# Turing Machines

## Part One

What problems can we solve with a computer?

# The Problem

- Finite automata accept precisely the regular languages
- We may need unbounded memory to recognize context-free languages described by CFGs (e.g. $\{ \mathbf{a}^n\mathbf{b}^n \mid n \in \mathbb{N} \}$)
- But CFGs aren't a means of deciding whether or not to accept a given string—they only describe how to generate them
- How do we build an **automaton** with finitely many states but unbounded memory?

# A Brief History Lesson

# Fin de siècle optimism

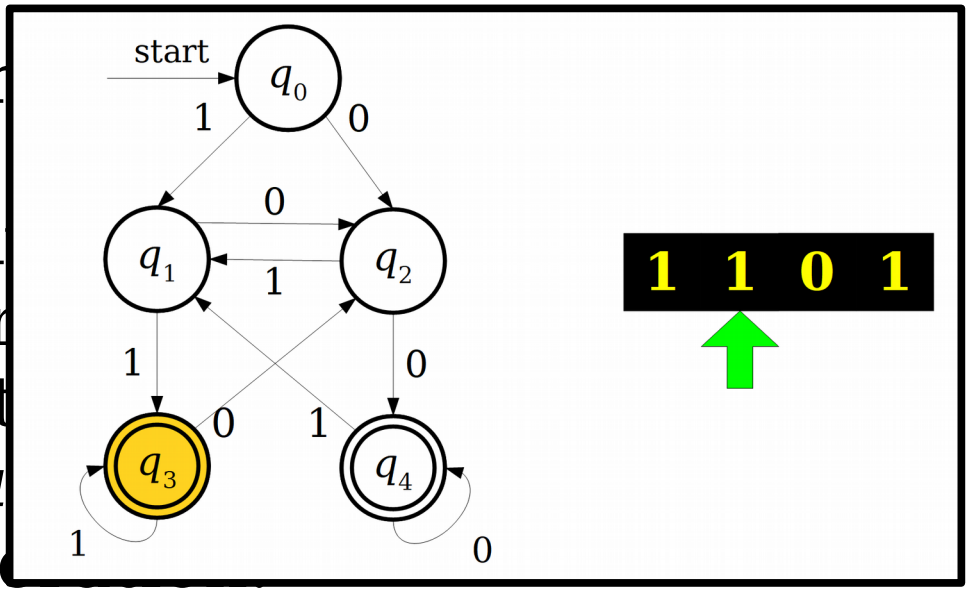
# Technology will solve all of mankind's problems! No more wars or sad, ever!




# Hilbert’s Vision

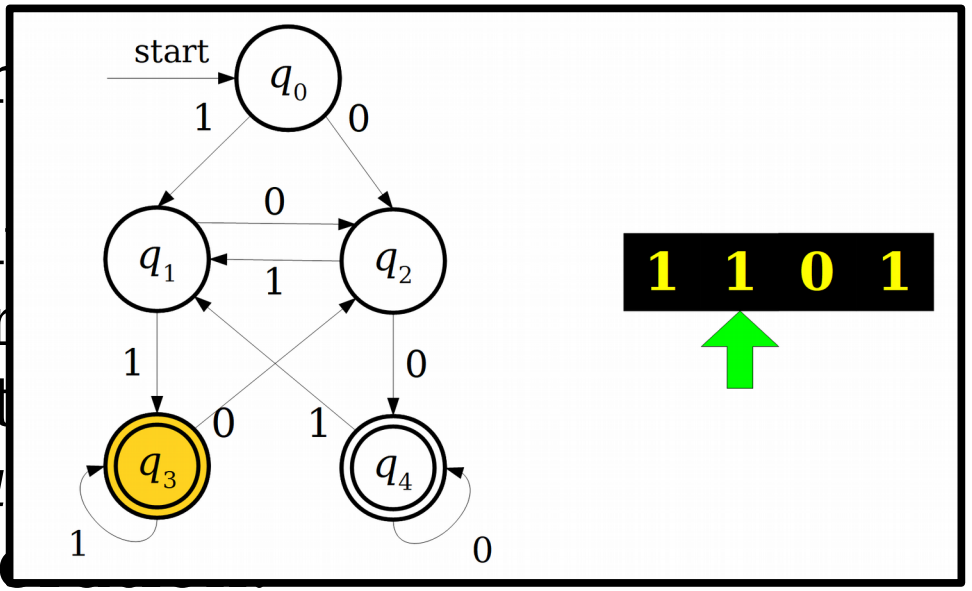
• 1900: International Congress of Mathematicians meeting in Paris
• Hilbert proposes 23 unsolved problems as the agenda for the coming years
• An important theme is not simply proving more theorems, but achieving *automation* of theorem-proving, even theorem generation.
• Humanity lives in leisure while all Truth flows effortlessly into our hands on a ticker tape!

# Hilbert's Vision




• 1900: International Congr
meeting in Paris
• Hilbert proposes 23 unsol
agenda for the coming year
• An important theme is not
theorems, but achieving *au*
proving, even theorem gen
• Humanity lives in leisure while all Truth flows effortlessly into our hands on a ticker tape!

# Hilbert's Vision




• 1900: International Congr
meeting in Paris
• Hilbert proposes 23 unsol
agenda for the coming year
• An important theme is not
theorems, but achieving *au*
proving, even theorem gen
• Humanity lives in leisure while all Truth flows effortlessly into our hands on a ticker tape!

"No one shall expel us from the Paradise that Cantor has created!" -David Hilbert

1900 Hilbert: optimism!
Reality: womp womp

# 1900 Hilbert: optimism! Reality: womp womp

- Hilbert’s agenda is both a spectacular success and a spectacular failure

# 1900 Hilbert: optimism!
# Reality: womp womp

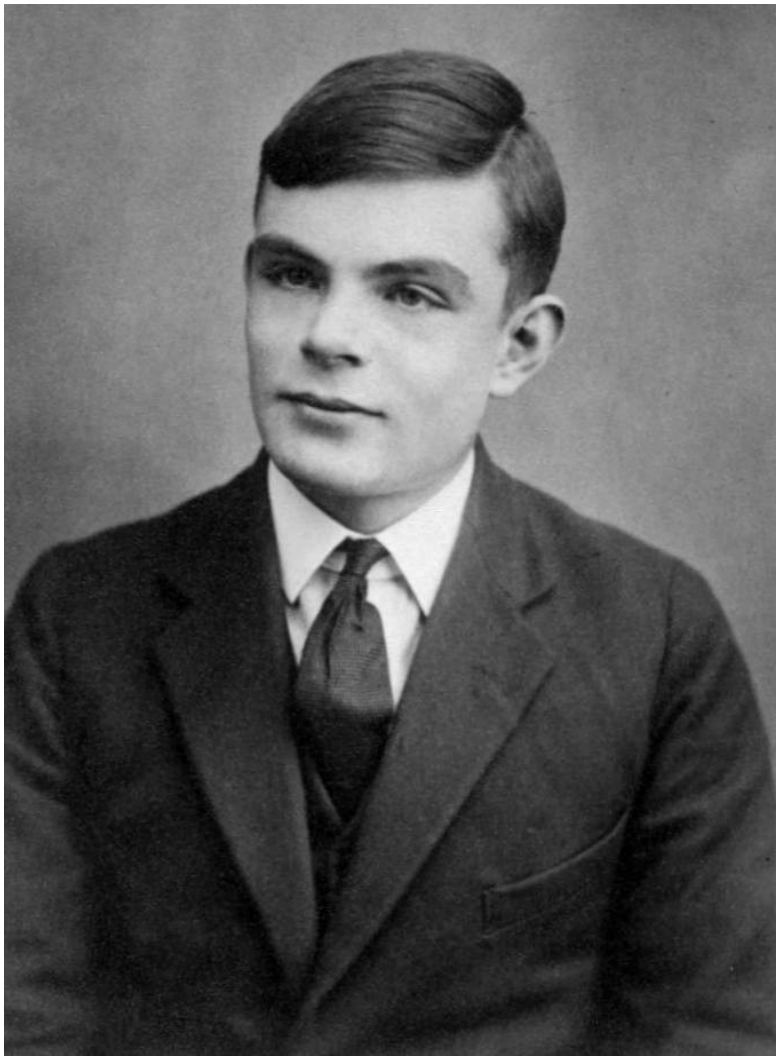
- Hilbert’s agenda is both a spectacular success and a spectacular failure
- Inspires some of the most impactful theoretical work in mathematical history, human history
- Brings us heroes like **Alan Turing** and **Kurt Gödel**!

# 1900 Hilbert: optimism!
# Reality: womp womp

- Hilbert's agenda is both a spectacular success and a spectacular failure
- Inspires some of the most impactful theoretical work in mathematical history, human history
- Brings us heroes like **Alan Turing** and **Kurt Gödel**!
- These incredible results came about *because* of Hilbert, but consist of *utterly demolishing* all the pillars of Hilbert's vision of automated knowledge creation, within just a few years

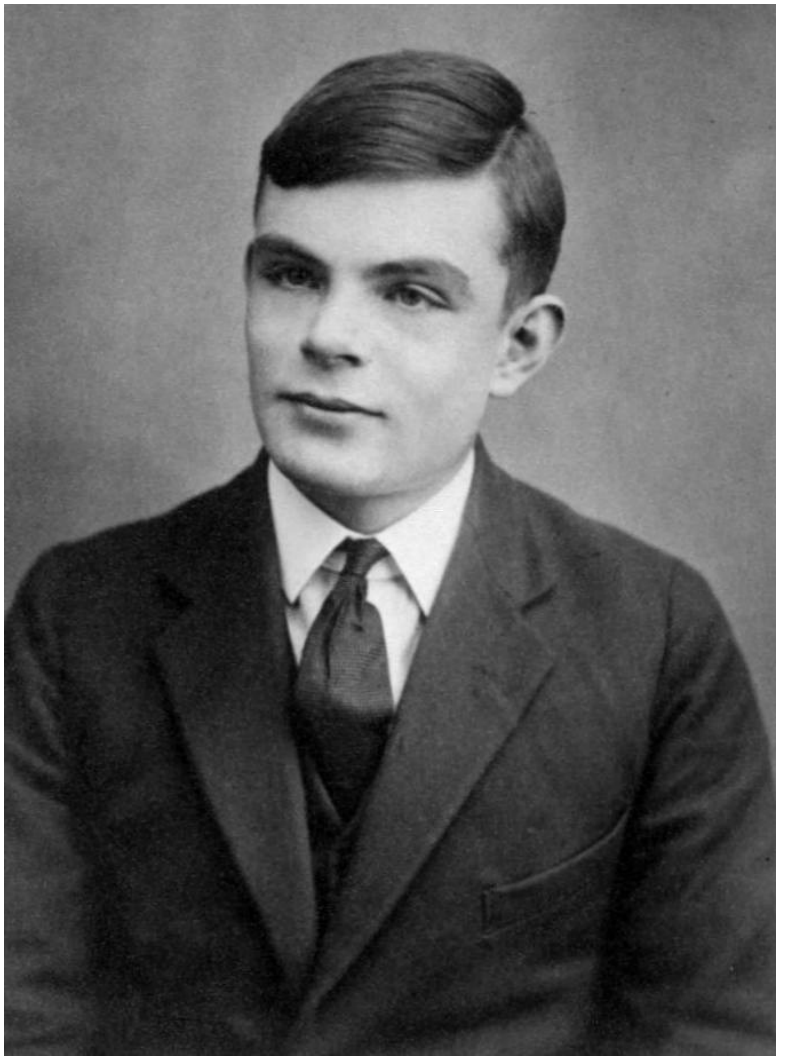
# Alan Turing

- 1912-1954
- Invented Turing Machines (CS103)
- Proved some problems are unsolvable (CS103)
- “Father of AI,” Turing Test
- Along with the largely female computer and codebreaking staff at Bletchley Park, played crucial role in Allied victory in WWII
- CS equivalent of Nobel Prize is named Turing Award in his honor



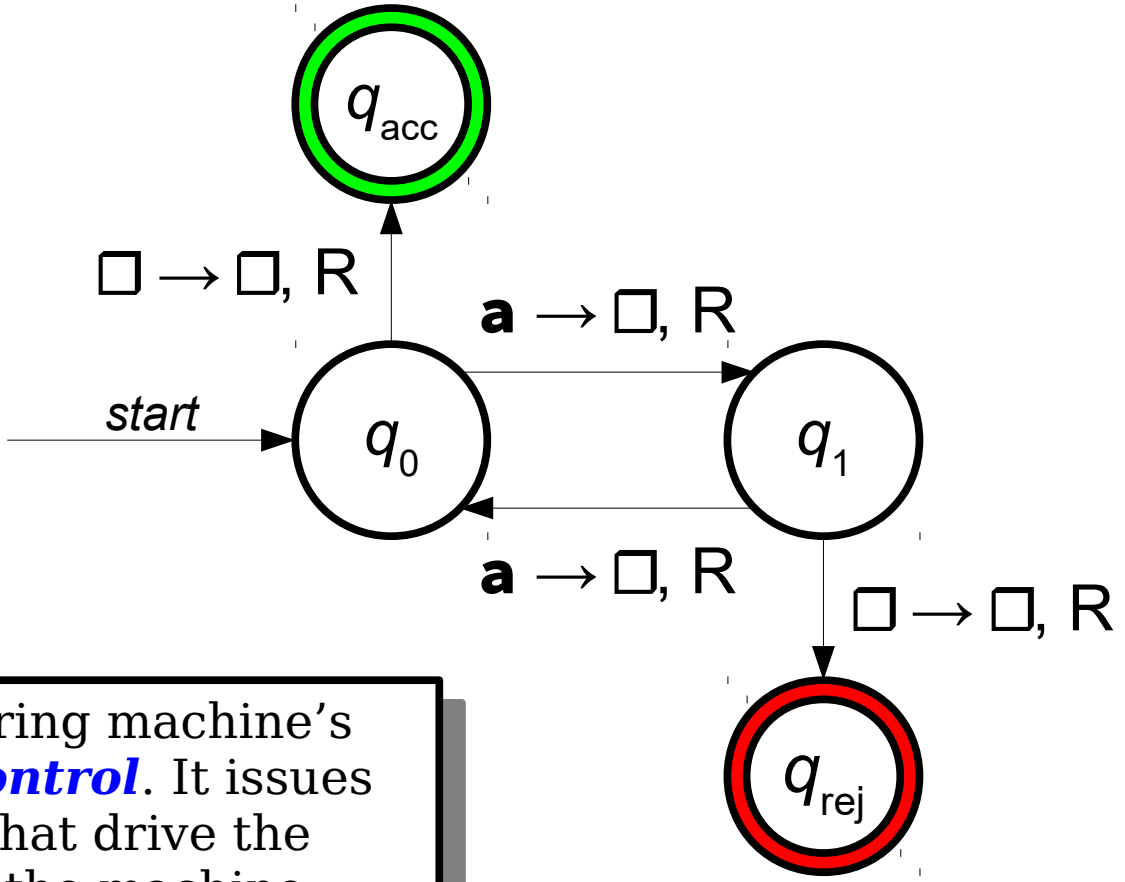
# Alan Turing

- Happily out gay man
- Participated in London's lively LGBTQ+ scene
- In 1952, Turing was robbed, reported the robbery to police, who instead investigated *him* for homosexuality
- Convicted, sentenced to hormonal "treatments"
- Died by suicide in 1954
- Not officially pardoned by the government until 2013



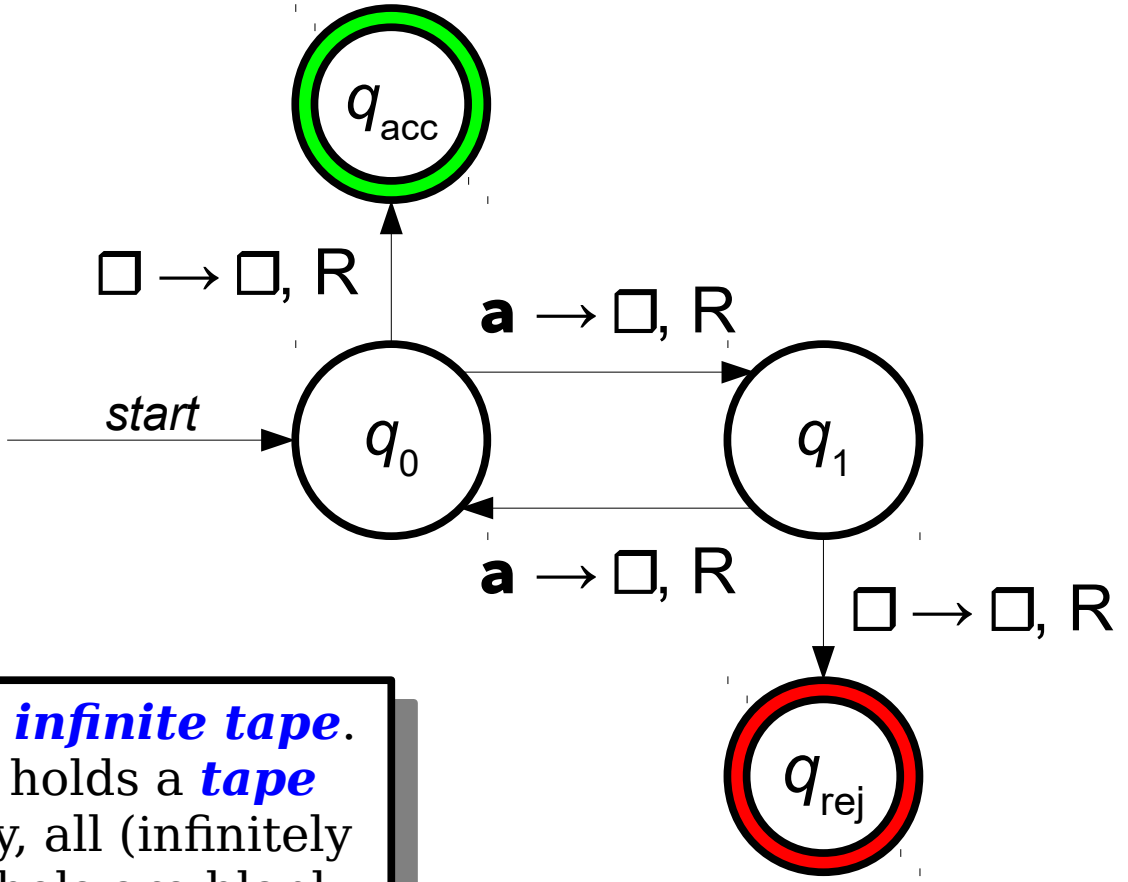
# What Turing Discovered

# A Simple Turing Machine

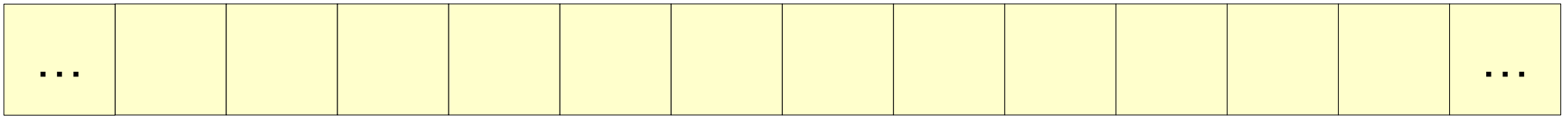



This is the Turing machine's ***finite state control***. It issues commands that drive the operation of the machine.

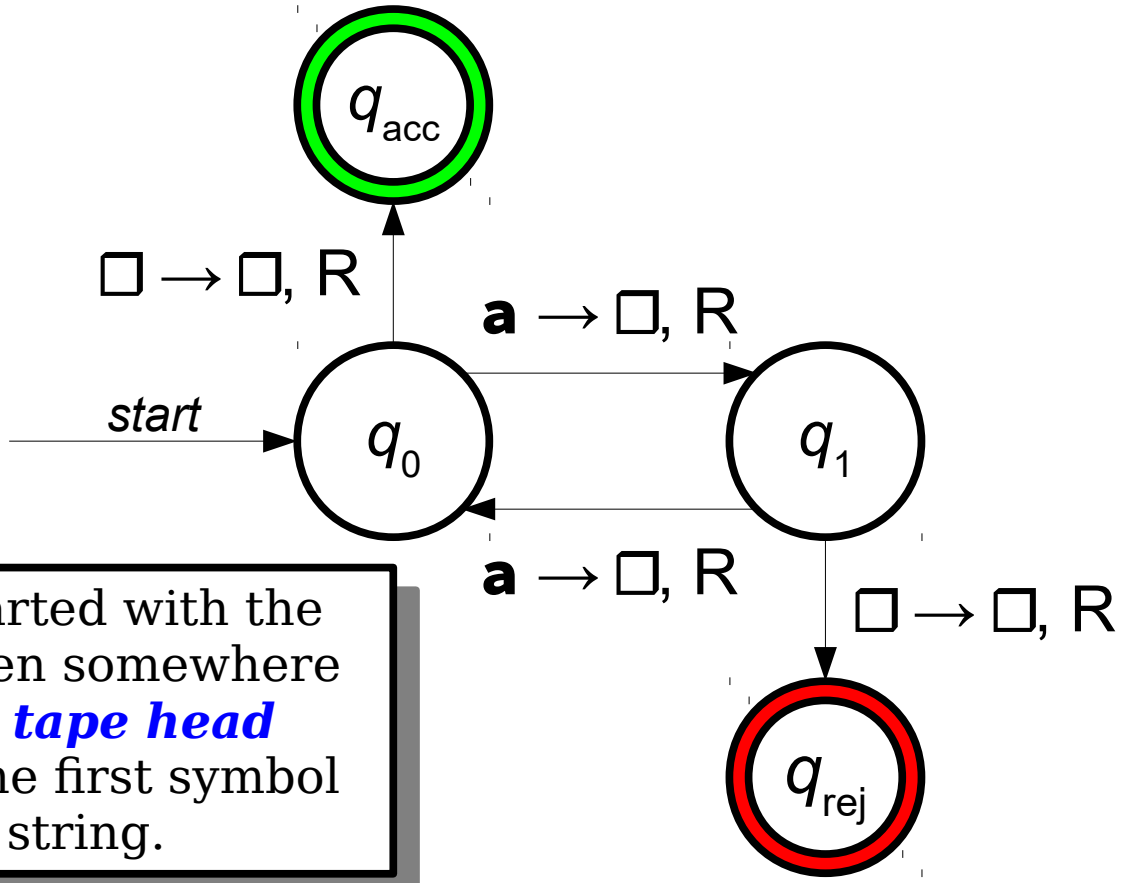
# A Simple Turing Machine



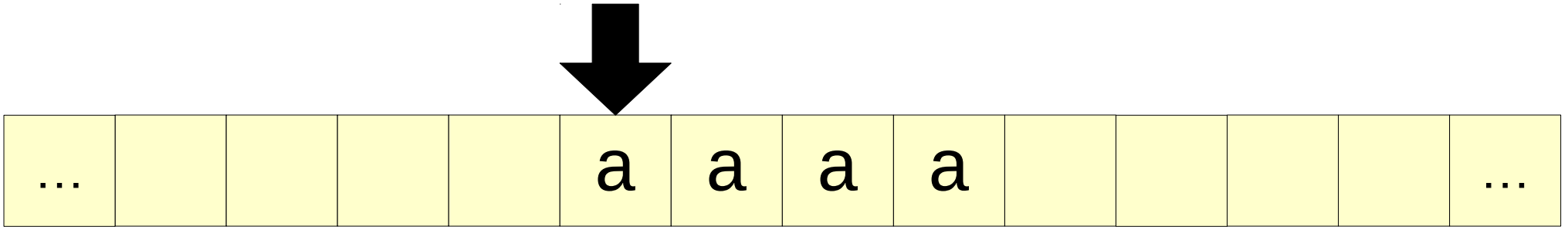

This is the TM's ***infinite tape***. Each tape cell holds a ***tape symbol***. Initially, all (infinitely many) tape symbols are blank.



