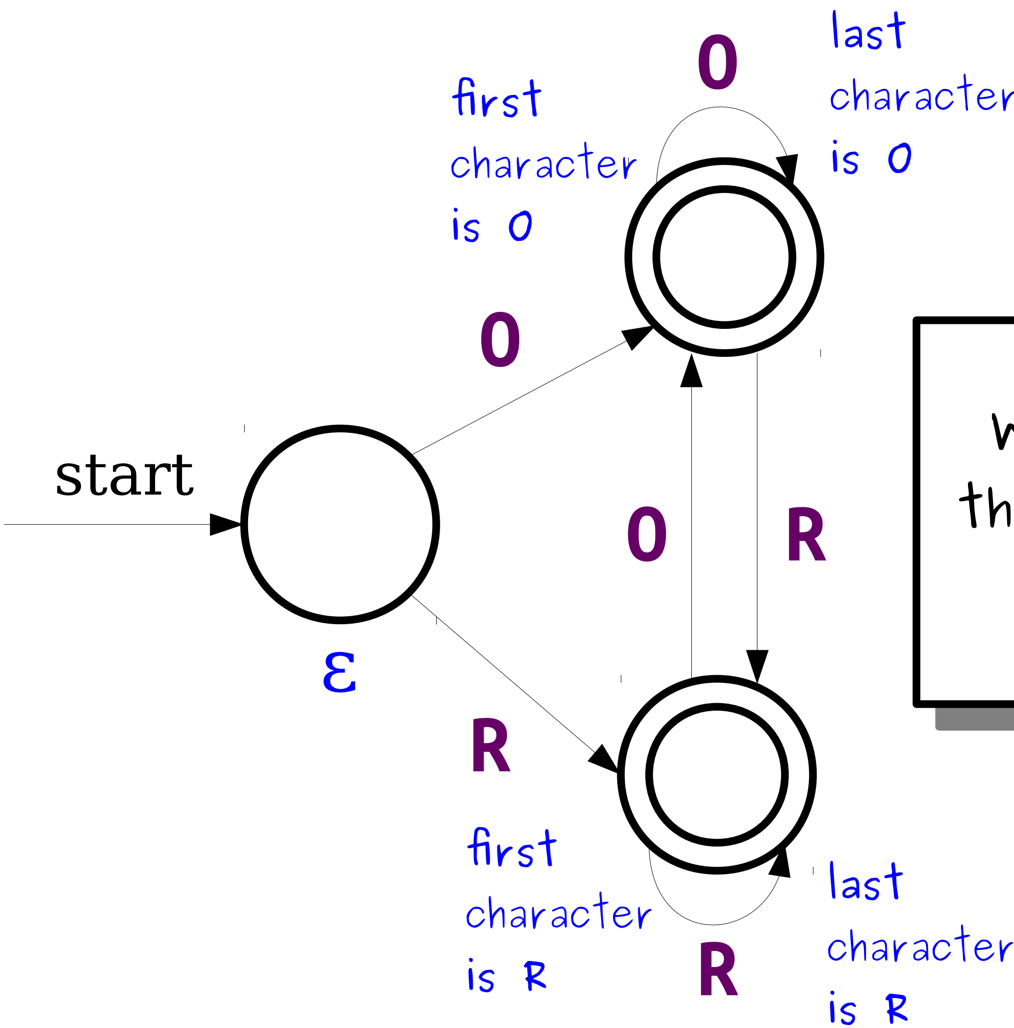
Oreo Sandwiches

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



Oreo Sandwiches

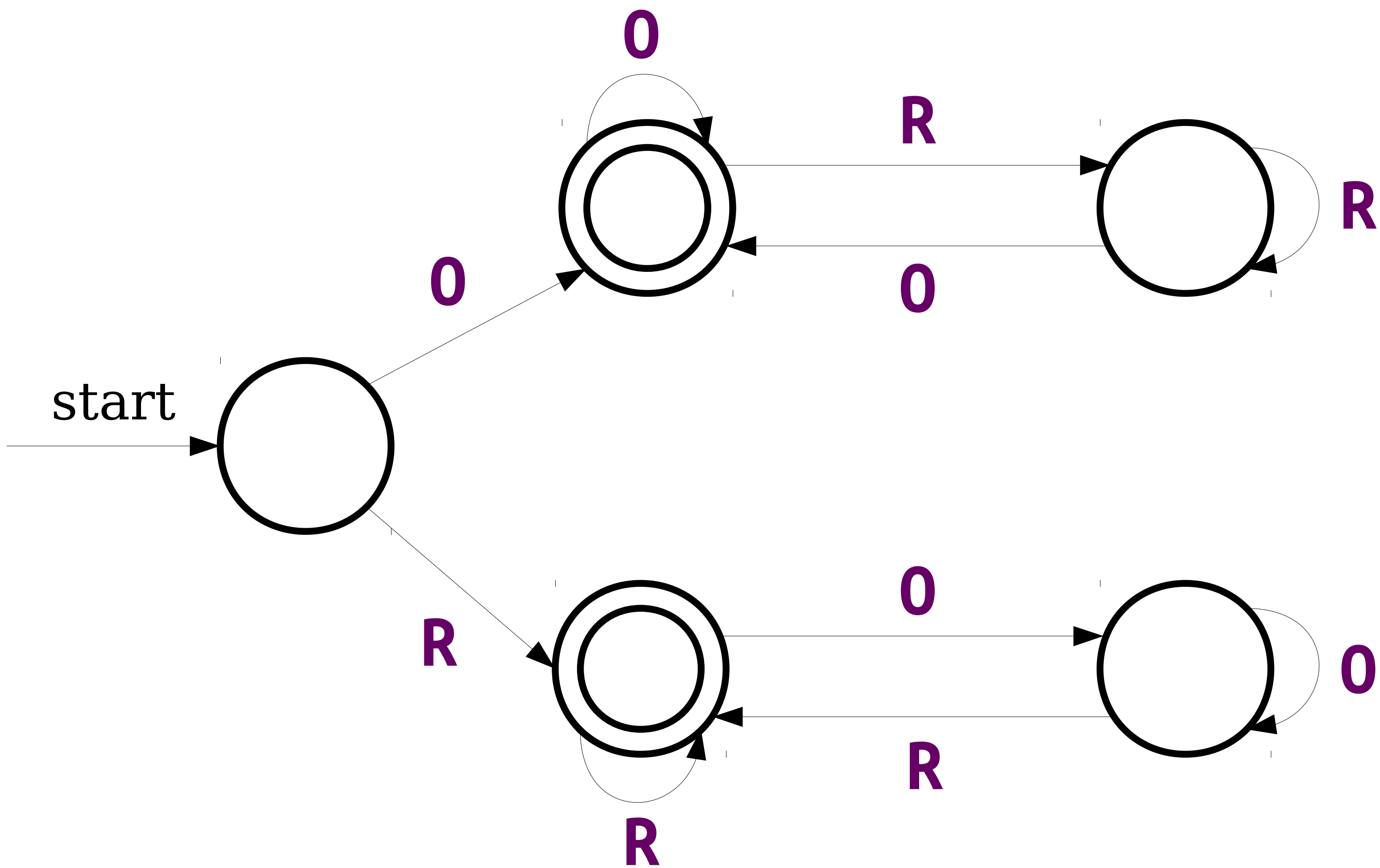
$L = \{ w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of w are the same $\}$



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

Oreo Sandwiches

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



Nonregular Languages

Approaching Myhill-Nerode

- The challenge in using the Myhill-Nerode theorem is finding the right set of strings.
- ***General intuition:***
  - Start by thinking about what information a computer "must" remember in order to answer correctly.
  - Choose a group of strings that all require different information.
  - Prove that those strings are distinguishable relative to the language in question.