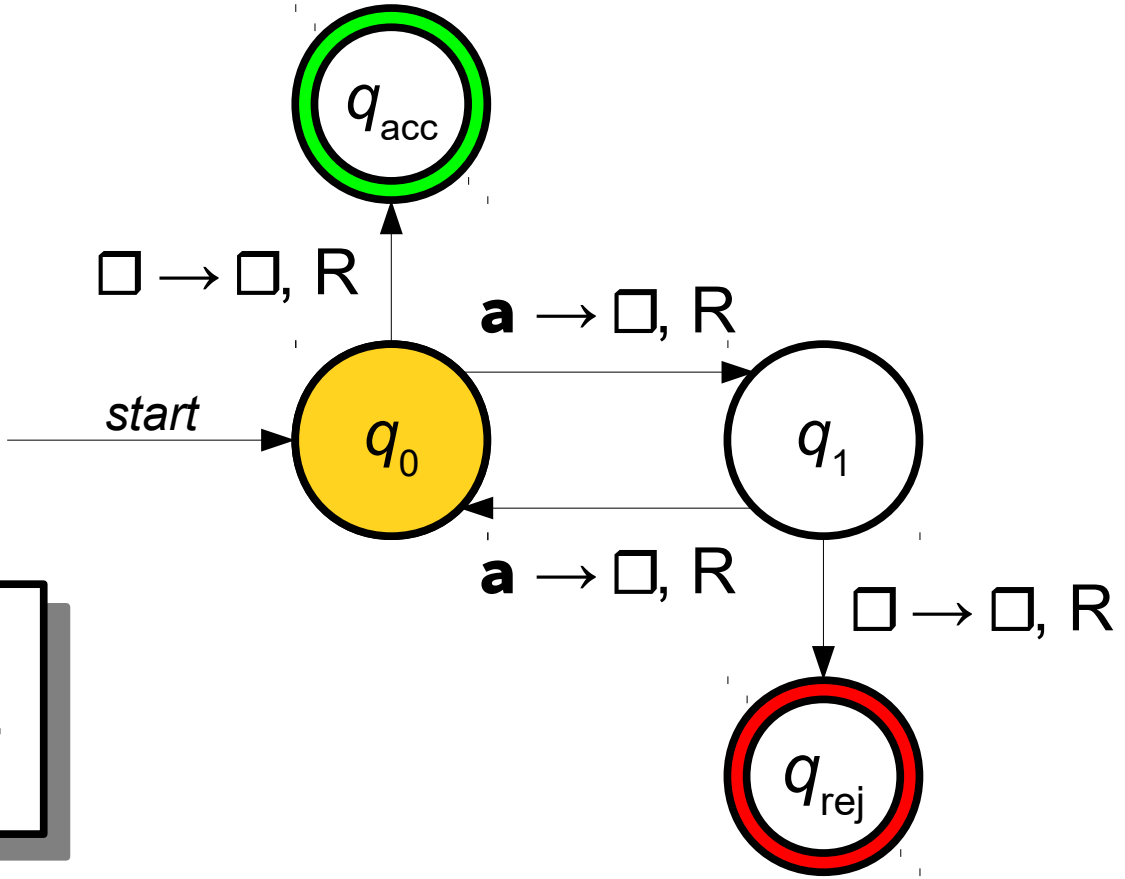
# A Simple Turing Machine



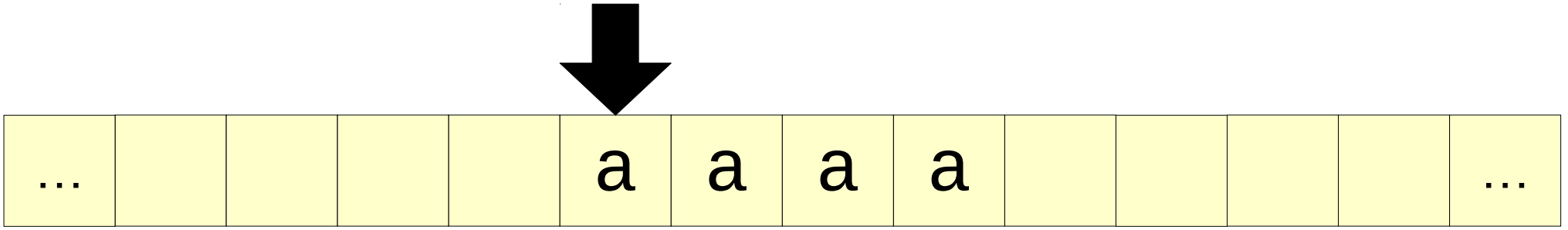

The machine is started with the ***input string*** written somewhere on the tape. The ***tape head*** initially points to the first symbol of the input string.



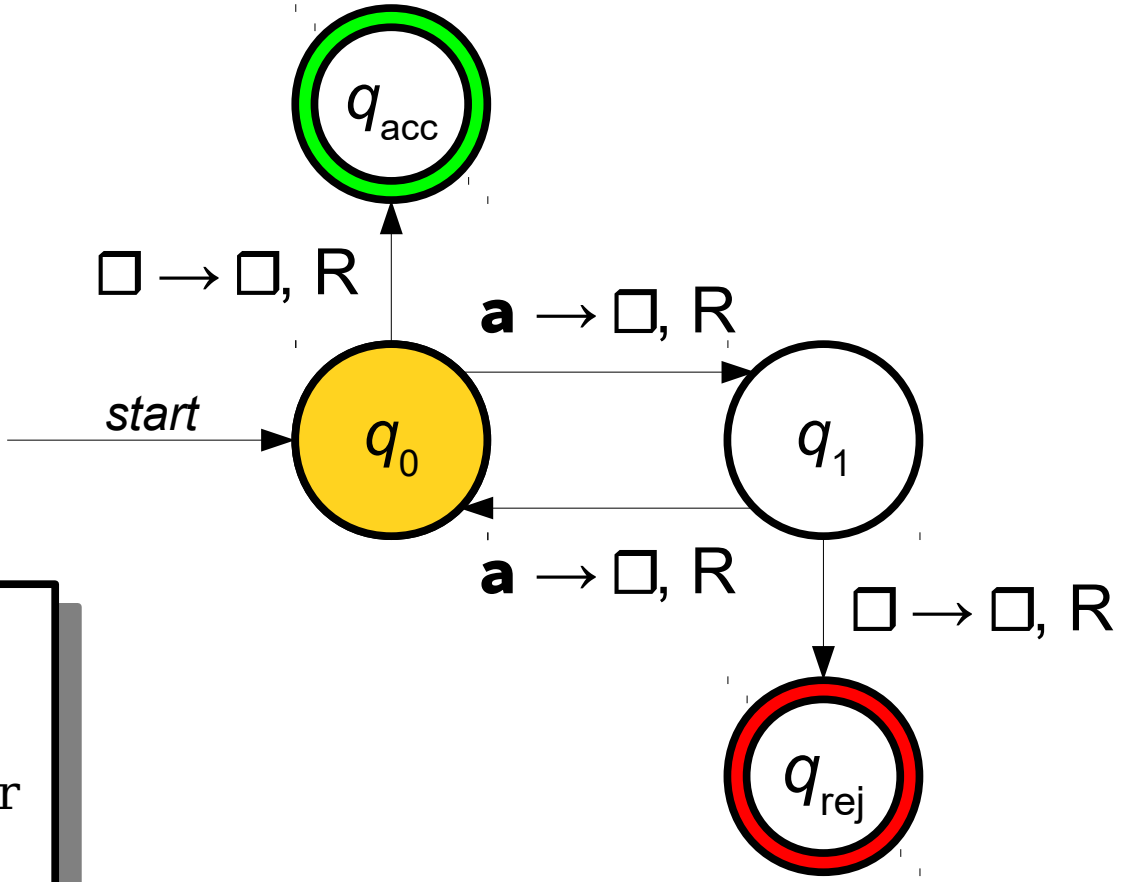
# A Simple Turing Machine



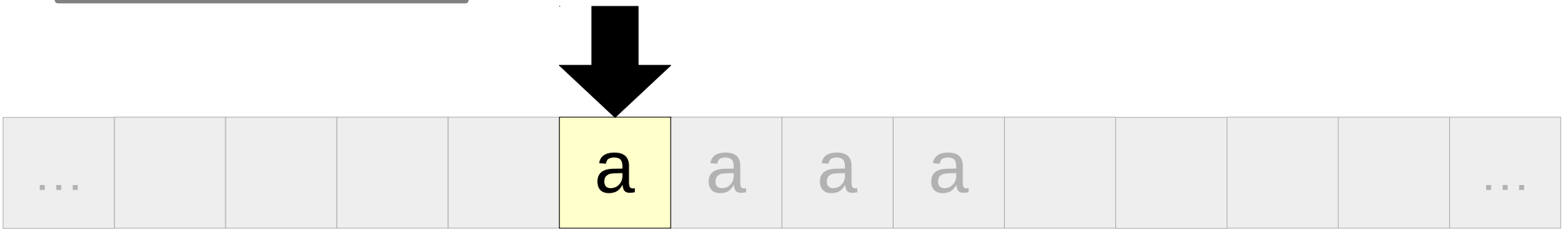

Like DFAs and NFAs, TMs begin execution in their ***start state***.



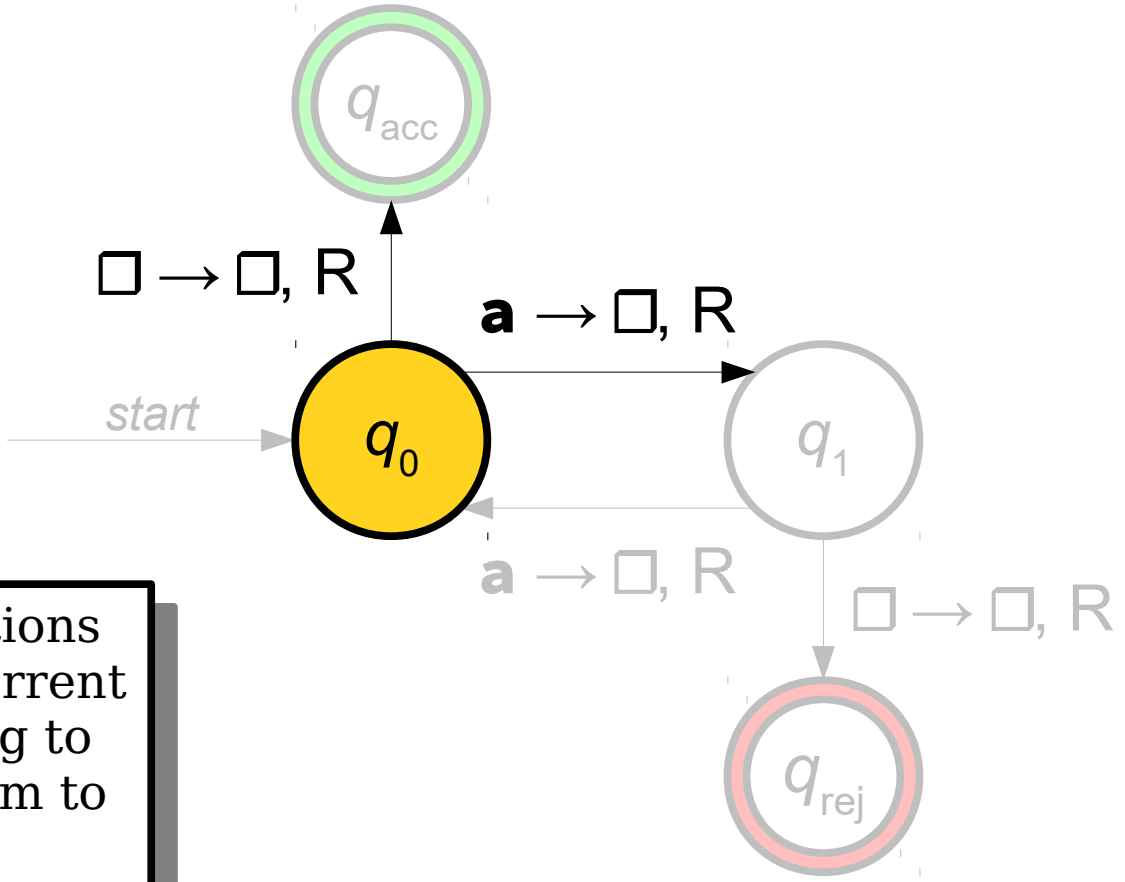
# A Simple Turing Machine



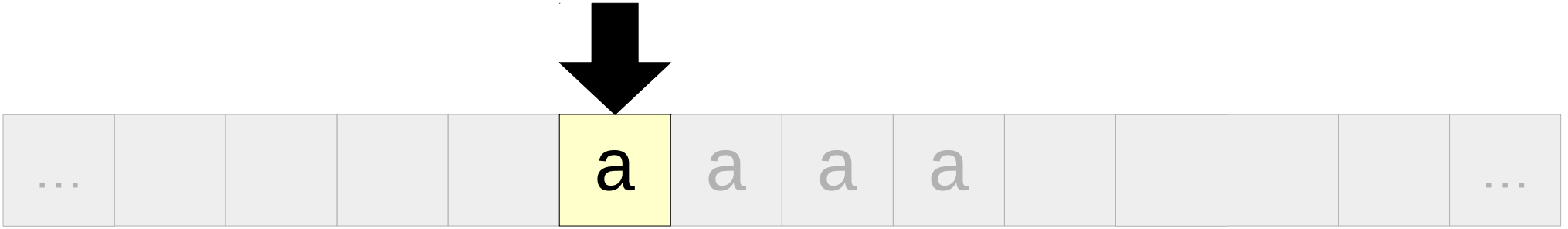

At each step, the TM only looks at the symbol immediately under the ***tape head***.



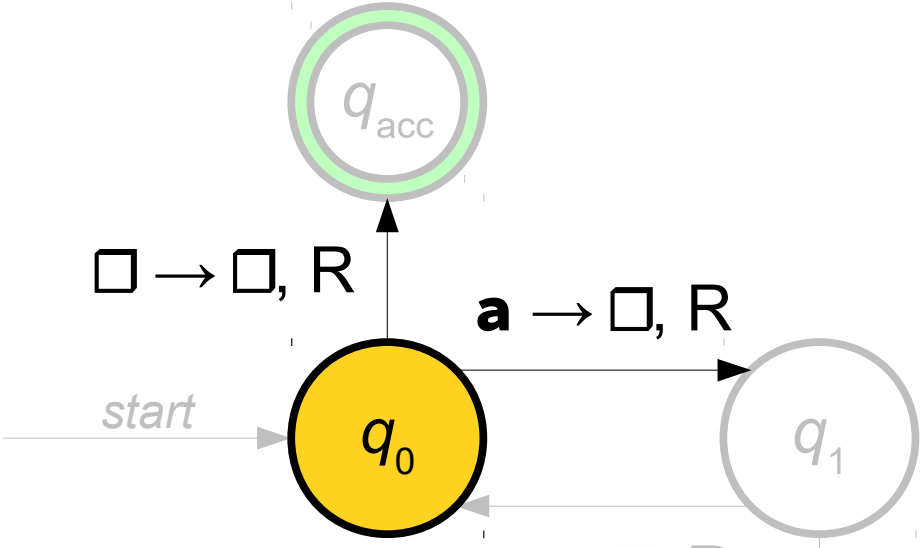
# A Simple Turing Machine




These two transitions originate at the current state. We're going to choose one of them to follow.



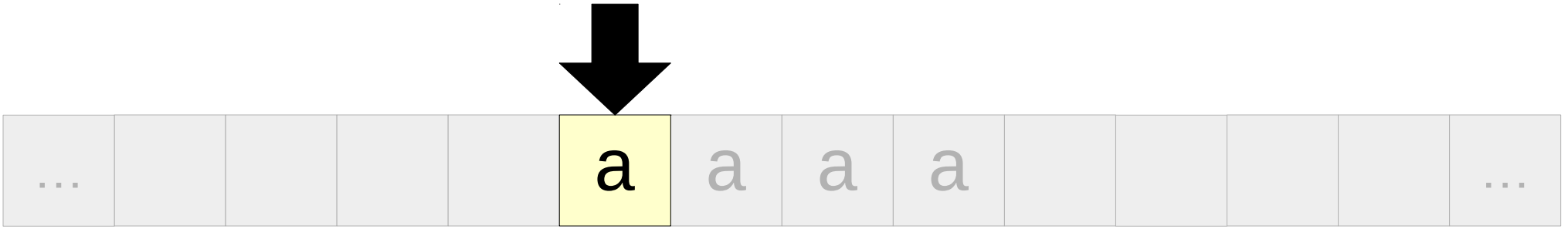
# A Simple Turing Machine



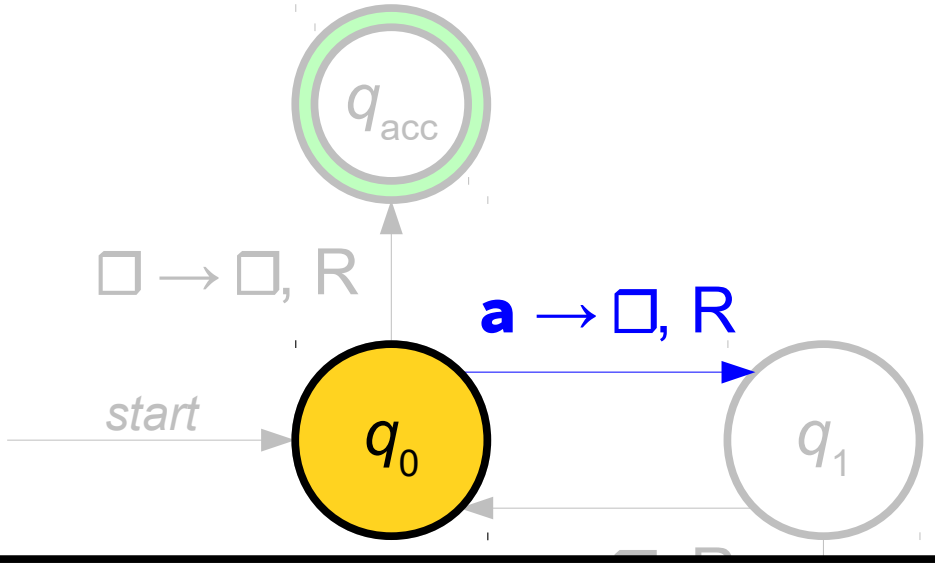

Each transition has the form

***read → write, dir***

and means "if symbol ***read*** is under the tape head, replace it with ***write*** and move the tape head in direction ***dir*** (L or R). The □ symbol denotes a blank cell.



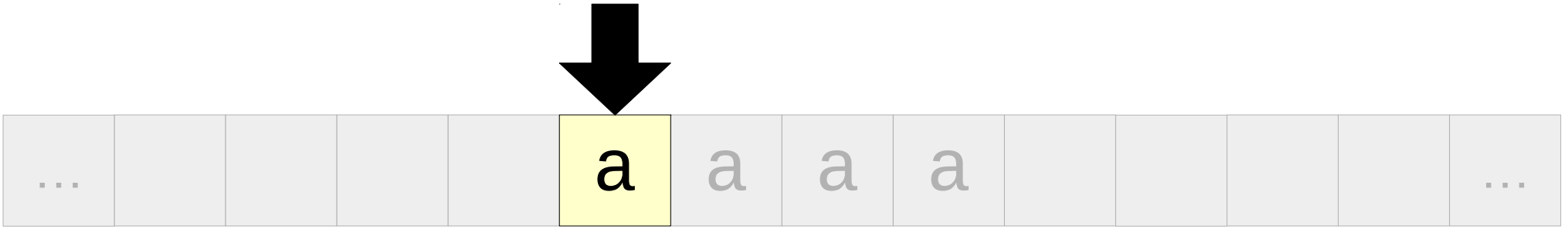
# A Simple Turing Machine



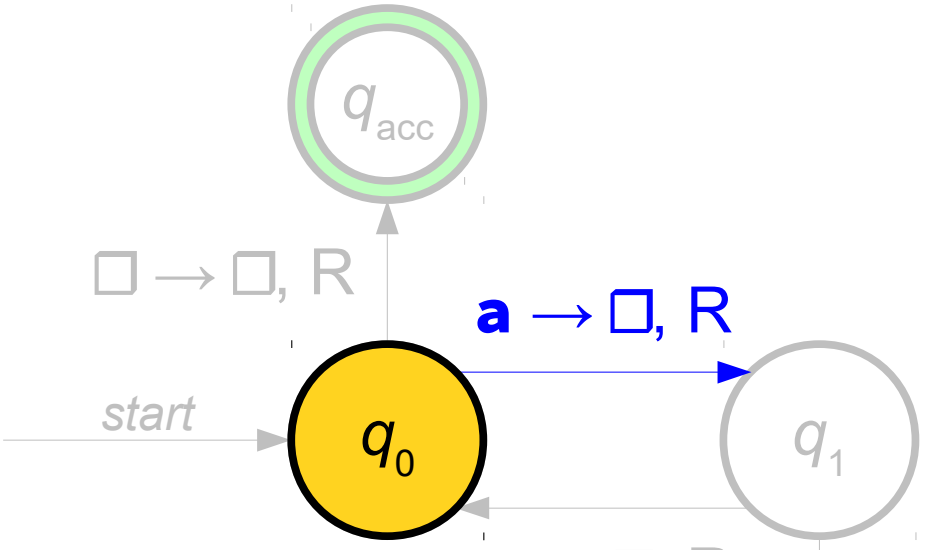

Each transition has the form

***read → write, dir***

and means "if symbol ***read*** is under the tape head, replace it with ***write*** and move the tape head in direction ***dir*** (L or R). The □ symbol denotes a blank cell.



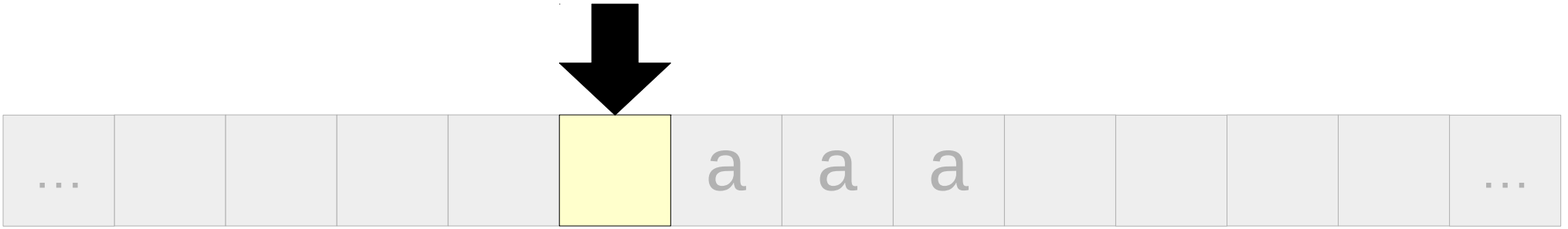
# A Simple Turing Machine



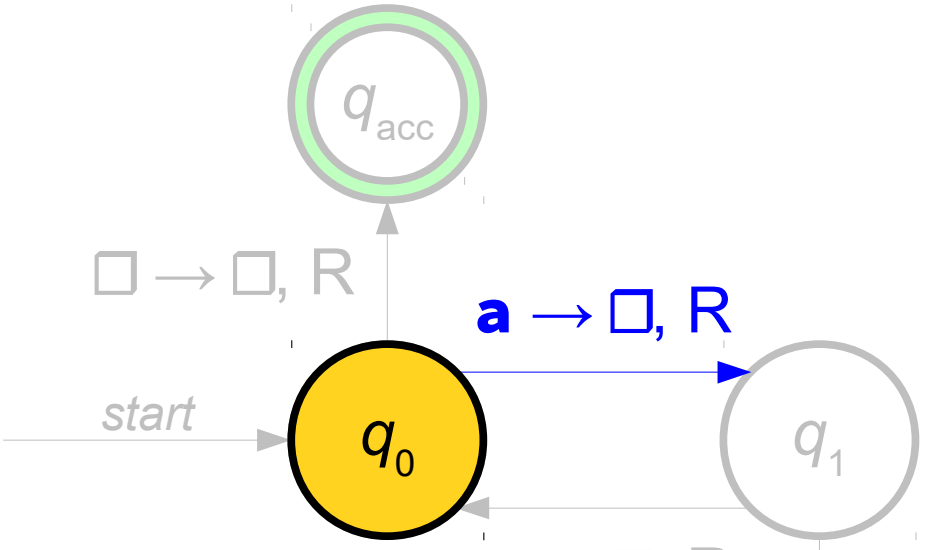

Each transition has the form

***read → write, dir***

and means "if symbol ***read*** is under the tape head, replace it with ***write*** and move the tape head in direction ***dir*** (L or R). The □ symbol denotes a blank cell.



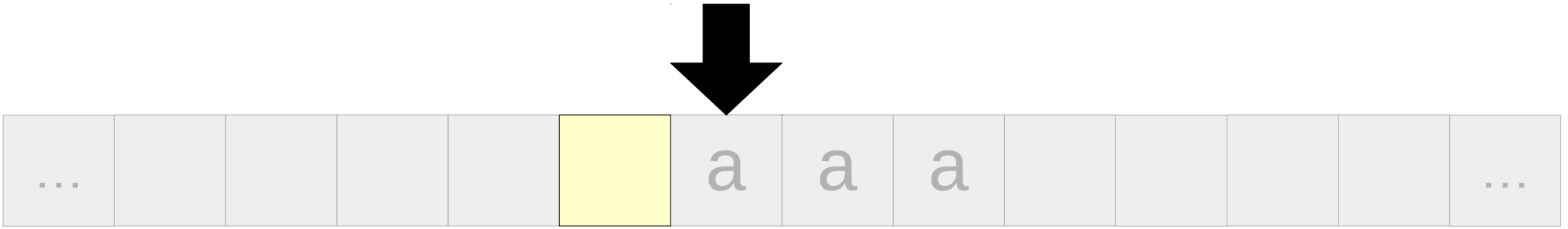
# A Simple Turing Machine



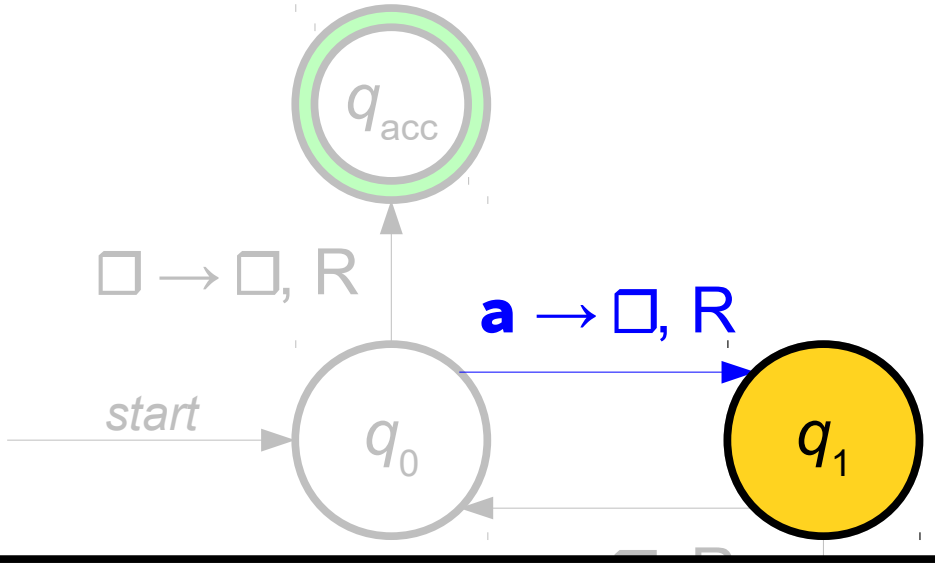

Each transition has the form

***read → write, dir***

and means "if symbol ***read*** is under the tape head, replace it with ***write*** and move the tape head in direction ***dir*** (L or R). The □ symbol denotes a blank cell.



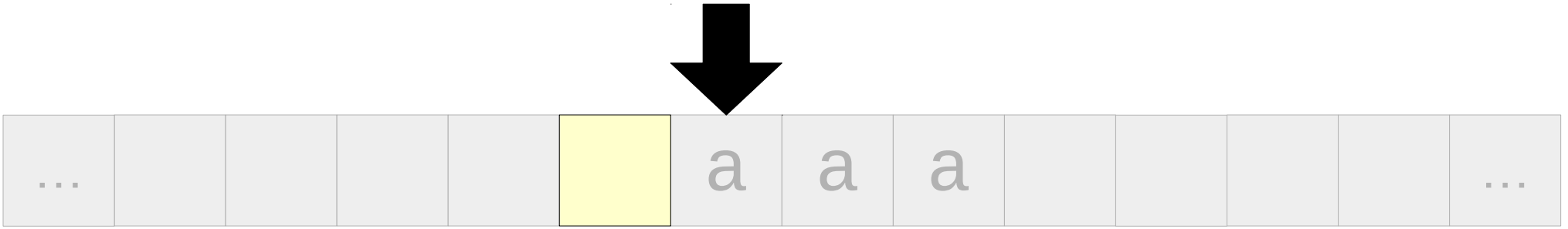
# A Simple Turing Machine



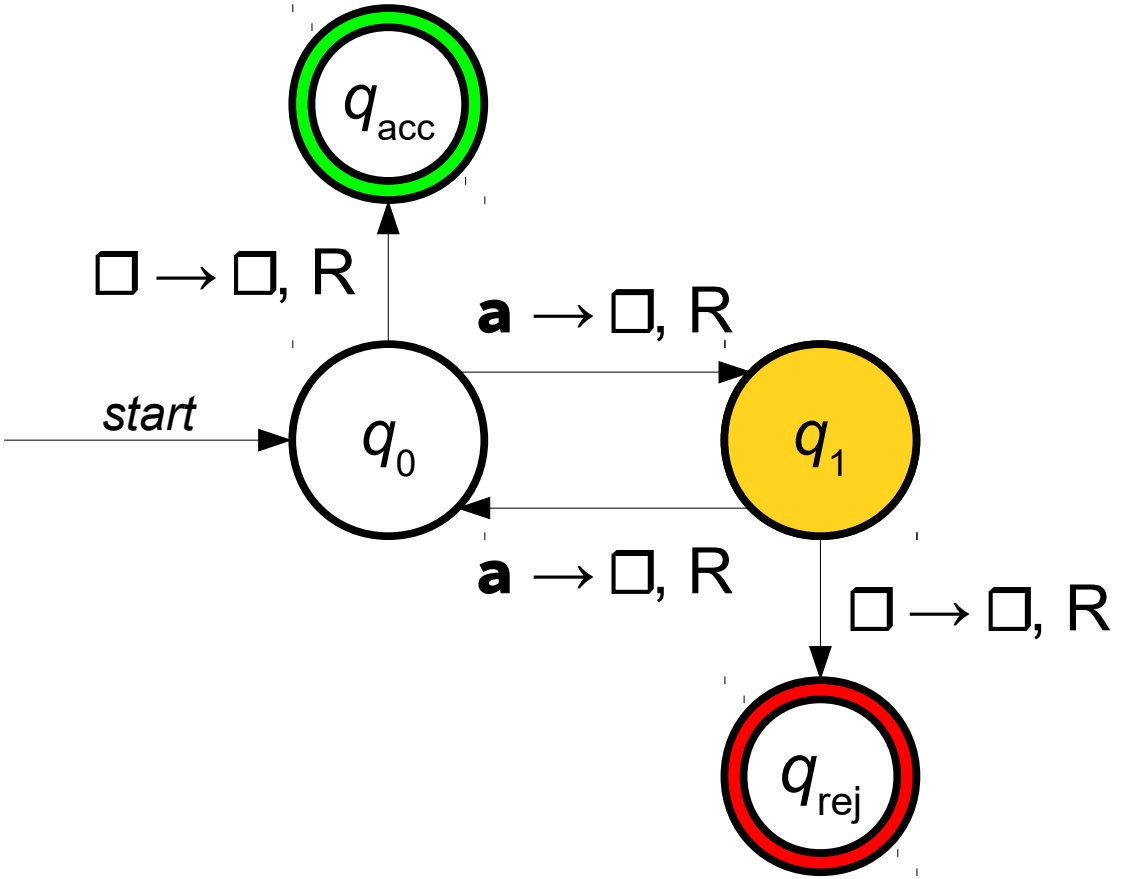

Each transition has the form

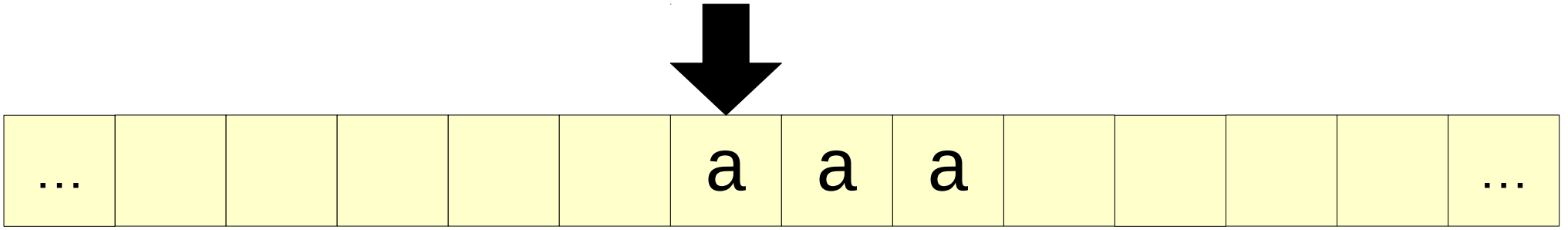
***read → write, dir***

and means "if symbol ***read*** is under the tape head, replace it with ***write*** and move the tape head in direction ***dir*** (L or R). The □ symbol denotes a blank cell.