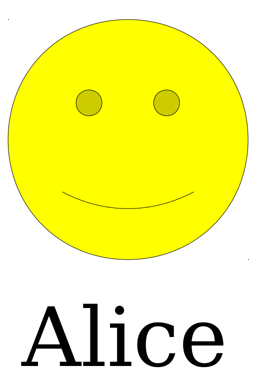
An Analogy

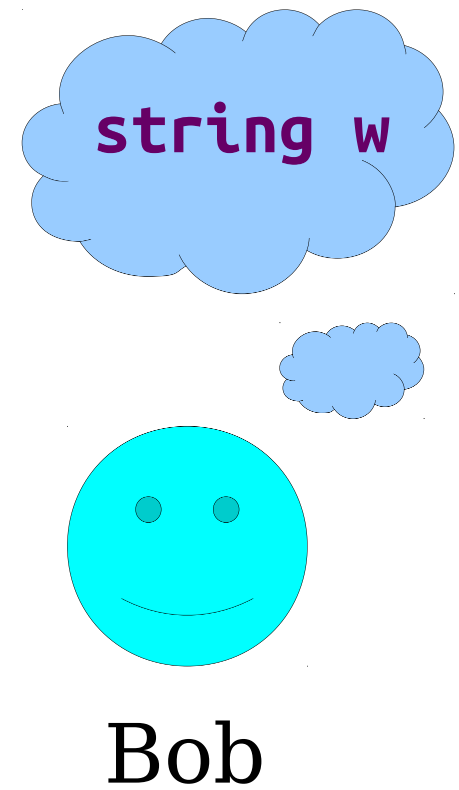
Imagine a scenario where Bob is thinking of a string and Alice has to figure out whether that string is in a particular language

An Analogy

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

language L





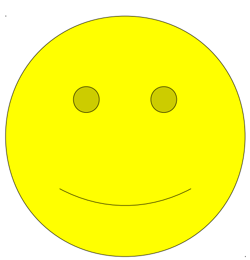
An Analogy

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

An Analogy

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

$L = \{ \ w \text{ is divisible by } 5 \ \}$



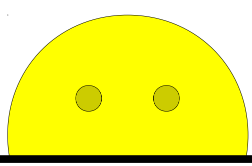
Alice

Bob

An Analogy

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

$L = \{ \ w \text{ is divisible by } 5 \ \}$

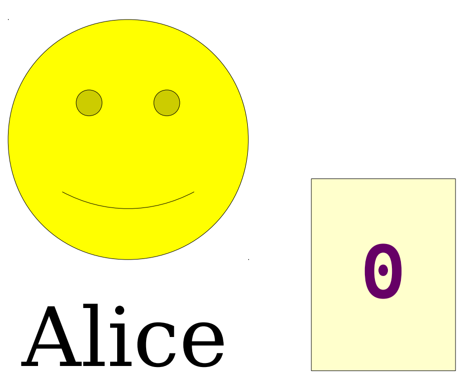


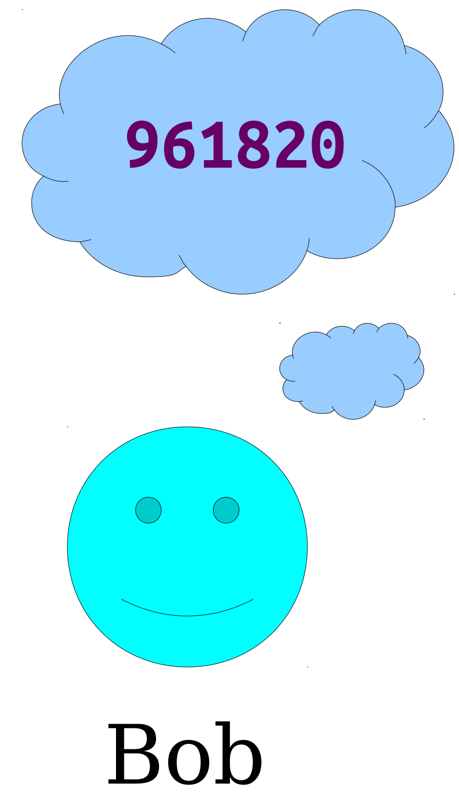
Initially it seems like Alice has to remember the whole number that Bob is sending to her, but we only care about divisibility by 5 here so we can get away with remembering a lot less.

An Analogy

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

L = { w is divisible by 5 }

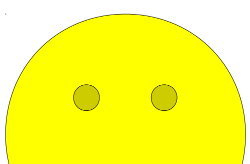




An Analogy

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

$L = \{\ w \text{ is divisible by } 5\ \}$

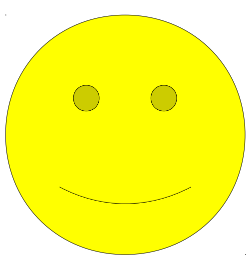


The number that Bob is thinking of could get unboundedly large, but the size of what Alice needs to remember remains constant (finite).