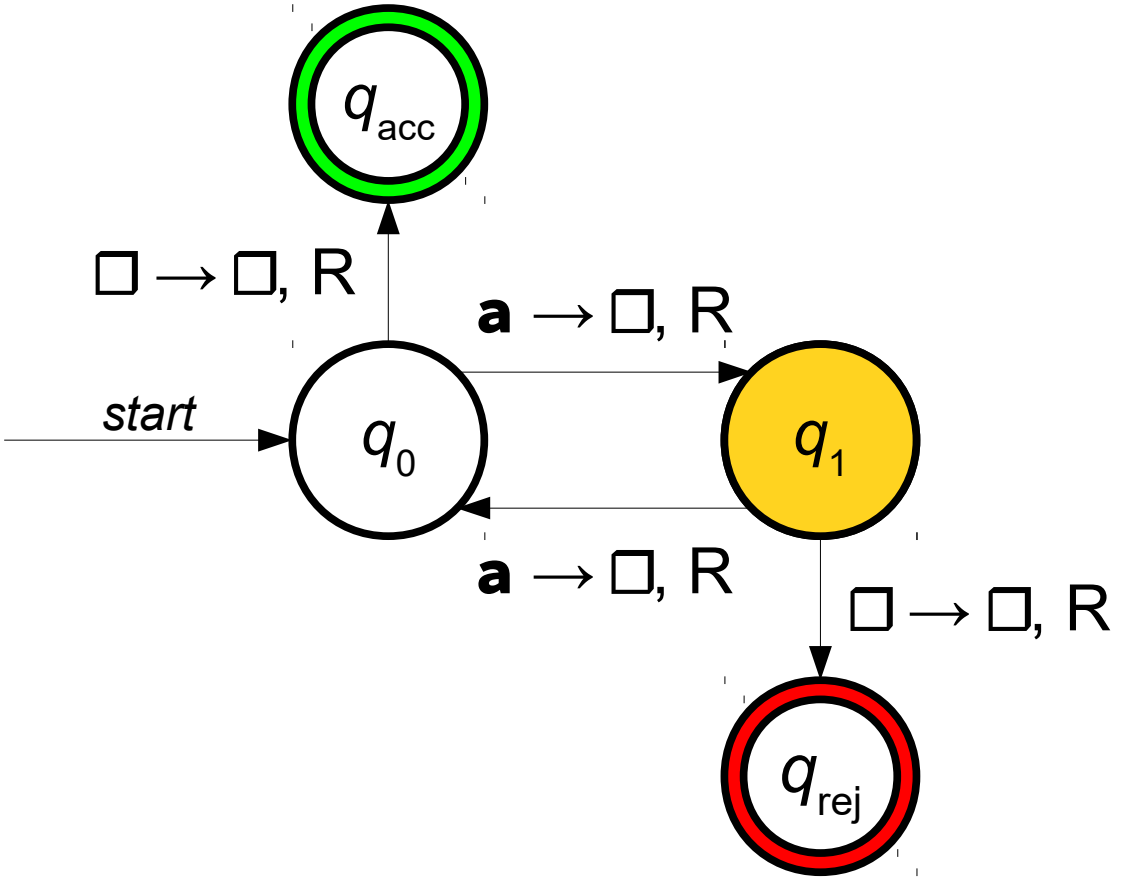


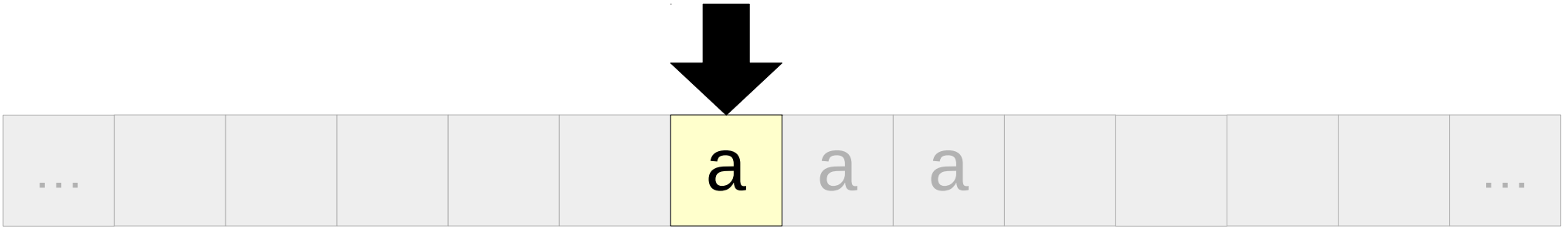
# A Simple Turing Machine



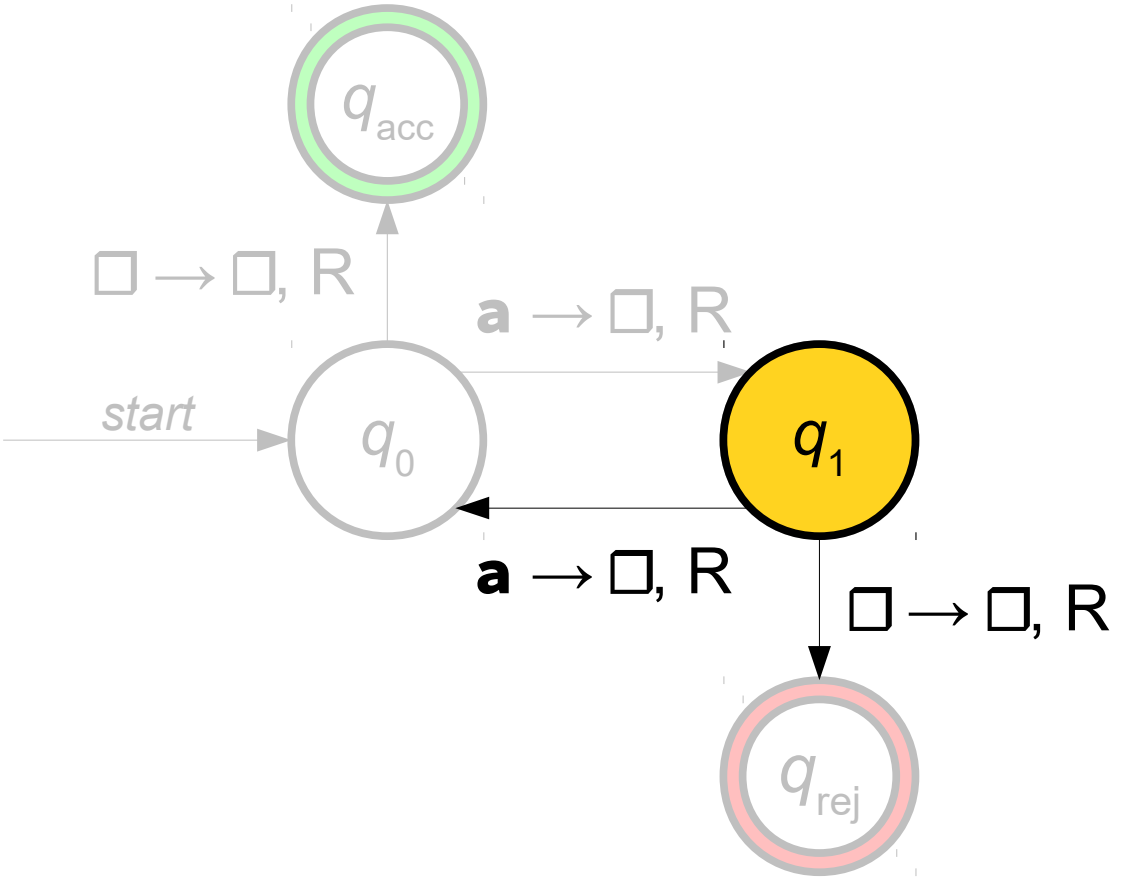



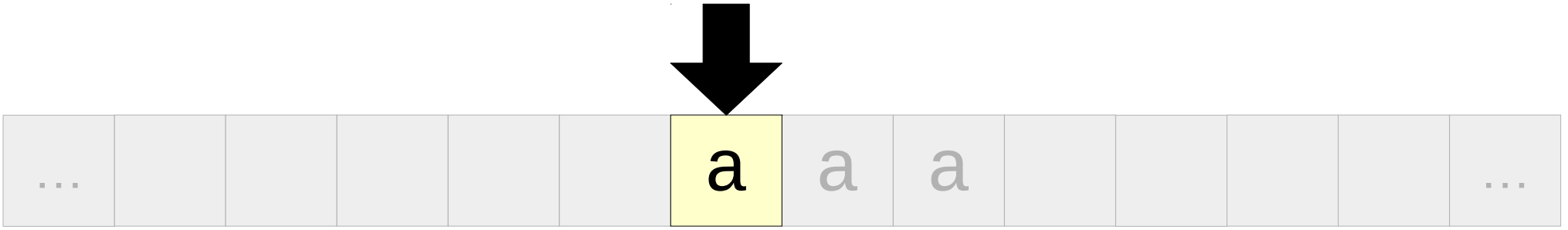
# A Simple Turing Machine



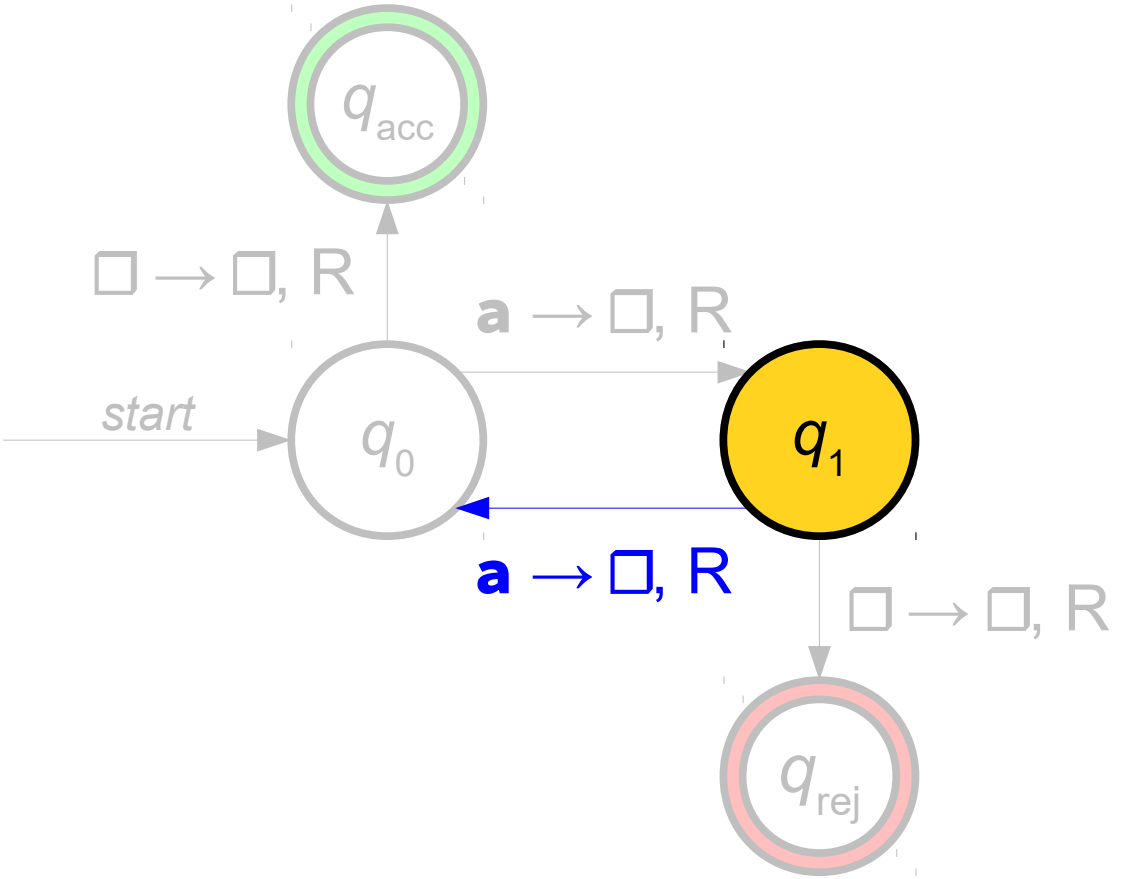



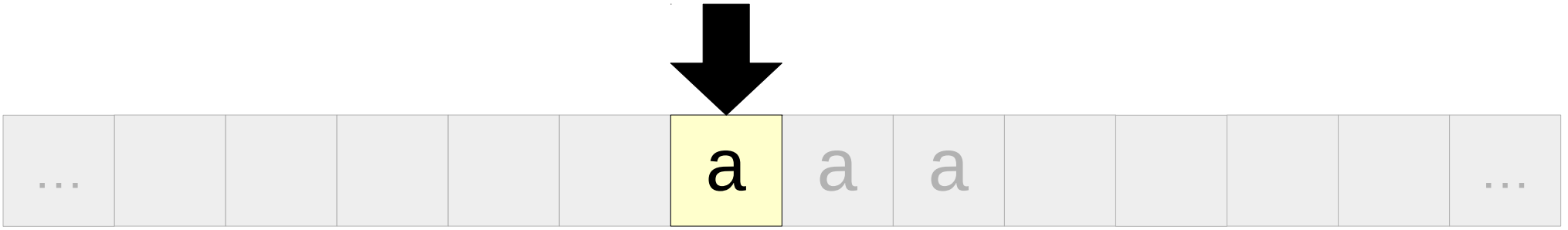
# A Simple Turing Machine



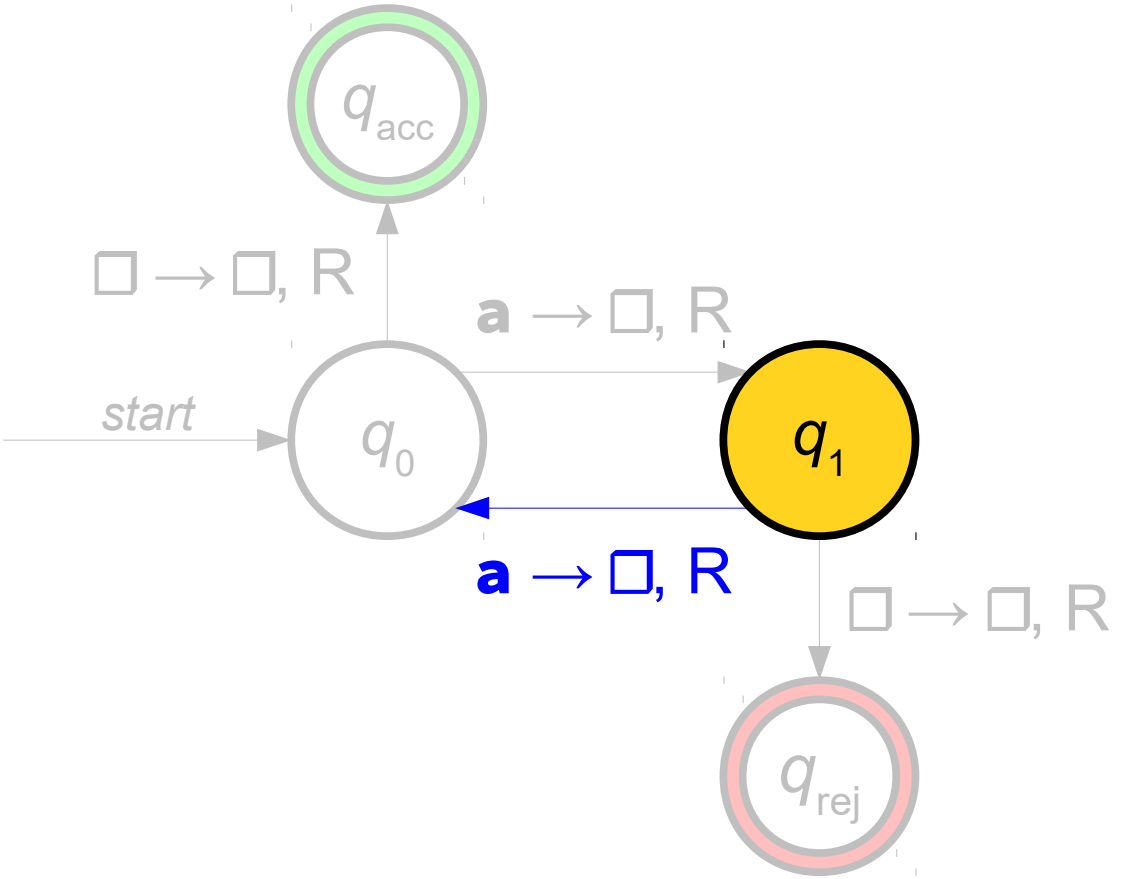



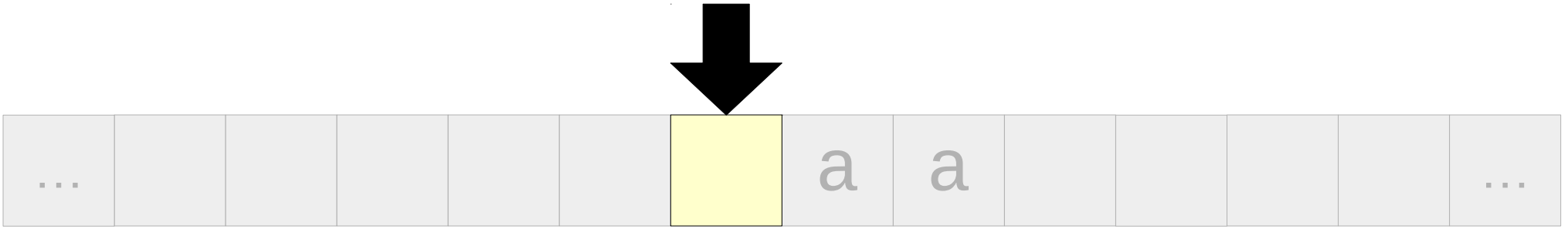
# A Simple Turing Machine



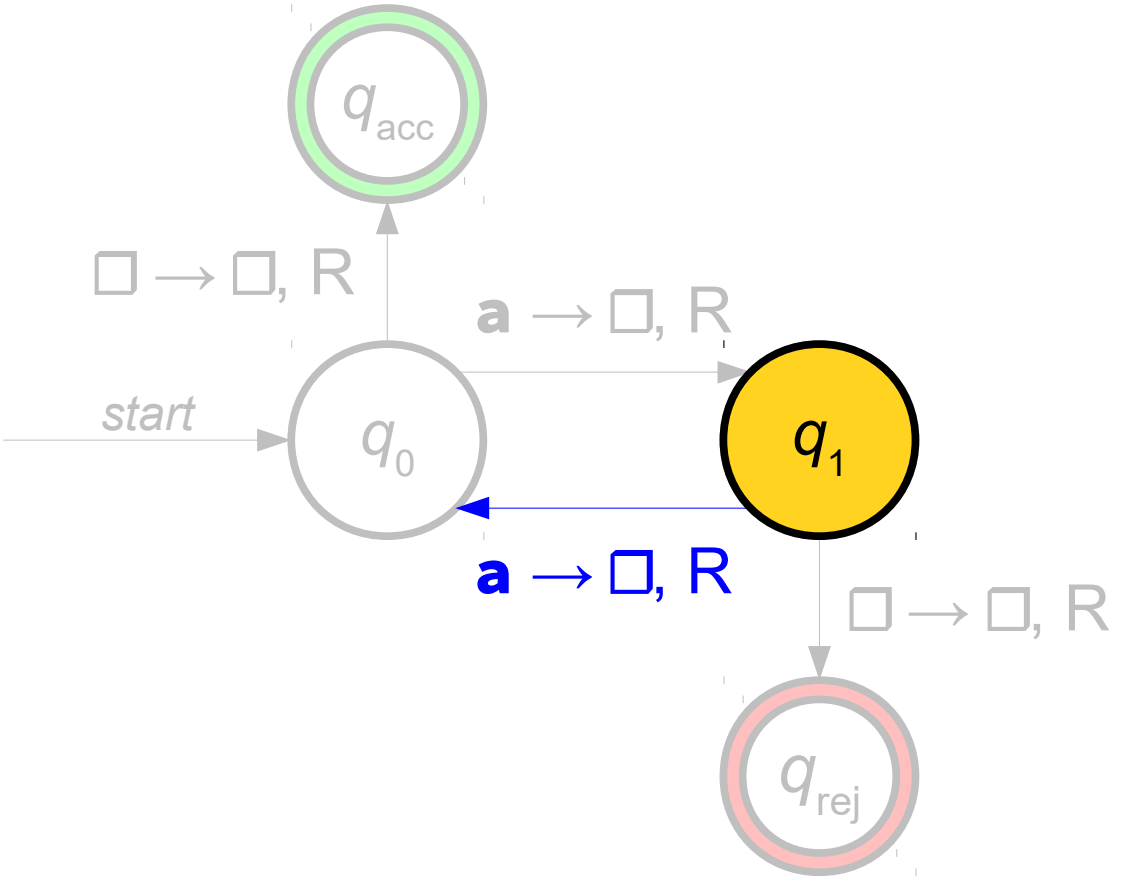



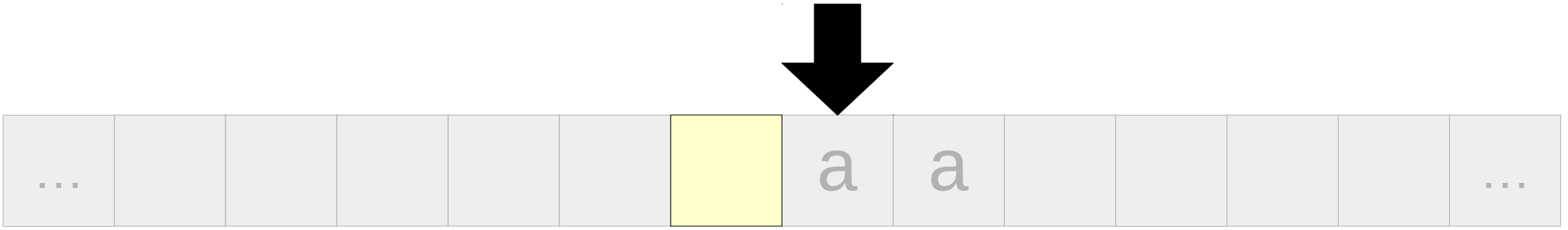
# A Simple Turing Machine



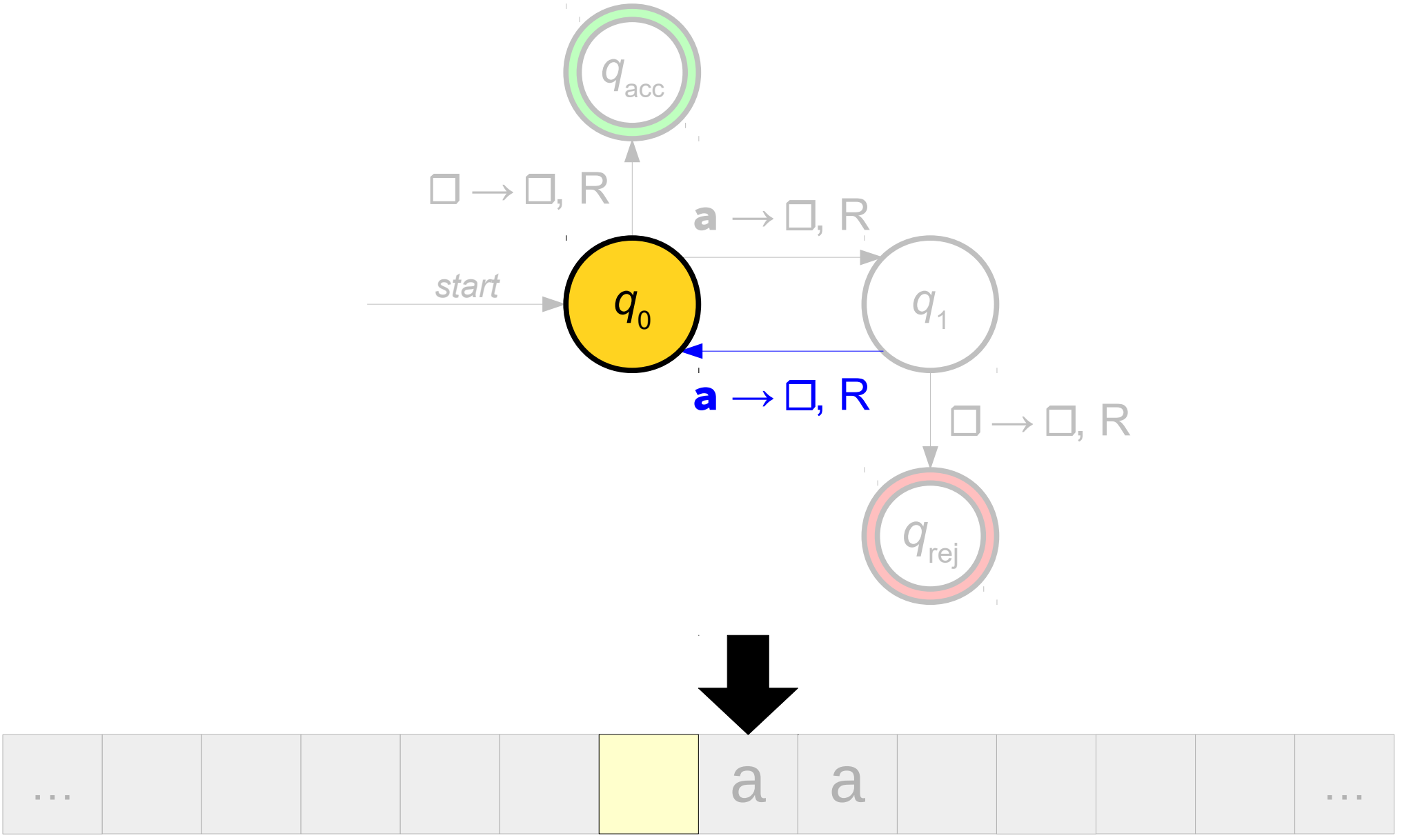



# A Simple Turing Machine

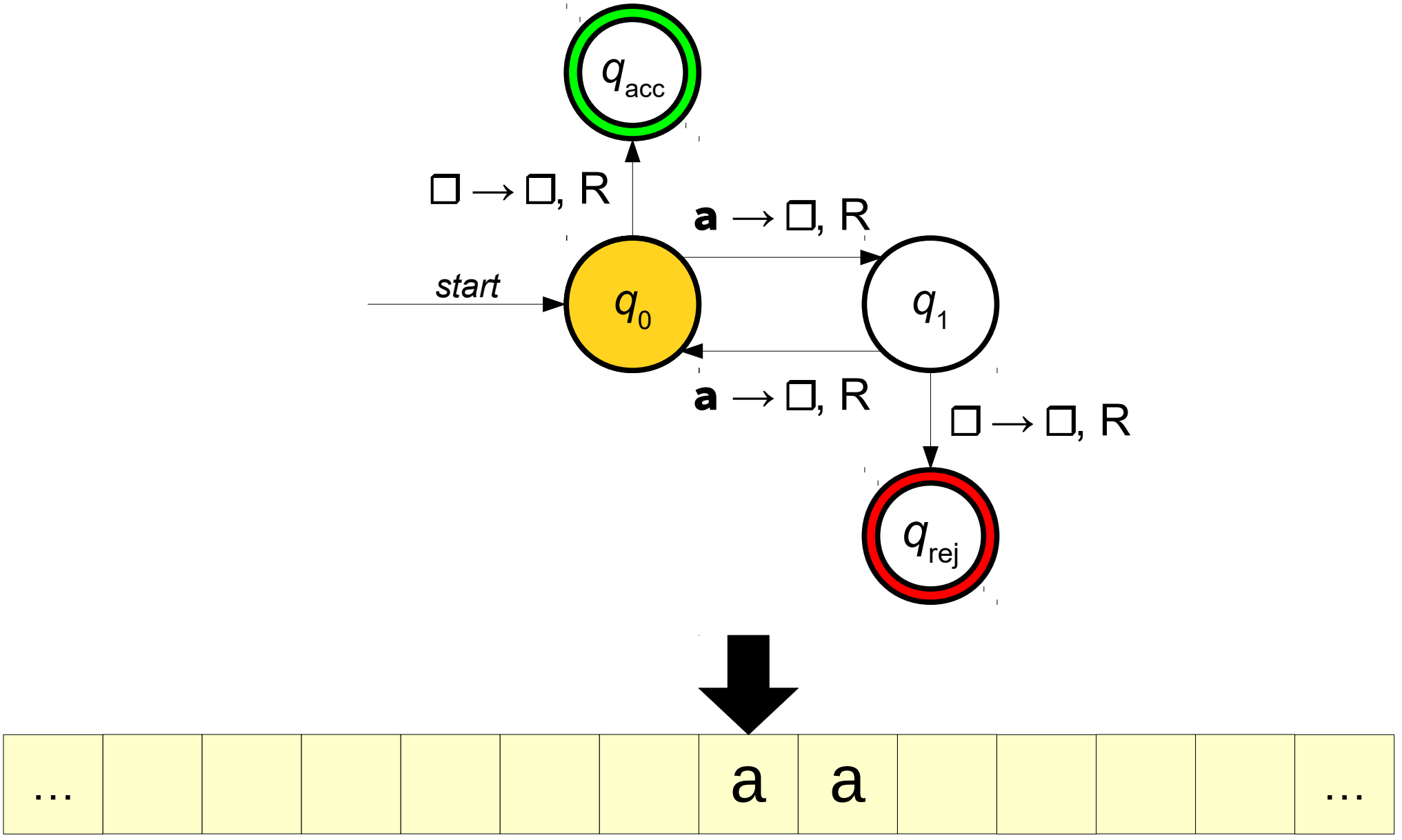


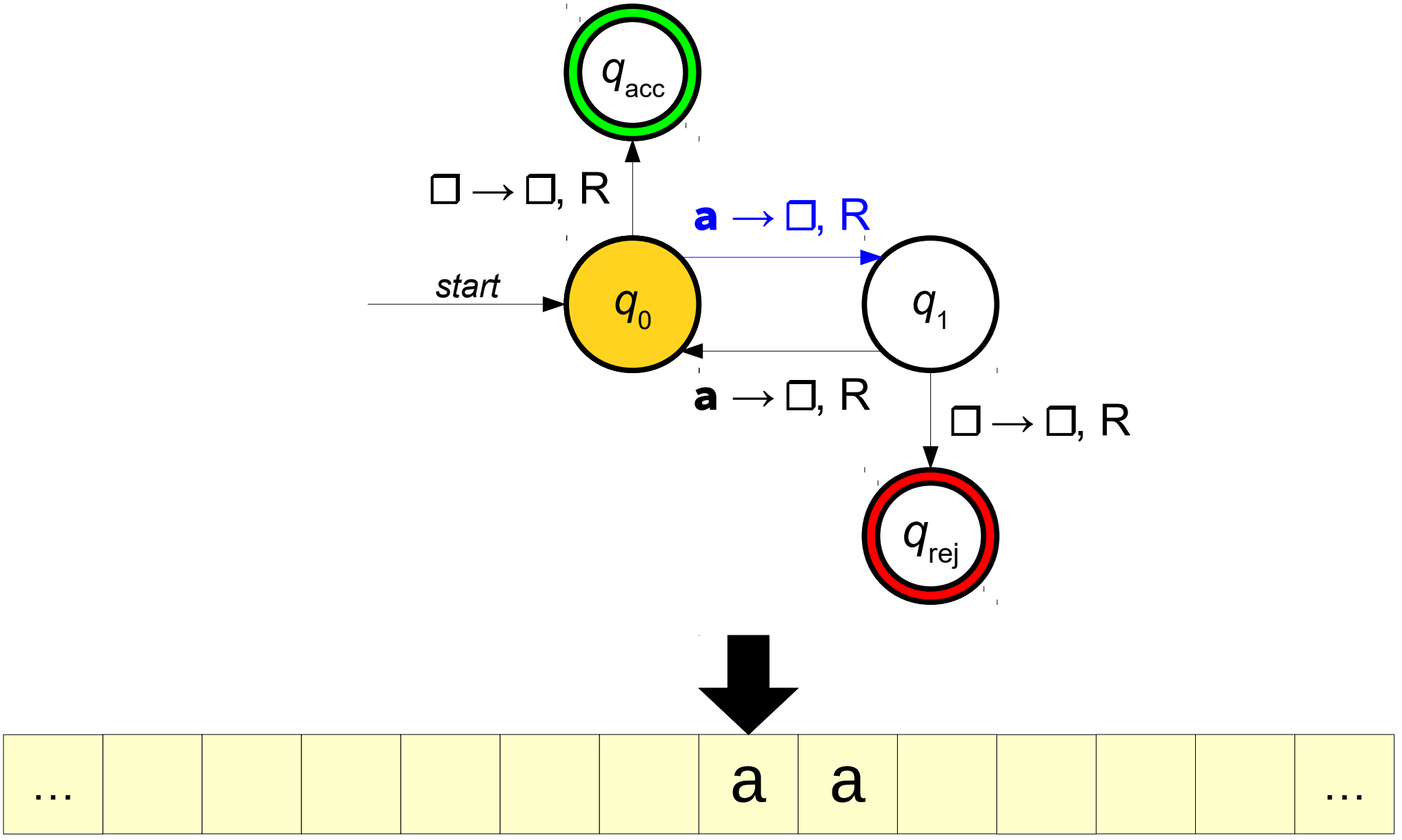
# A Simple Turing Machine



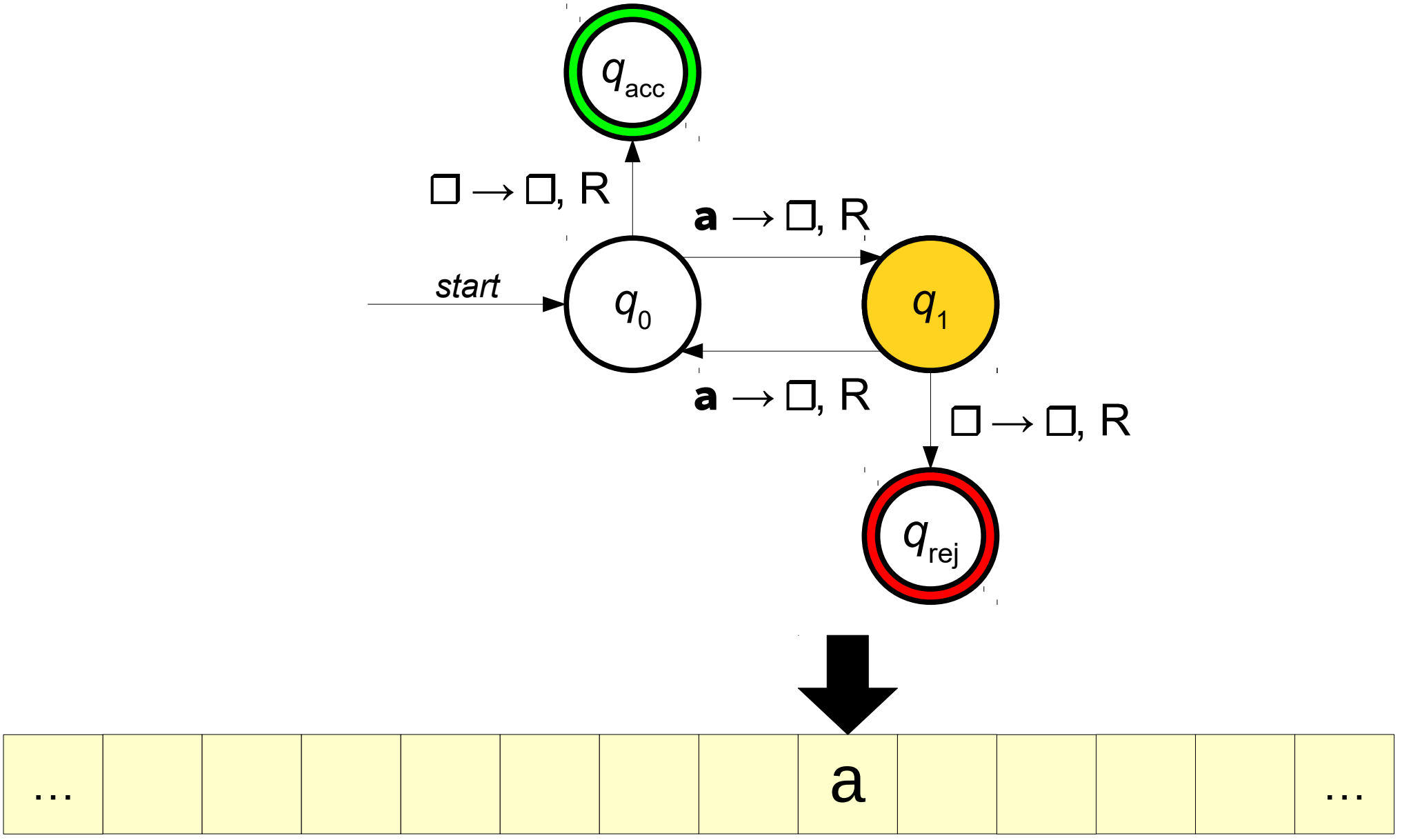
# A Simple Turing Machine



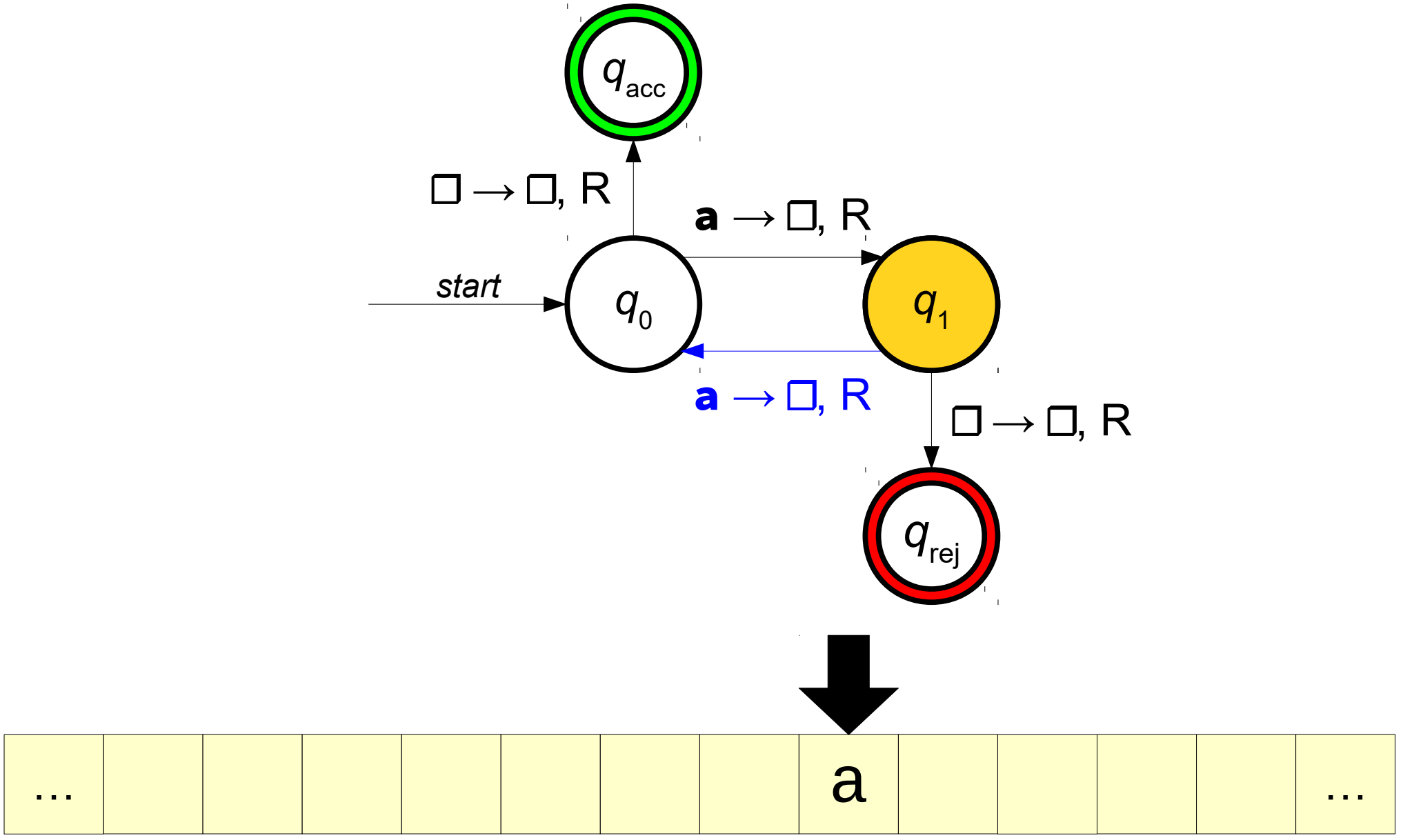
# A Simple Turing Machine



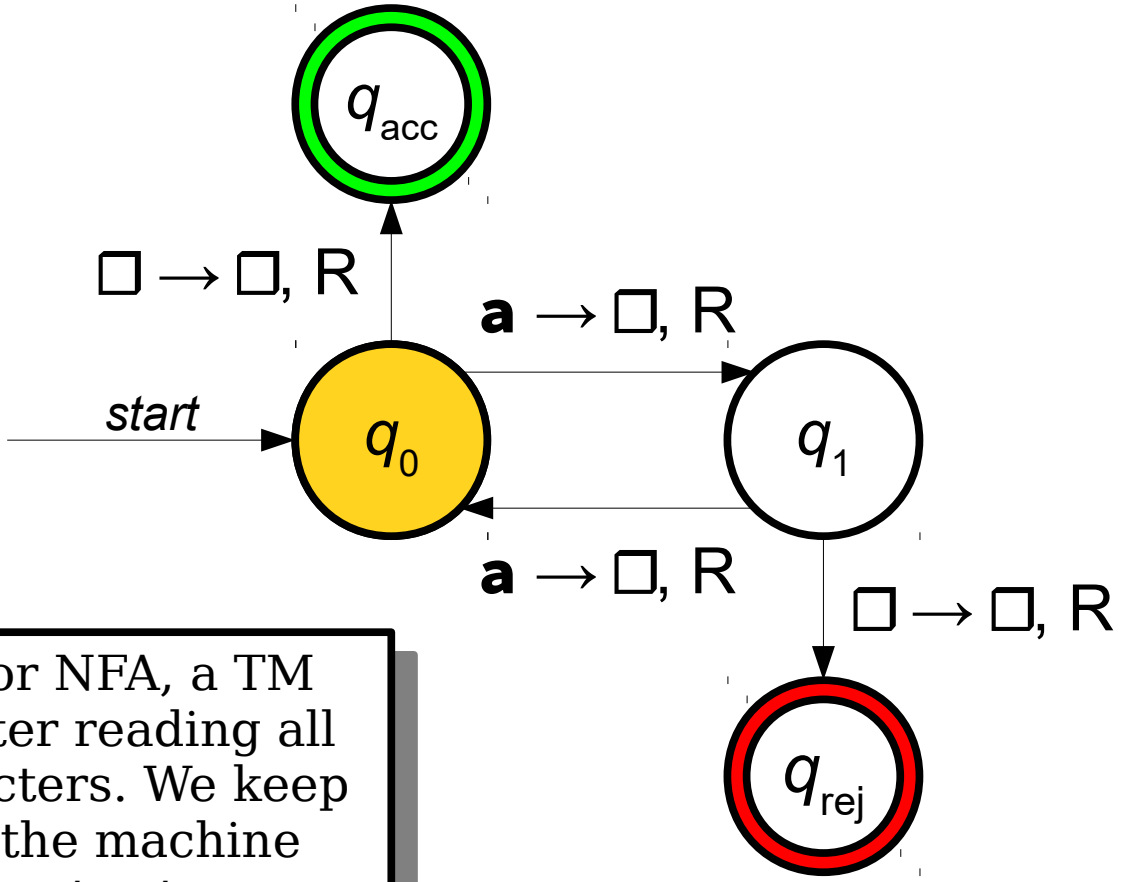
# A Simple Turing Machine



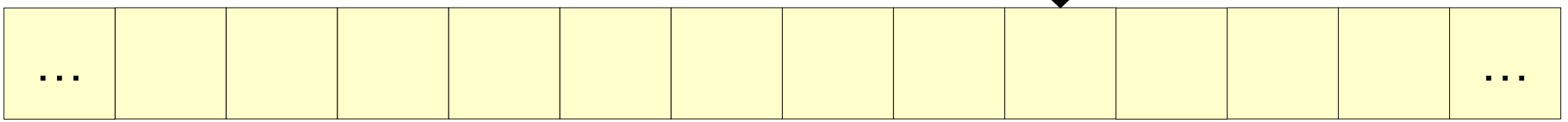
# A Simple Turing Machine



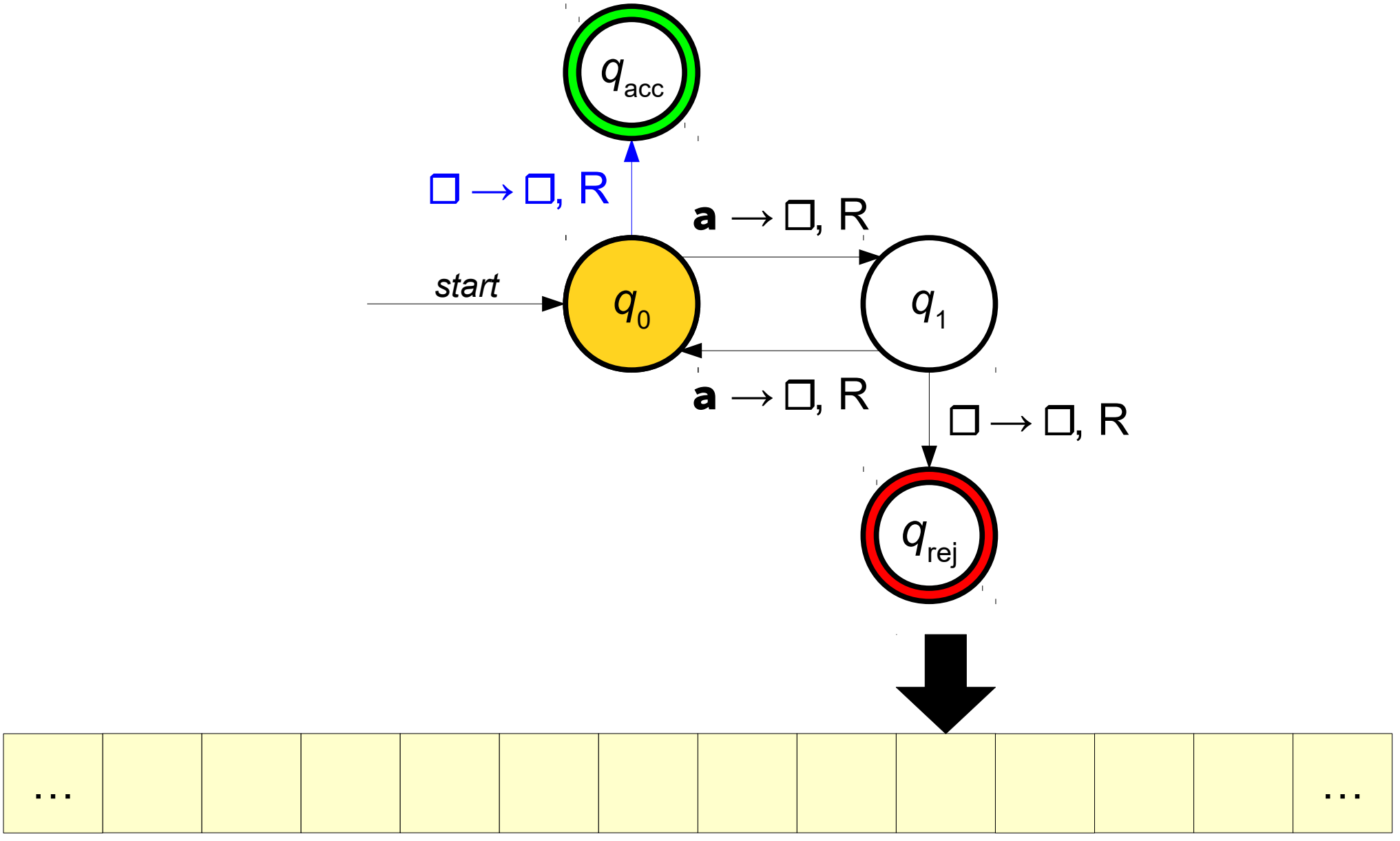
# A Simple Turing Machine



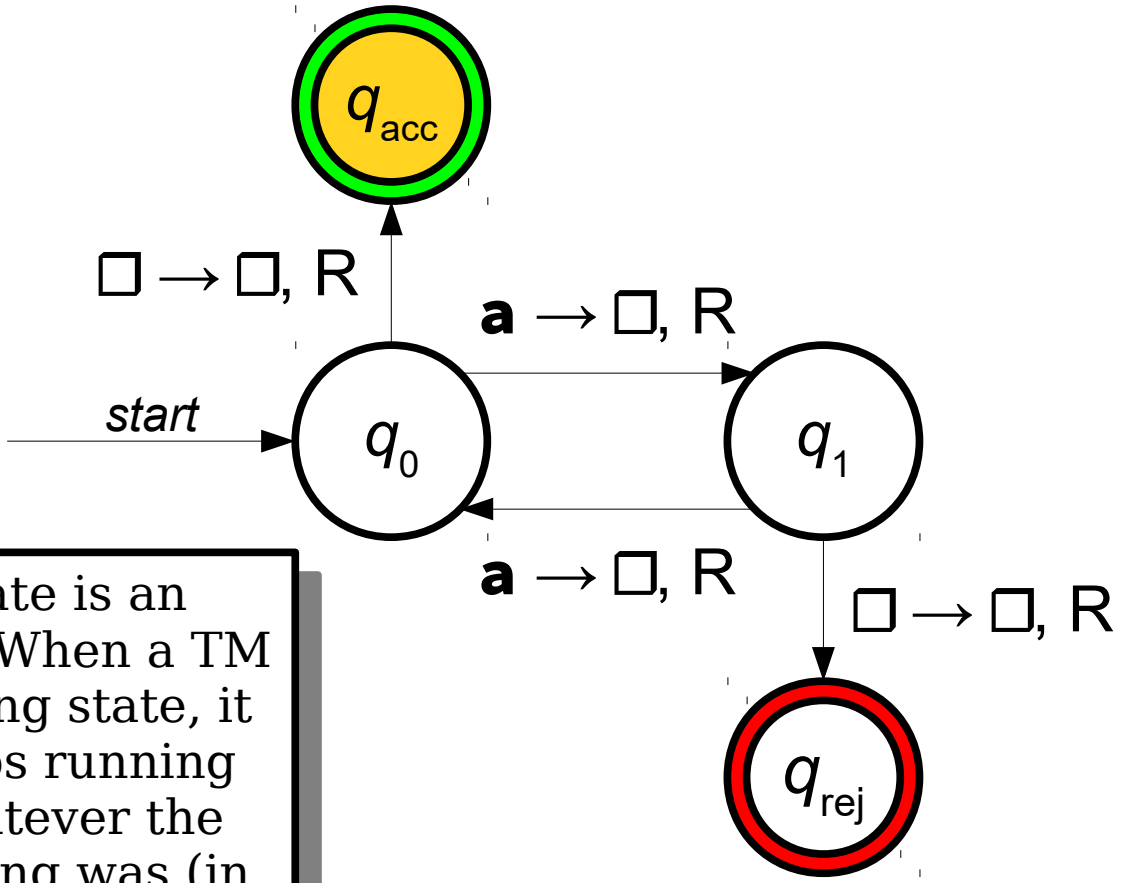

Unlike a DFA or NFA, a TM doesn't stop after reading all the input characters. We keep running until the machine explicitly says to stop.



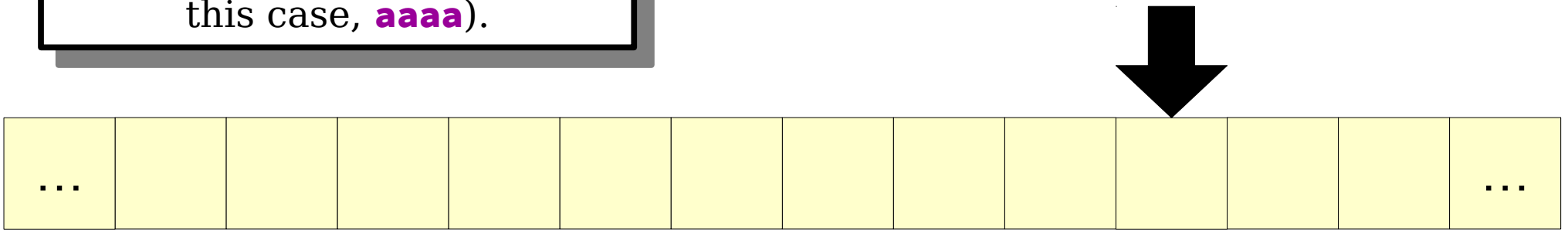
# A Simple Turing Machine



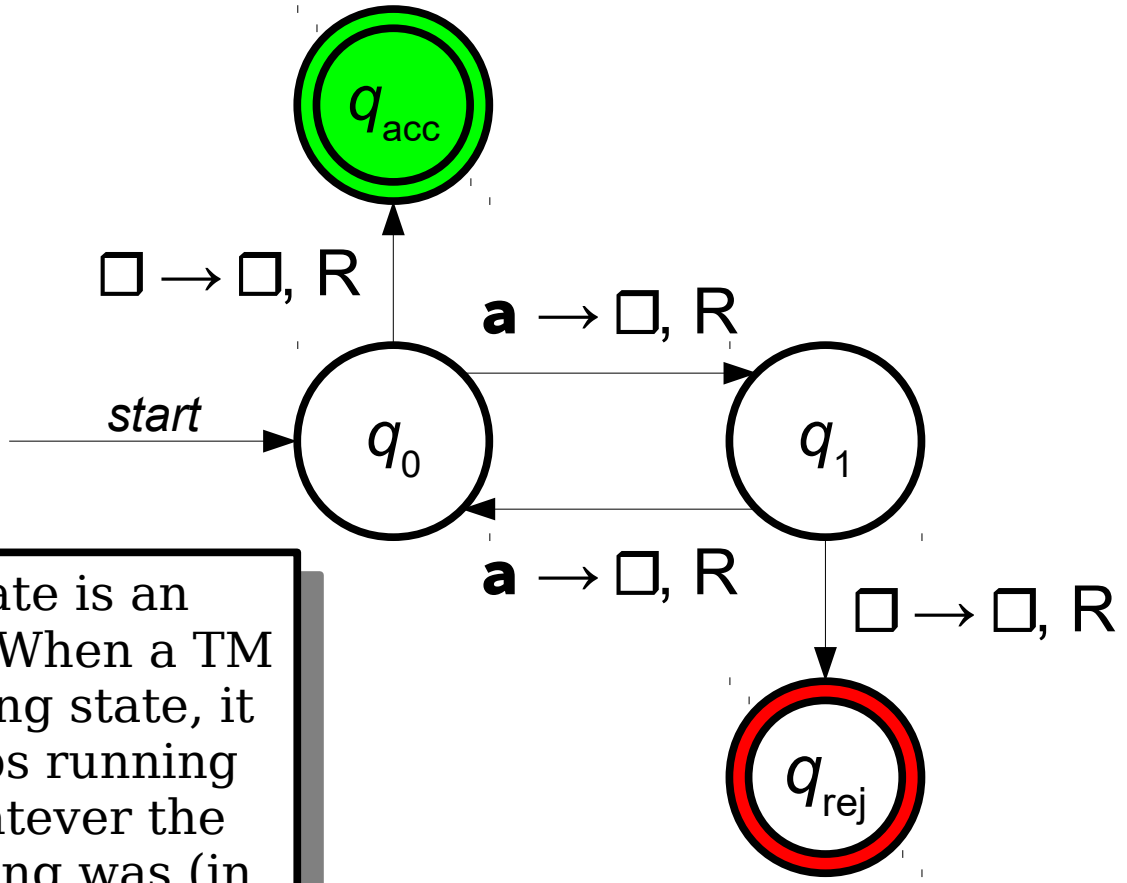
# A Simple Turing Machine



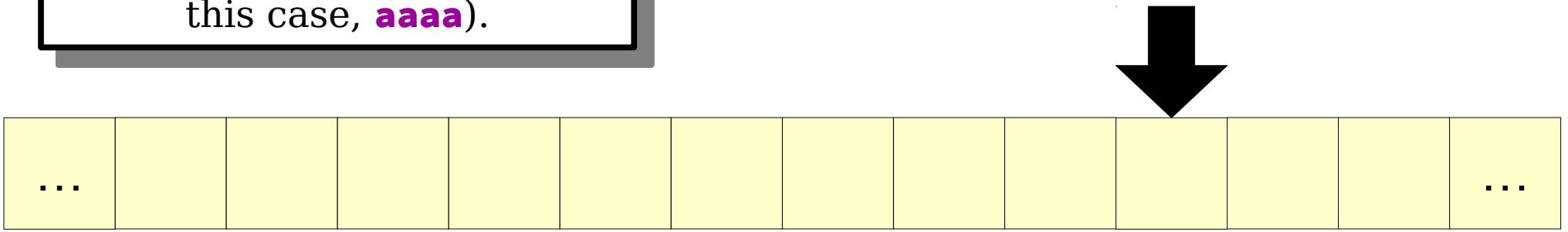

This special state is an ***accepting state***. When a TM enters an accepting state, it *immediately* stops running and accepts whatever the original input string was (in this case, **aaaa**).