An Analogy

Let’s contrast this with one of the non-regular languages we saw in class:

$L = \{ \mathbf{a}^n \mathbf{b}^n \mid n \in \mathbb{N} \}$



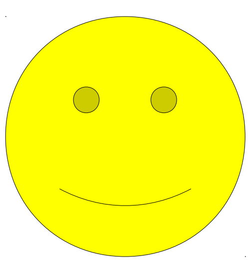
Alice

Bob

An Analogy

Alice needs to remember how many **a**'s she's seen so far, since she needs to verify that the number of **b**'s matches

$$L = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$$



Alice

Bob

An Analogy

Alice needs to remember how many **a**'s she's seen so far, since she needs to verify that the number of **b**'s matches

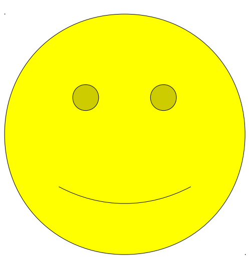
$$L = \{ \mathbf{a}^n\mathbf{b}^n \mid n \in \mathbb{N} \}$$

As the size of Bob's string gets larger, the amount of memory Alice needs also increases. Since Bob's string could get unboundedly large, we need infinite memory.

An Analogy

Key insight: if Alice has to remember ***infinitely*** many things, or one of ***infinitely*** many possibilities, the language is probably not regular

language L



Alice

Bob

Context-Free Grammars

Storing Information in Nonterminals

- ***Key idea:*** Different non-terminals should represent different states or different types of strings.
  - For example, different phases of the build, or different possible structures for the string.
  - Think like the same ideas from DFA/NFA design where states in your automata represent pieces of information.

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Examples:

ε $\in L$	**a** $\notin L$
abb $\in L$	**b** $\notin L$
bab $\in L$	**ababab** $\notin L$
aababa $\in L$	**aabaaaaaa** $\notin L$
bbbbbb $\in L$	**bbbb** $\notin L$

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Examples:

ε $\in L$	**a** $\notin L$
a ⁞ **bb** $\in L$	**b** $\notin L$
b ⁞ **ab** $\in L$	**ab** ⁞ **abab** $\notin L$
aa ⁞ **baba** $\in L$	**aab** ⁞ **aaaaaa** $\notin L$
bb ⁞ **bbbb** $\in L$	**bbbb** $\notin L$

Storing Information in Nonterminals

- Let Σ = {**a**, **b**} and let $L = \{w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same }.
- One approach:

aaa	**bab**
abb	**bbb**
aaabab	**bbabbb**
aababa	**bbbaaaaaa**
aaaaaaaaa	**bbbbbabaa**

Observation 1:

Strings in this language are either: the first third is **a**s or the first third is **b**s.

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Observation 2:

Amongst these strings, for every **a** I have in the first third, I need two other characters in the last two thirds.

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa bab

abb bbb

aaabab bbabbb

aaaaa

babaa

Observation 2:

Amongst these strings, for every **a** I have in the first third, I need two other characters in the last two thirds.

This pattern of "for every x I see here, I need a y somewhere else in the string" is very common in CFGs!

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Observation 2:

Amongst these strings, for every **a** I have in the first third, I need two other characters in the last two thirds.

A → **a** **AXX** | **ε**   **X** → **a** | **b**

Storing Information in Nonterminals

- Let Σ = {**a**, **b**} and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa **bab**

abb

aaabab

aababa

aaaaaaaaa

Here the nonterminal **A** represents "a string where the first third is **a**'s" and the nonterminal **X** represents "any character"

A → **aAXX** | **ε** **X** → **a** | **b**

Storing Information in Nonterminals

- Let $\Sigma = \{$ **a**, **b** $\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

A → a**AXX** | ε   **X** → a | b

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

B → **bBXX** | **ε**   **X** → **a** | **b**

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Tying everything together:

  S → **A** | **B**

  A → **aAXX** | **ε**

  B → **bBXX** | **ε**

  X → **a** | **b**

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Tying everything together:

S → **A** | **B**

A → a**AXX** | ε

B → b**BXX** | ε

X → a | b

Overall strings in this language either follow the pattern of **A** or **B**.

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L = \{w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Tying everything together:

S → **A** | **B**

A → a**AXX** | ε

B → b**BXX** | ε

X → a | b

A represents "strings where the first third is a's"

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Tying everything together:

S → **A** | **B**

A → **aAXX** | **ε**

B → **bBXX** | **ε**

X → **a** | **b**

B represents "strings where the first third is **b**'s"

Storing Information in Nonterminals

- Let $\Sigma = \{$a, b$\}$ and let $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all the characters in the first third of w are the same $\}$.
- Tying everything together:

S → **A** | **B**

A → a**AXX** | ε

B → b**BXX** | ε

X → a | b

X represents "either an **a** or a **b**"