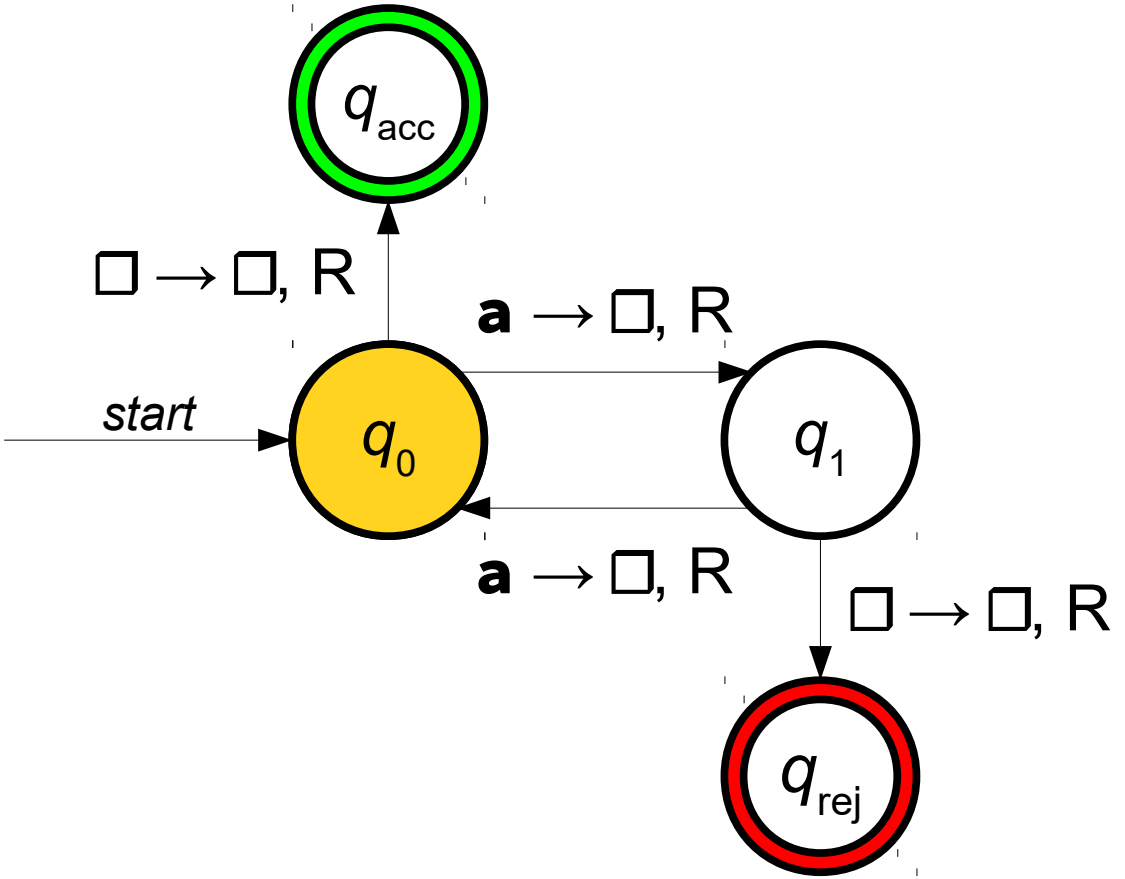
# A Simple Turing Machine

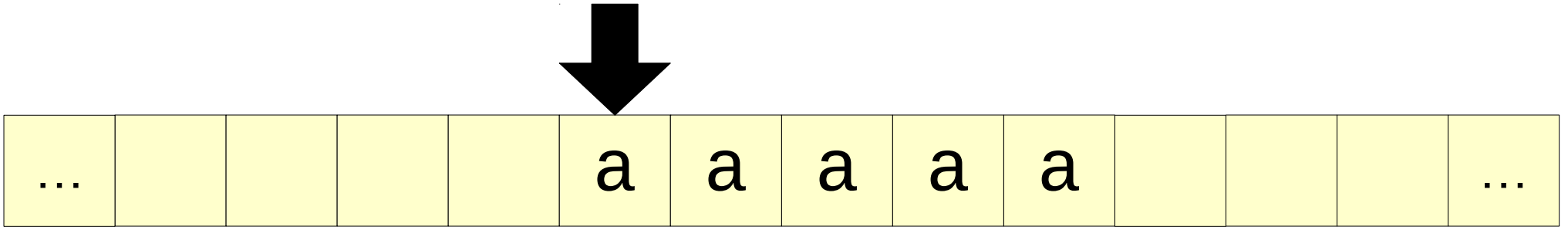



This special state is an ***accepting state***. When a TM enters an accepting state, it *immediately* stops running and accepts whatever the original input string was (in this case, **aaaa**).

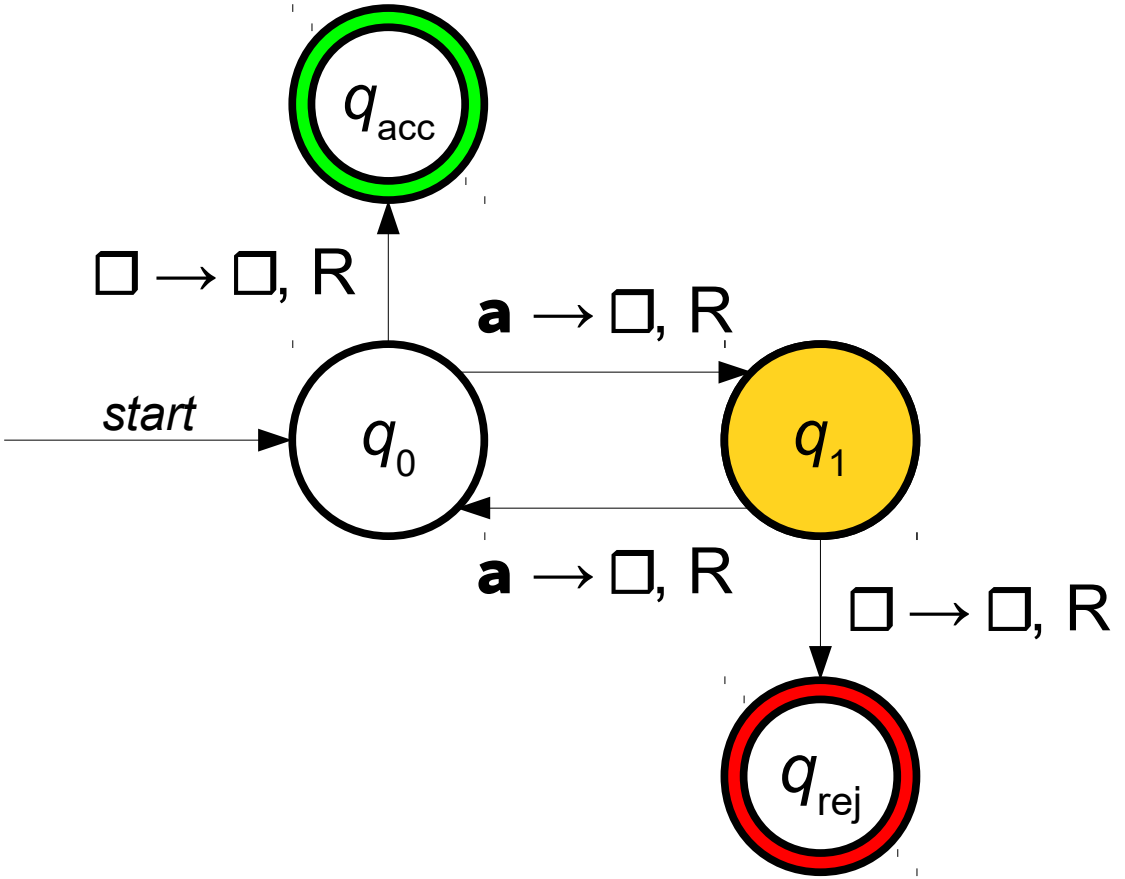


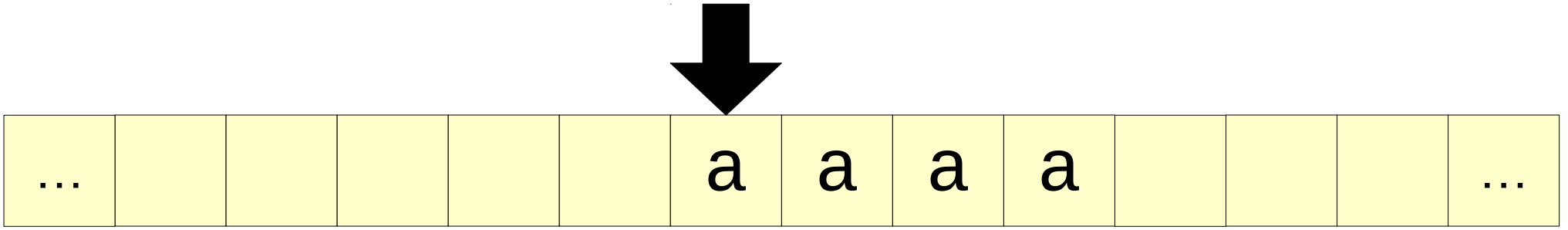
# A Simple Turing Machine



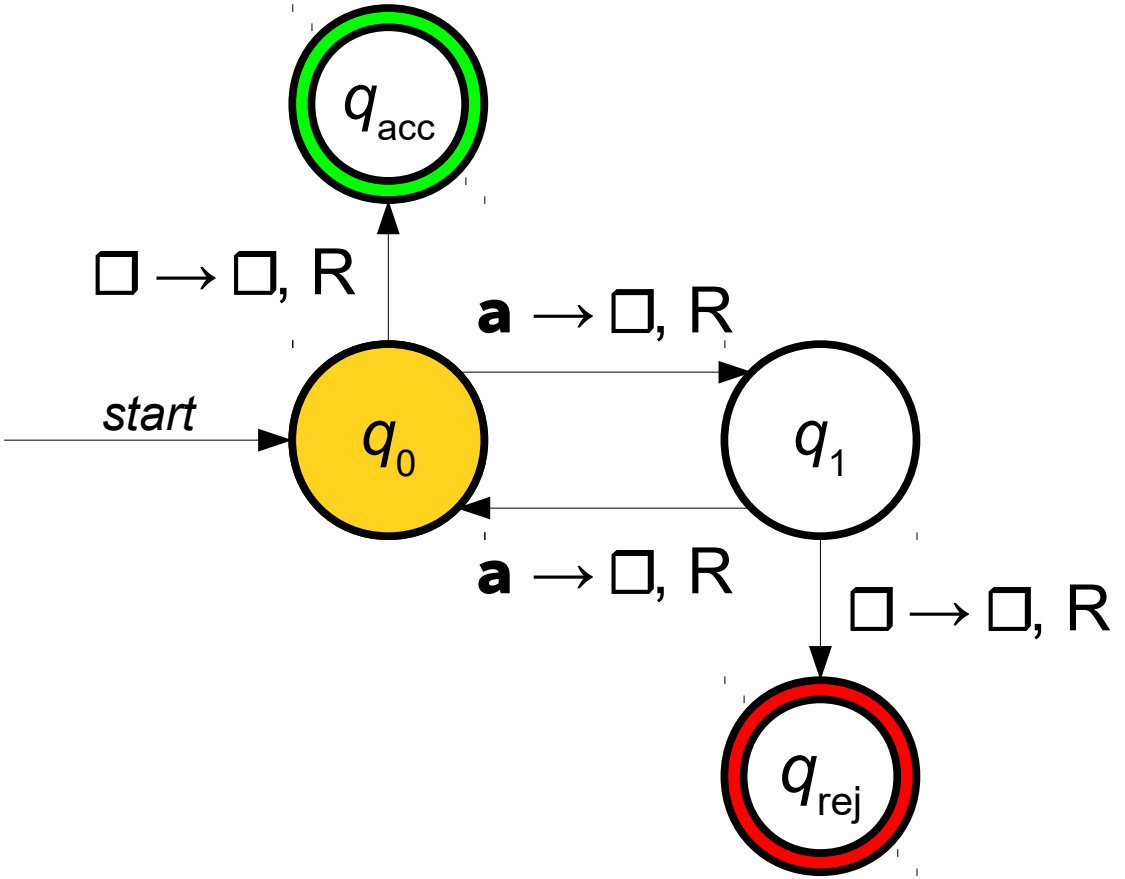



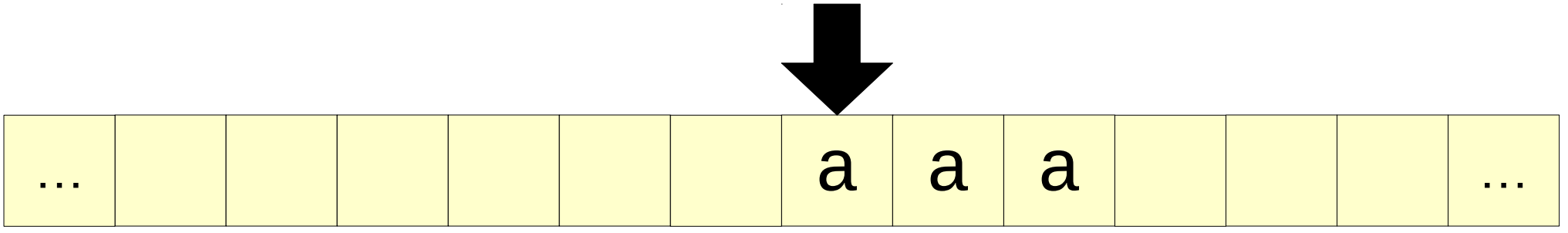
# A Simple Turing Machine



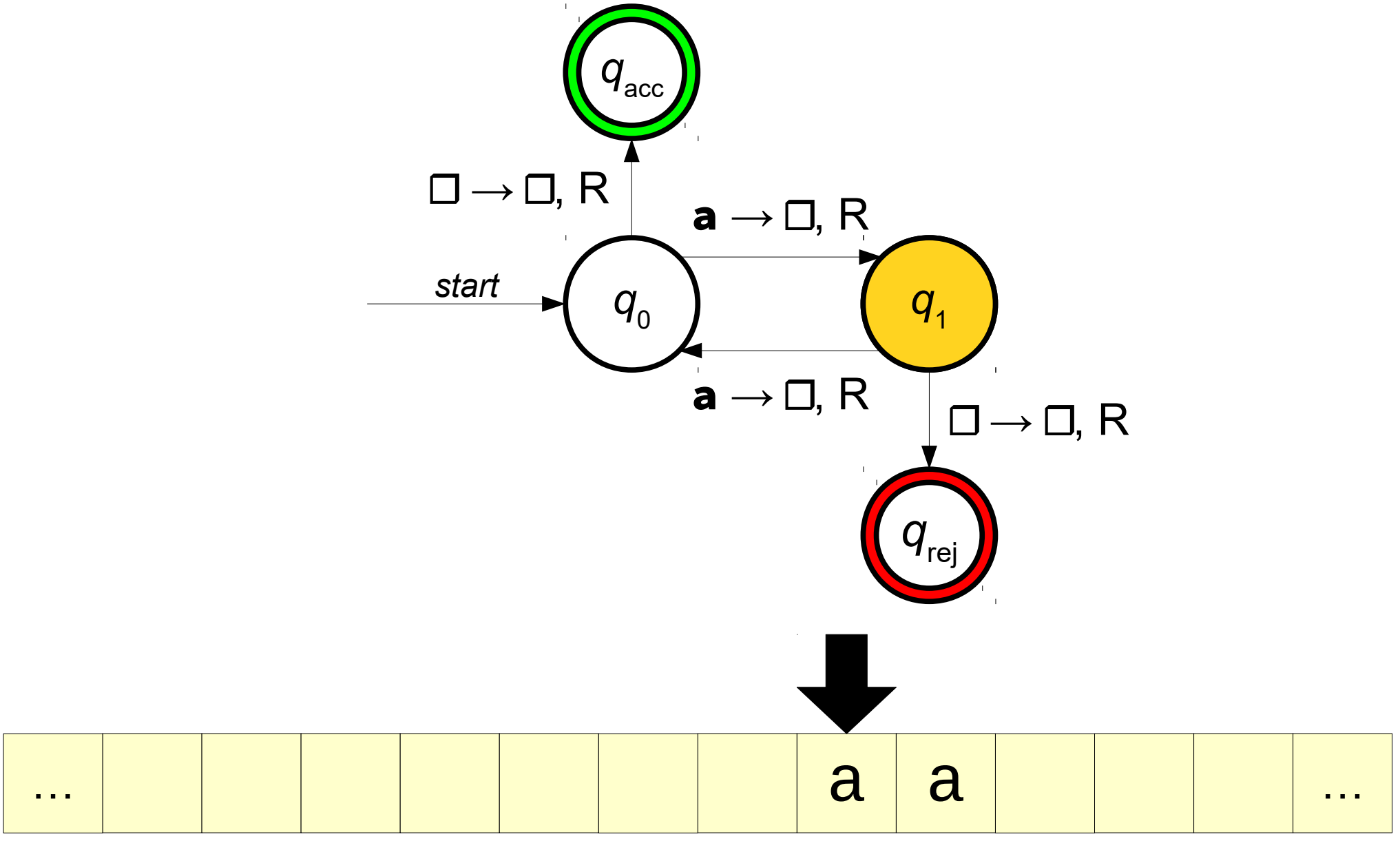



# A Simple Turing Machine

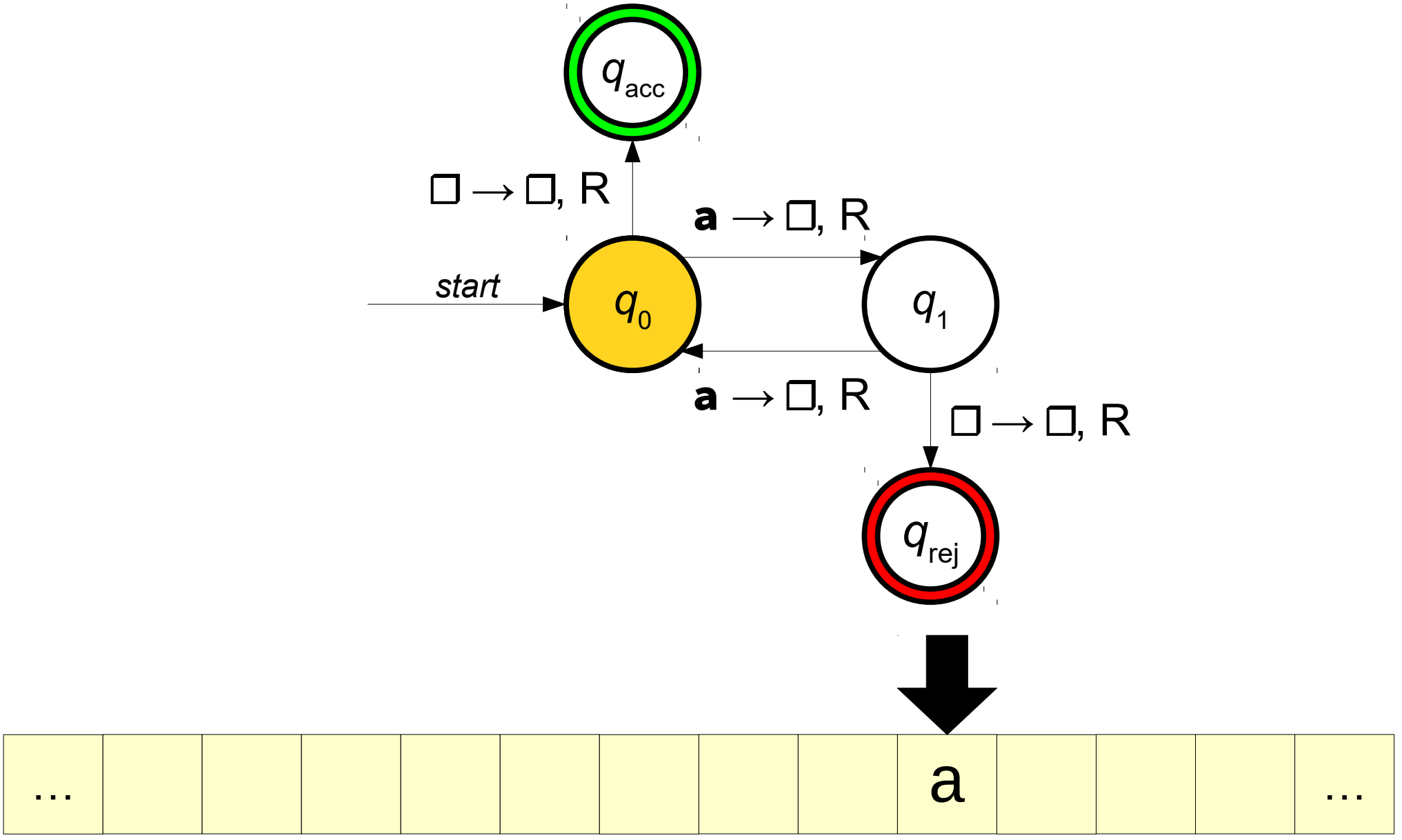


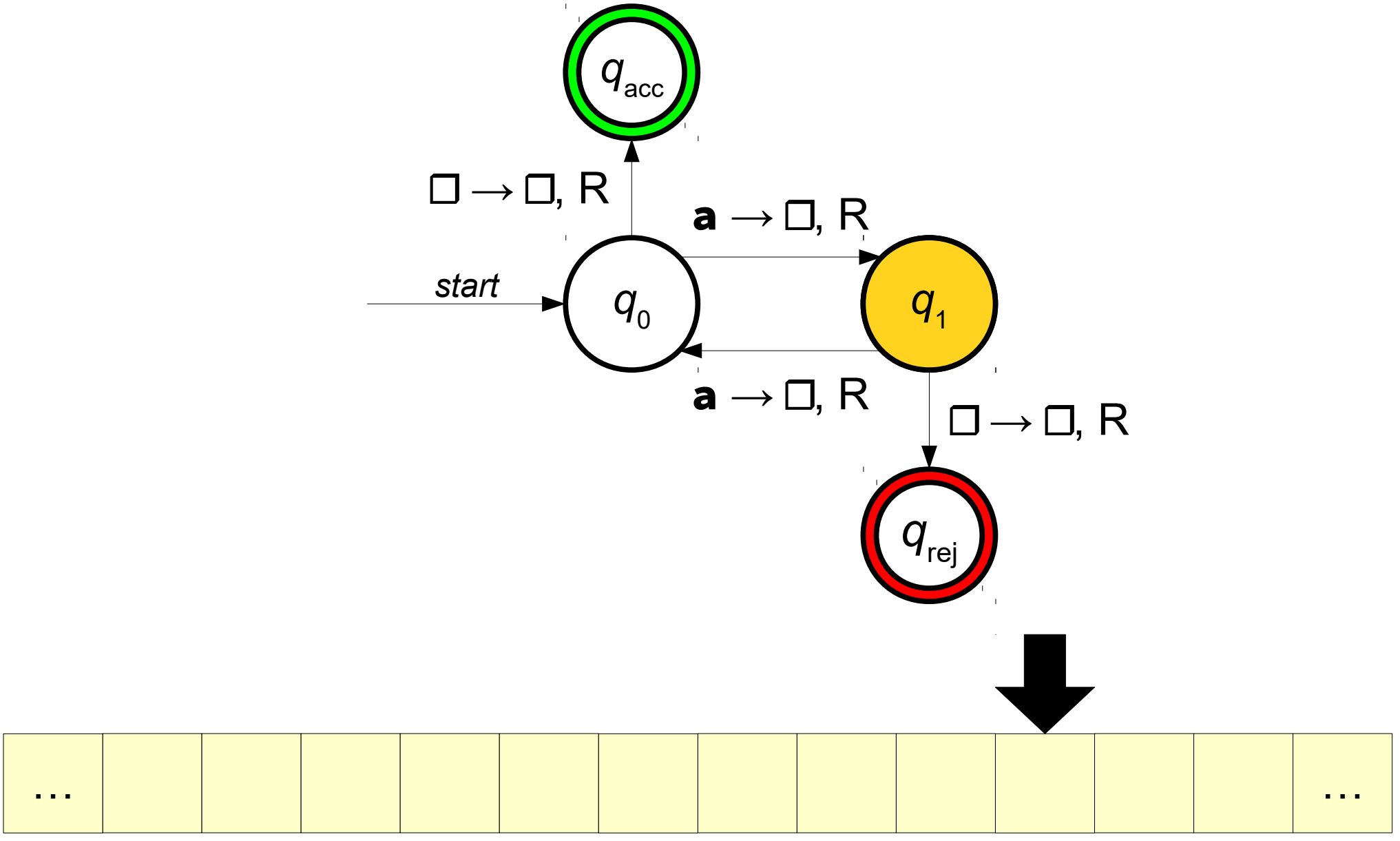
# A Simple Turing Machine



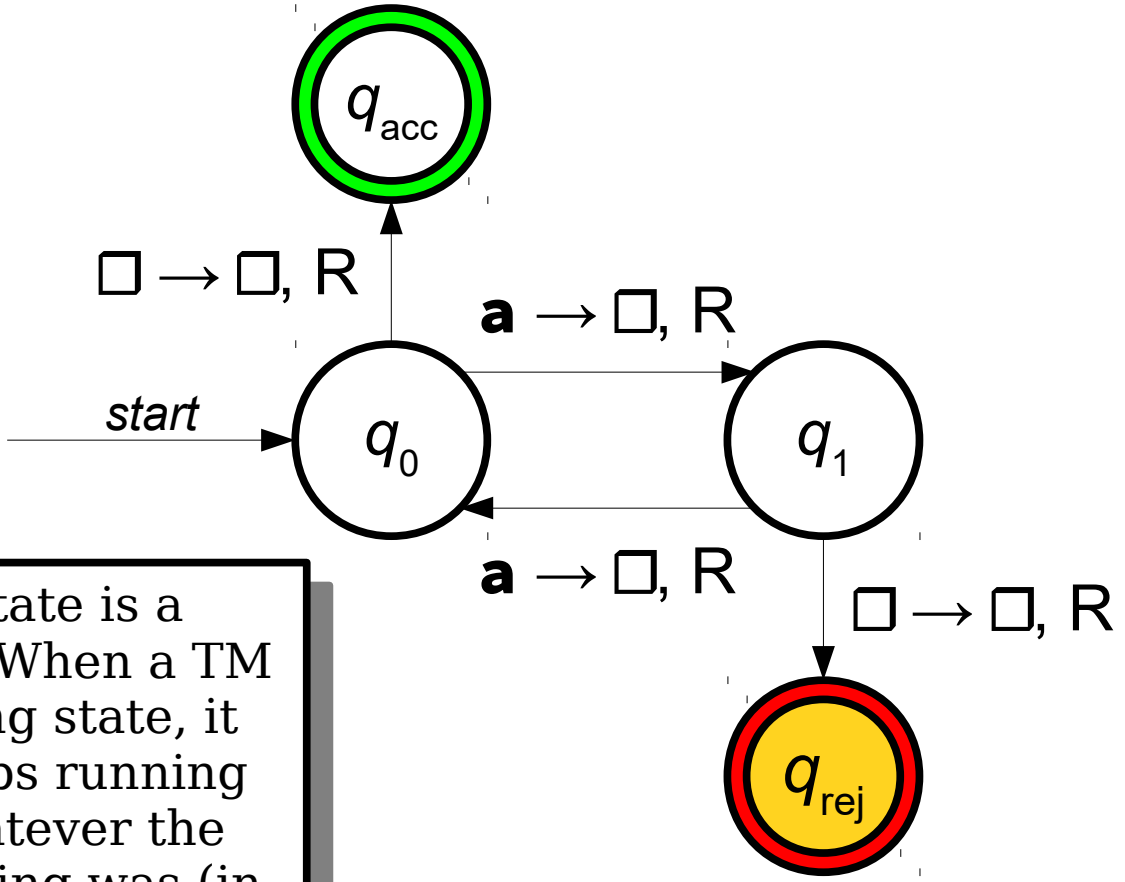
# A Simple Turing Machine



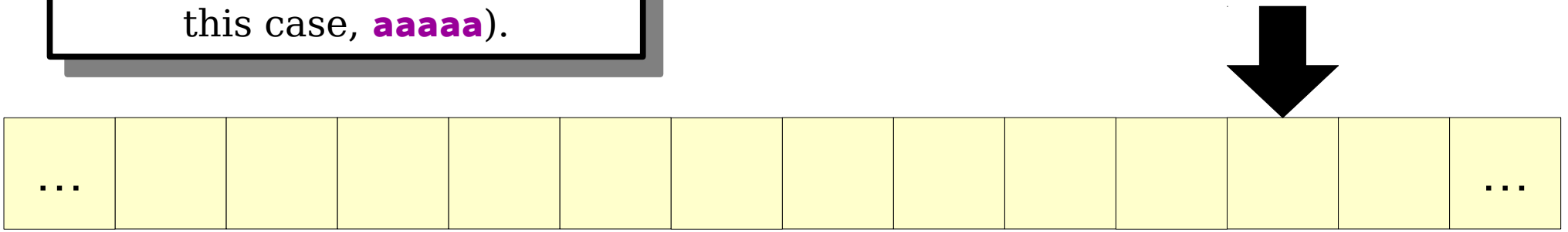
# A Simple Turing Machine



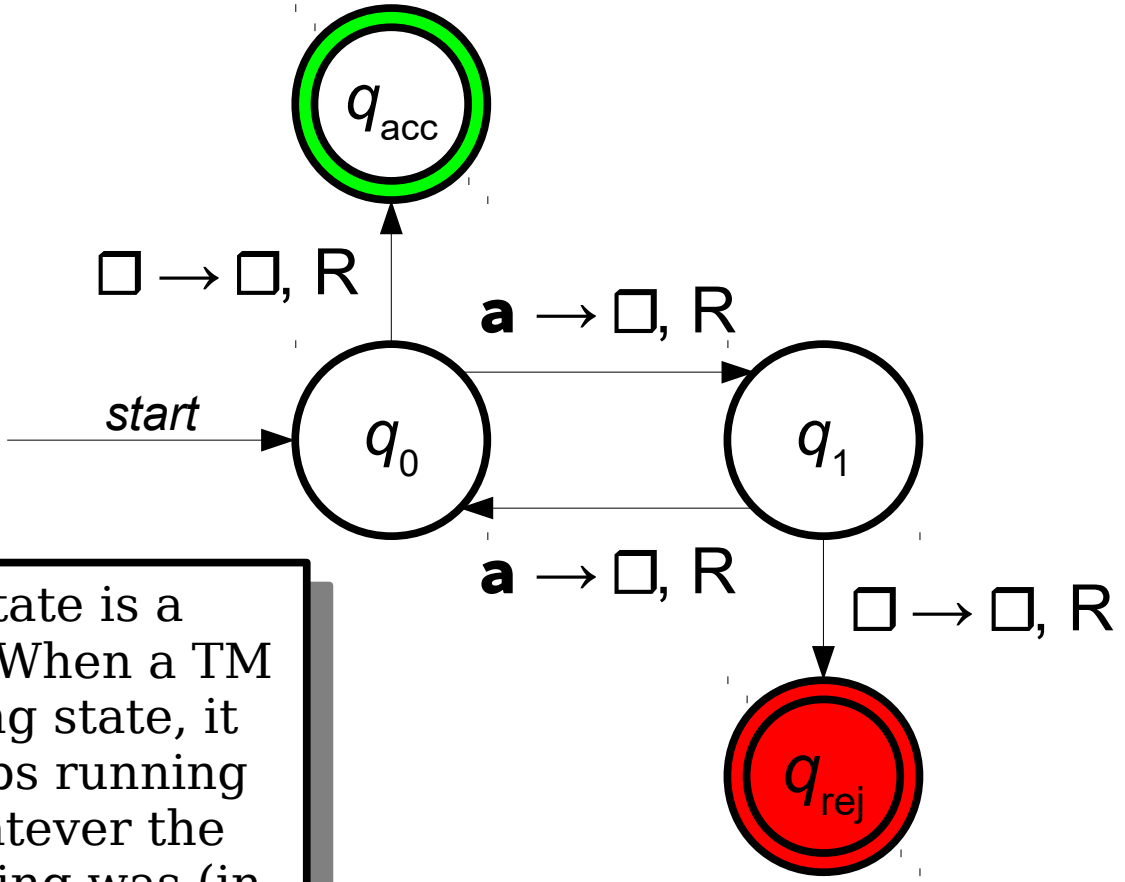
# A Simple Turing Machine



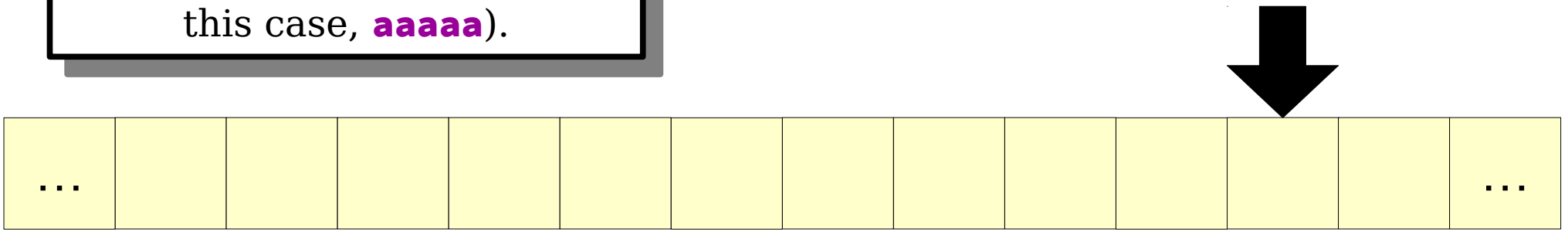

This special state is a ***rejecting state***. When a TM enters a rejecting state, it *immediately* stops running and rejects whatever the original input string was (in this case, **aaaaa**).



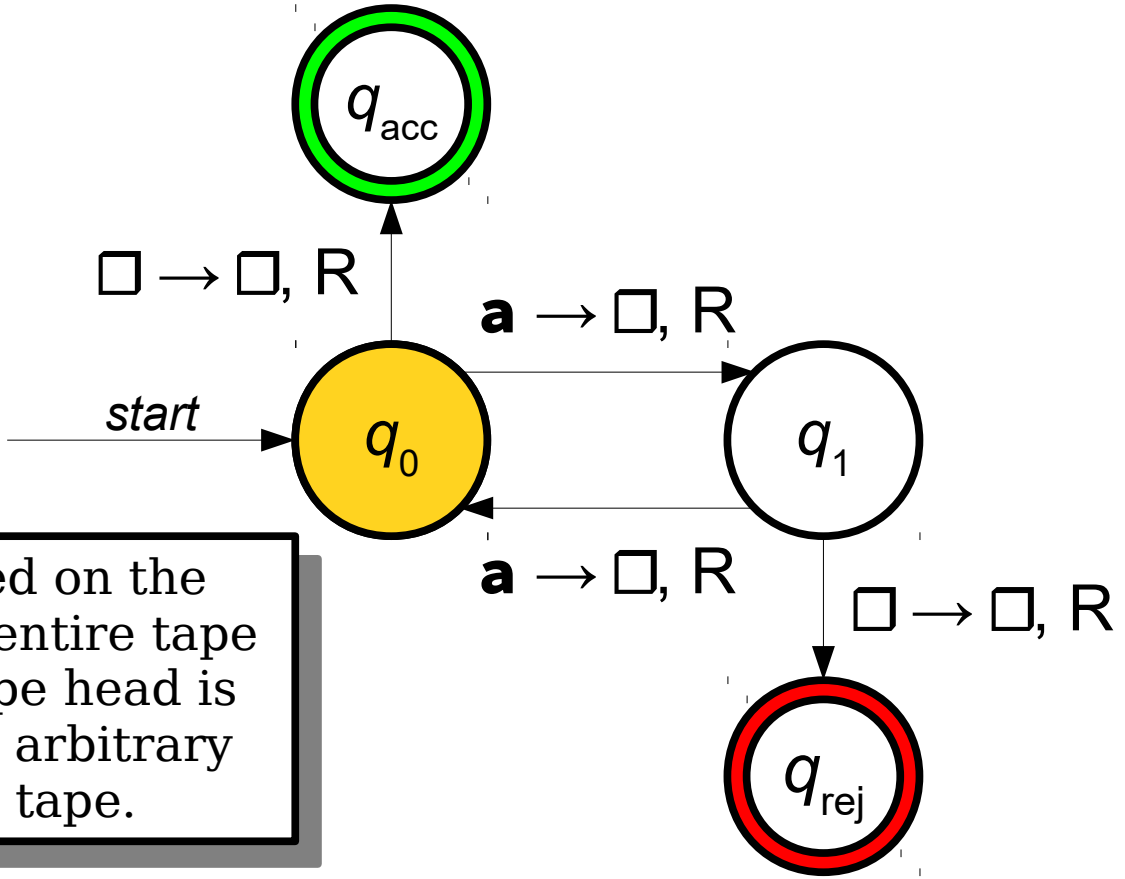
# A Simple Turing Machine



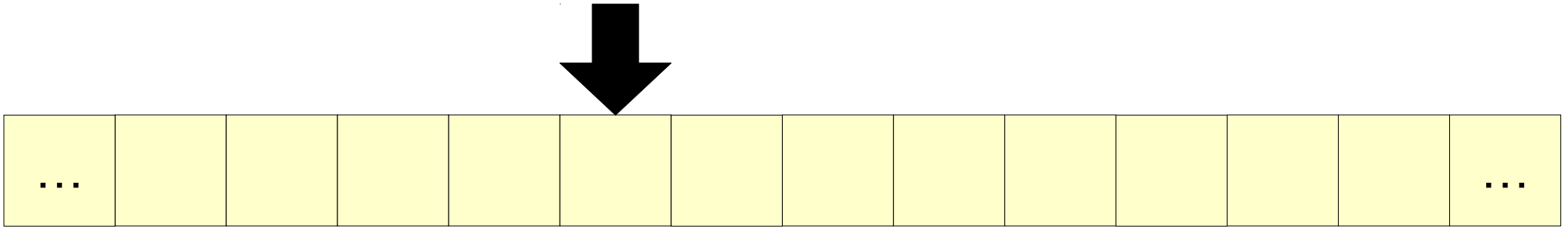

This special state is a ***rejecting state***. When a TM enters a rejecting state, it *immediately* stops running and rejects whatever the original input string was (in this case, **aaaaa**).



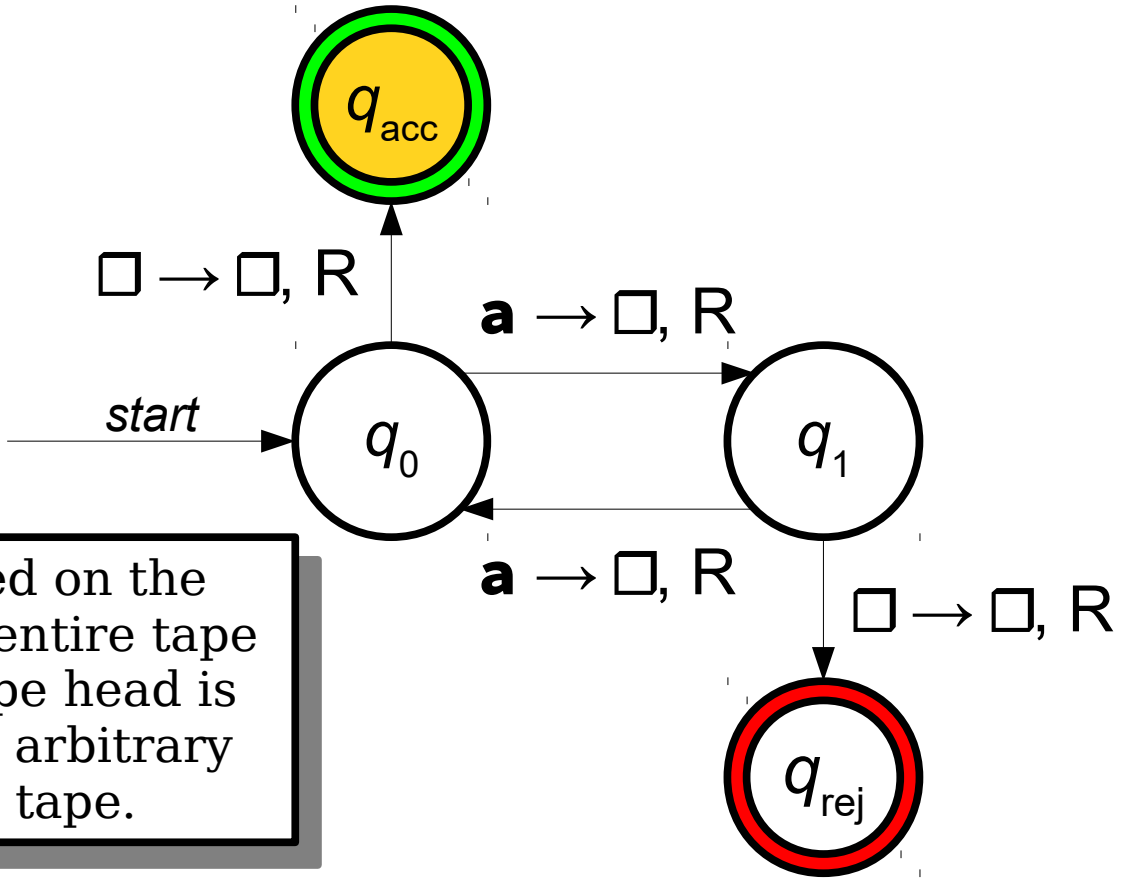
# A Simple Turing Machine



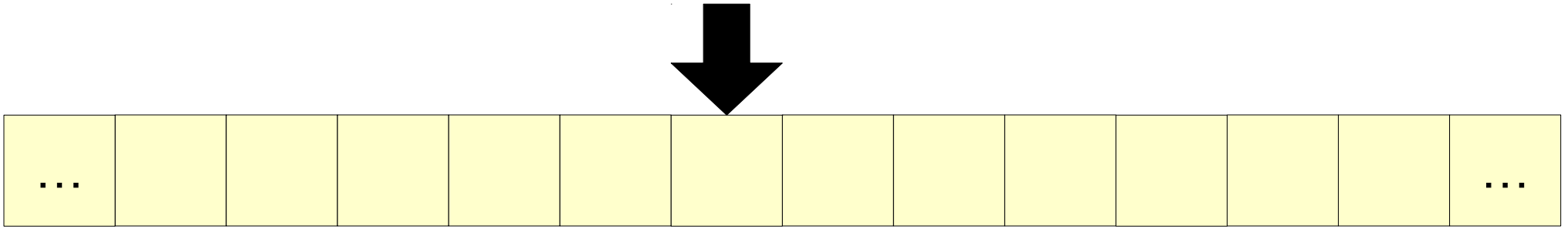

If the TM is started on the empty string ε, the entire tape is blank and the tape head is positioned at some arbitrary location on the tape.



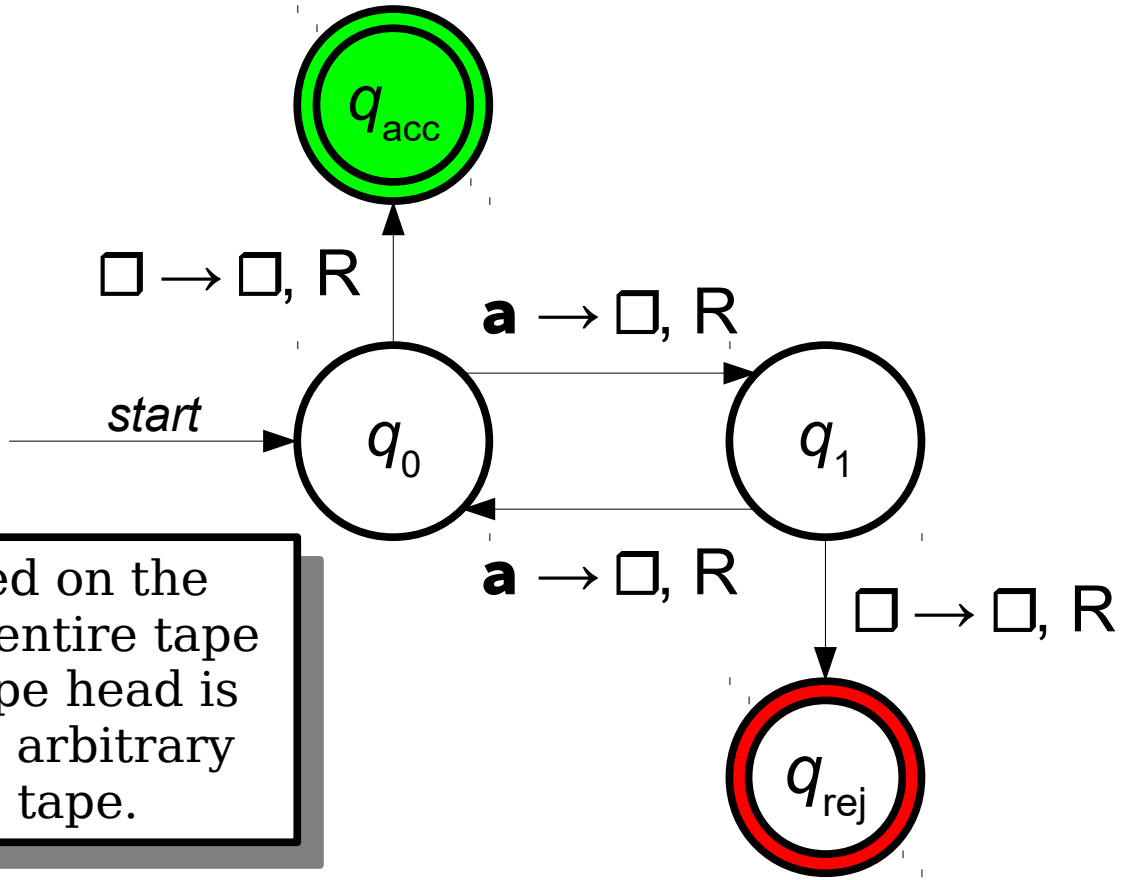
# A Simple Turing Machine



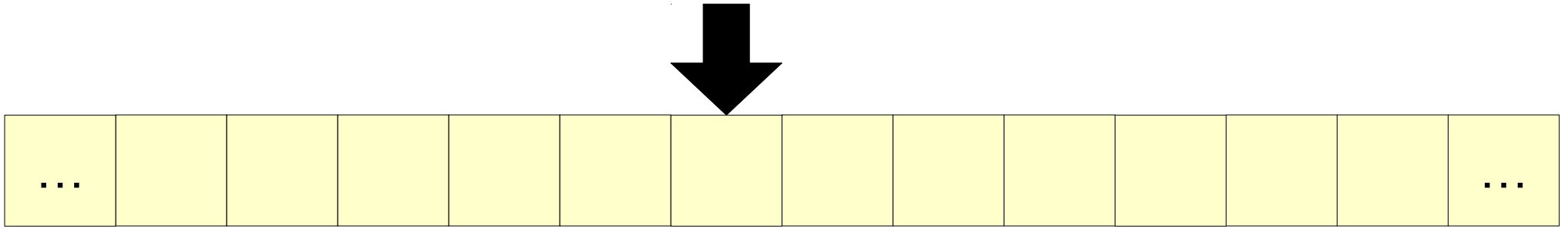

If the TM is started on the empty string ε, the entire tape is blank and the tape head is positioned at some arbitrary location on the tape.



# A Simple Turing Machine




If the TM is started on the empty string ε, the entire tape is blank and the tape head is positioned at some arbitrary location on the tape.



# The Turing Machine

- A Turing machine consists of three parts:
  - A ***finite-state control*** that issues commands,
  - an ***infinite tape*** for input and scratch space, and
  - a ***tape head*** that can read and write a single tape cell.
- At each step, the Turing machine
  - writes a symbol to the tape cell under the tape head, *[Q: what if you don't want to write?]*
  - changes state, and
  - moves the tape head to the left or to the right.

# Input and Tape Alphabets

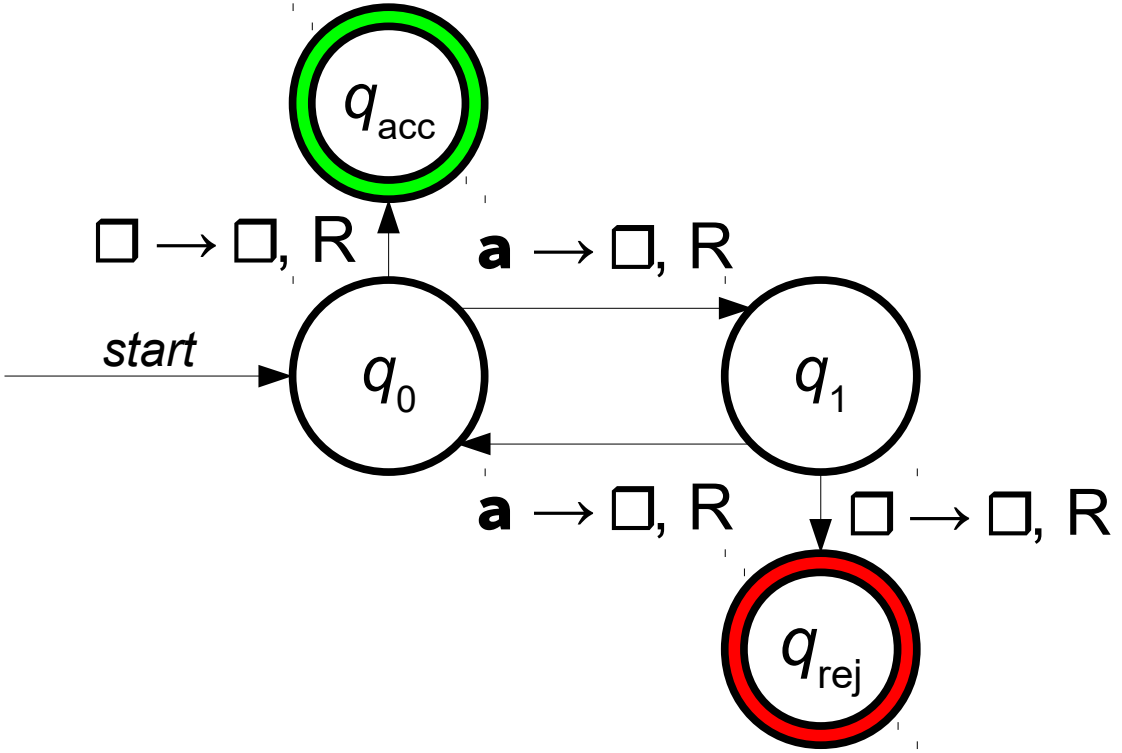
- A Turing machine has two alphabets:
  - An ***input alphabet*** Σ. All input strings are written in the input alphabet.
  - A ***tape alphabet*** Γ, where Σ ⊊ Γ. The tape alphabet contains all symbols that can be written onto the tape.
- The tape alphabet Γ can contain any number of symbols, but always contains at least one ***blank symbol***, denoted ❑. You are guaranteed ❑ ∉ Σ.
- At startup, the Turing machine begins with an infinite tape of ❑ symbols with the input written at some location. The tape head is positioned at the start of the input.

# Accepting and Rejecting States

- Unlike DFAs, Turing machines do not stop processing the input when they finish reading it.
- Turing machines decide when (and if!) they will accept or reject their input.
- Turing machines can enter infinite loops and never accept or reject; more on that later...

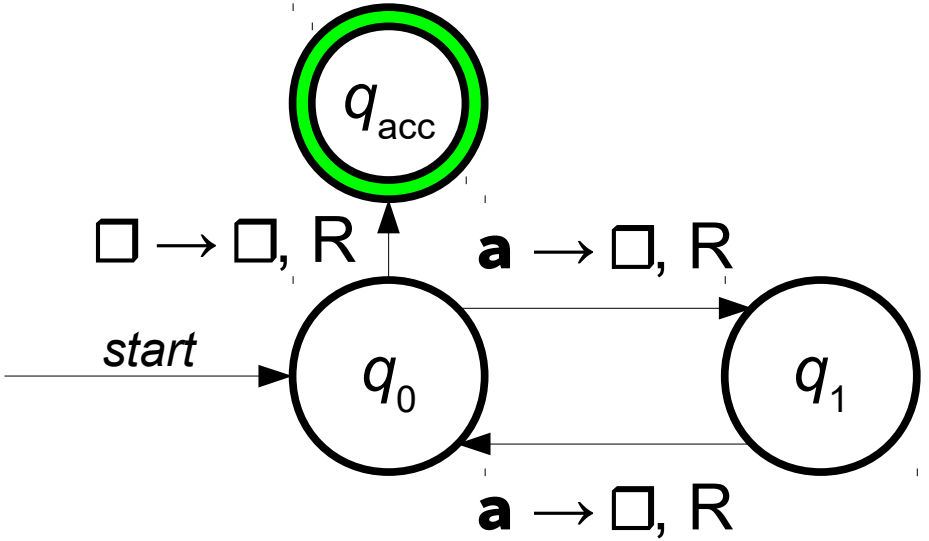
# Determinism

- Turing machines are ***deterministic***: for every combination of a (non-accepting, non-rejecting) state $q$ and a tape symbol $a \in \Gamma$, there must be exactly one transition defined for that combination of $q$ and $a$.
- Any transitions that are missing implicitly go straight to a rejecting state. We'll use this later to simplify our designs.

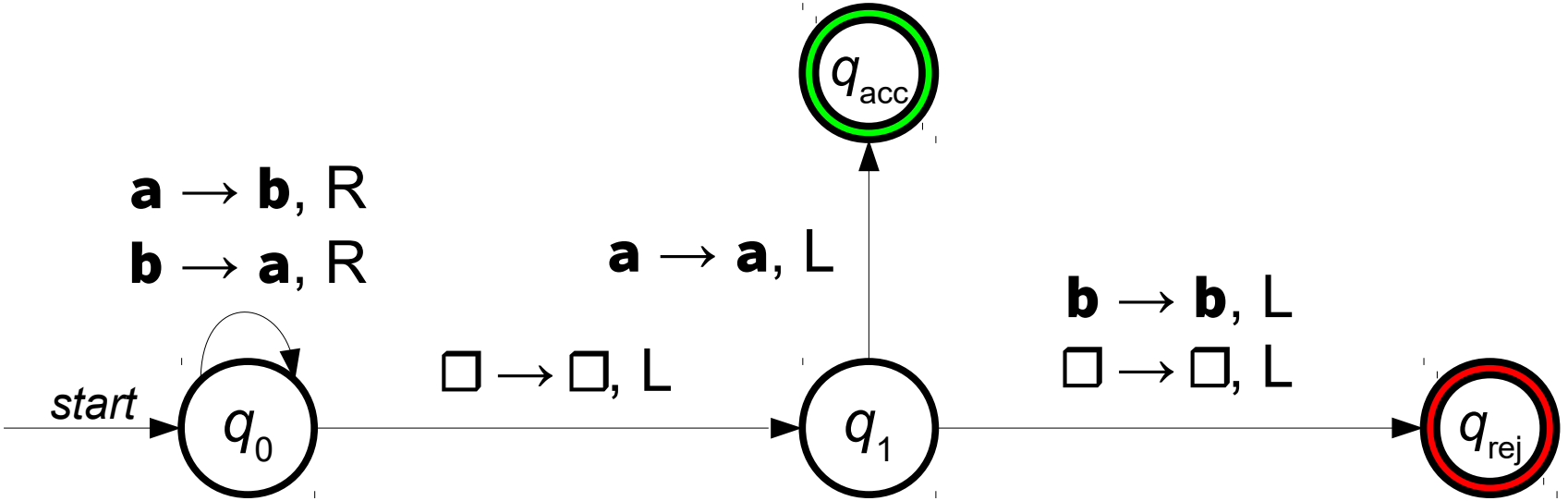


# Determinism

- Turing machines are ***deterministic***: for every combination of a (non-accepting, non-rejecting) state $q$ and a tape symbol $a \in \Gamma$, there must be exactly one transition defined for that combination of $q$ and $a$.
- Any transitions that are missing implicitly go straight to a rejecting state. We'll use this later to simplify our designs.

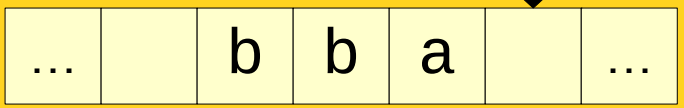



This machine is exactly the same as the previous one.

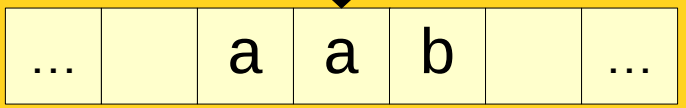



Run the TM shown above on the input string **bba**.
What will the tape look like when the TM finishes running?

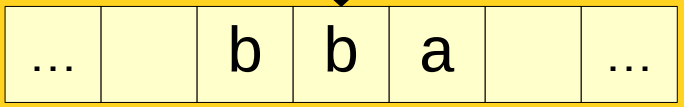
***A***.



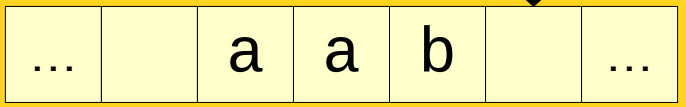

***B***.




***C***.

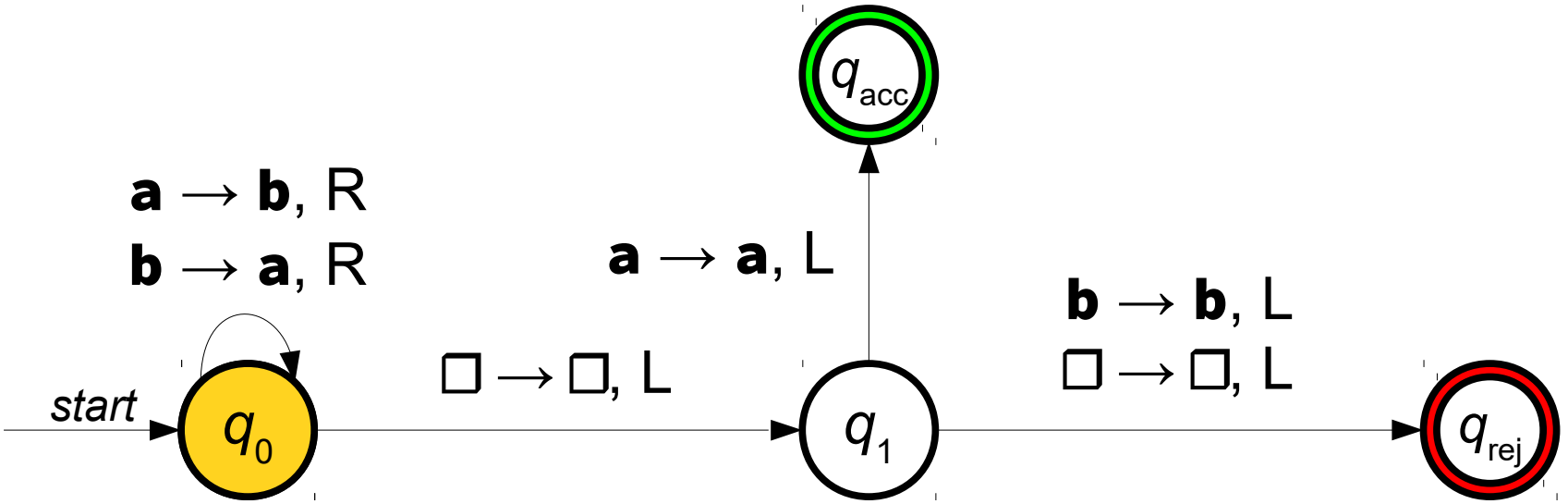


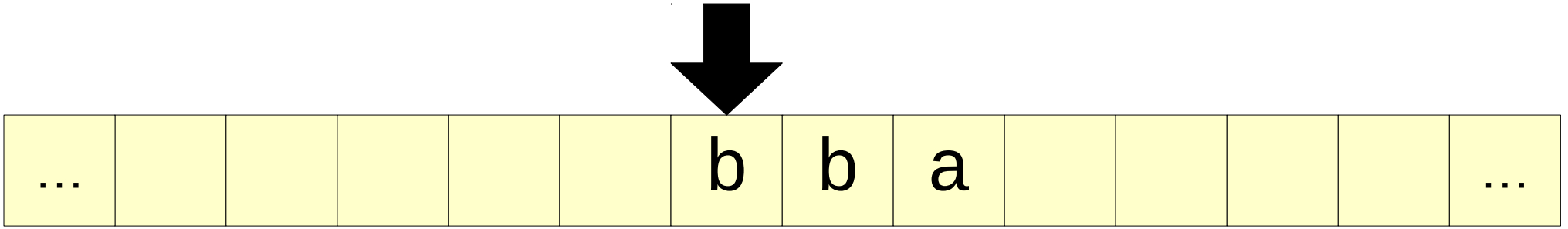

***D***.

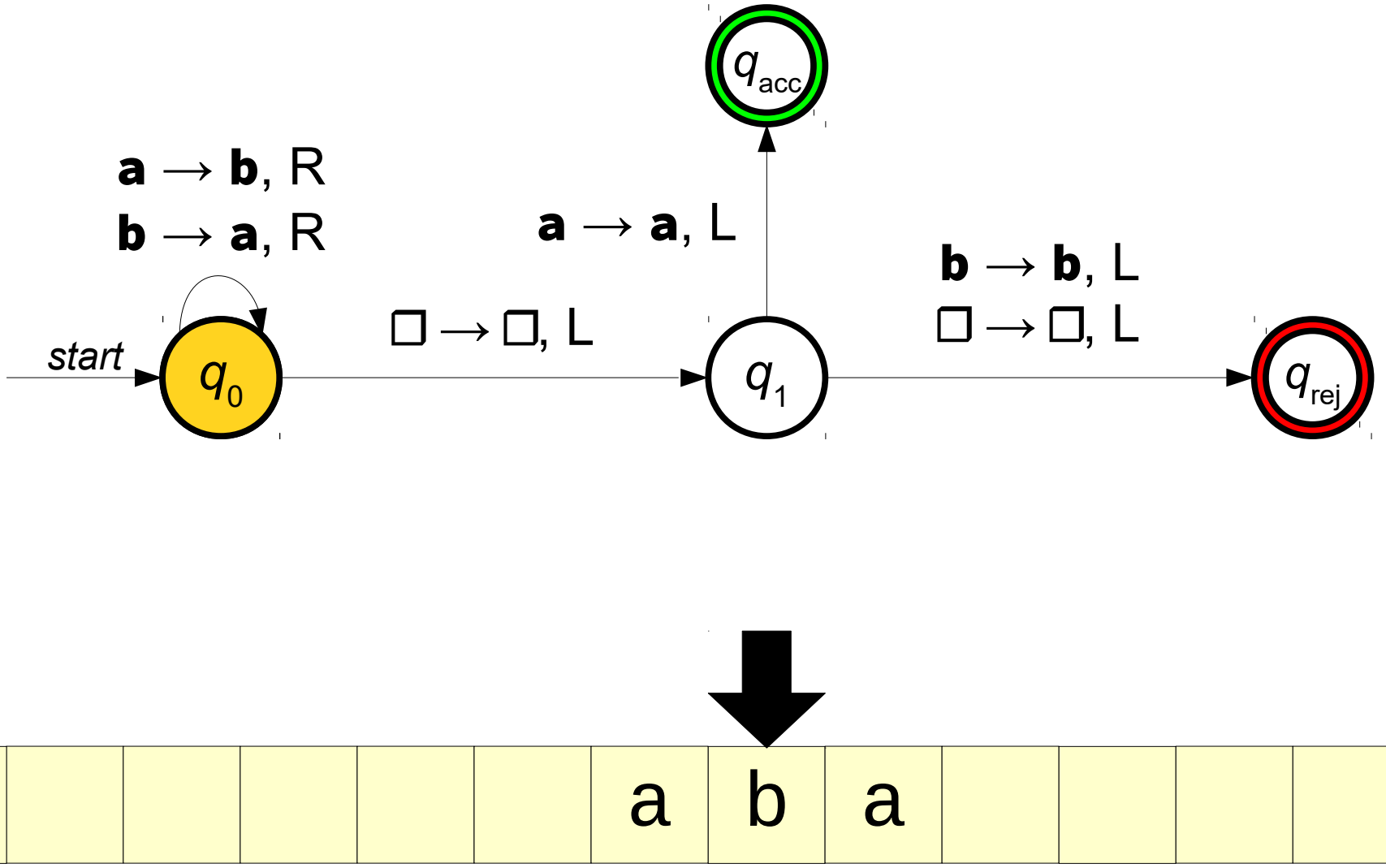


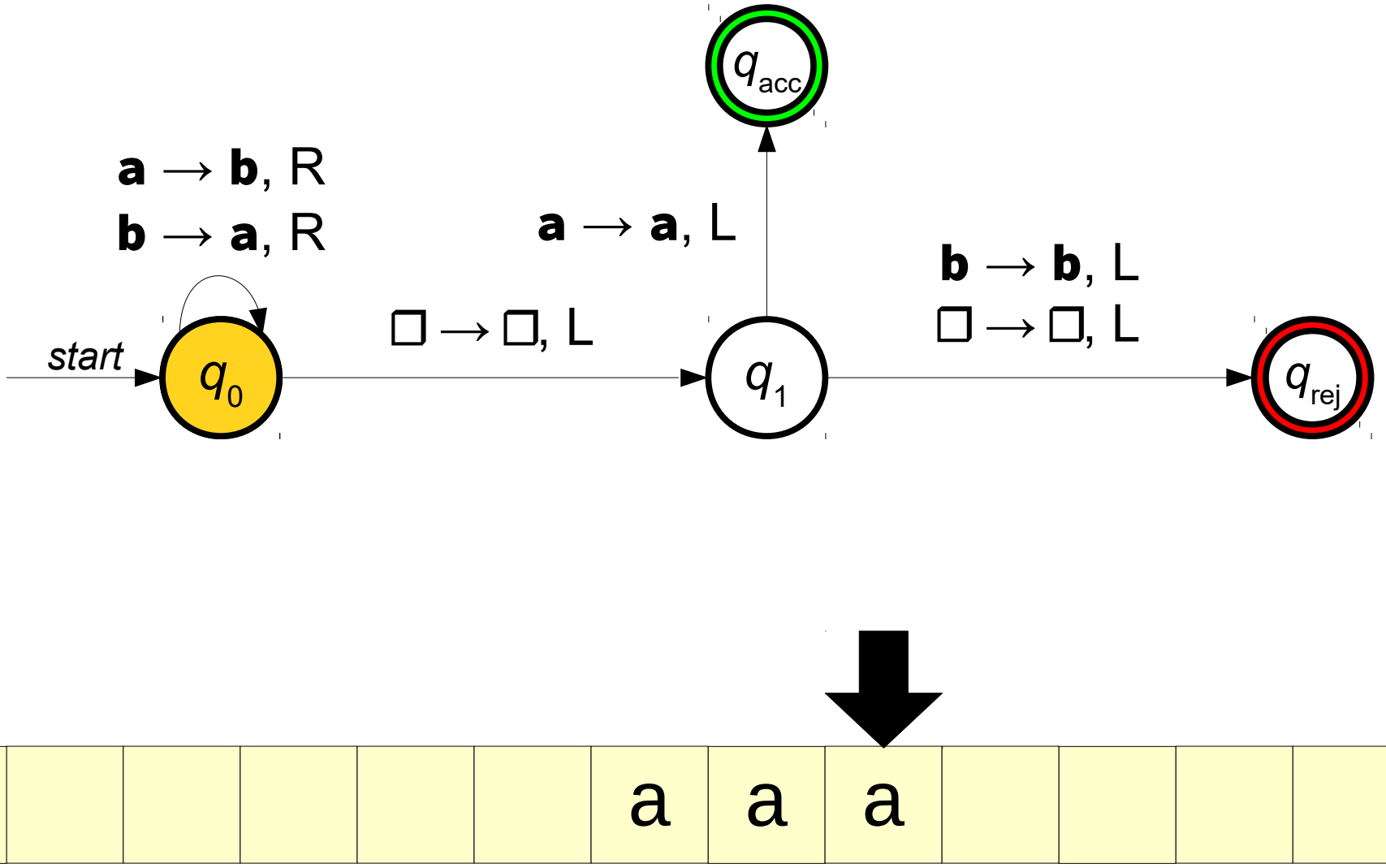

***E***. None of these, or two or more of these.

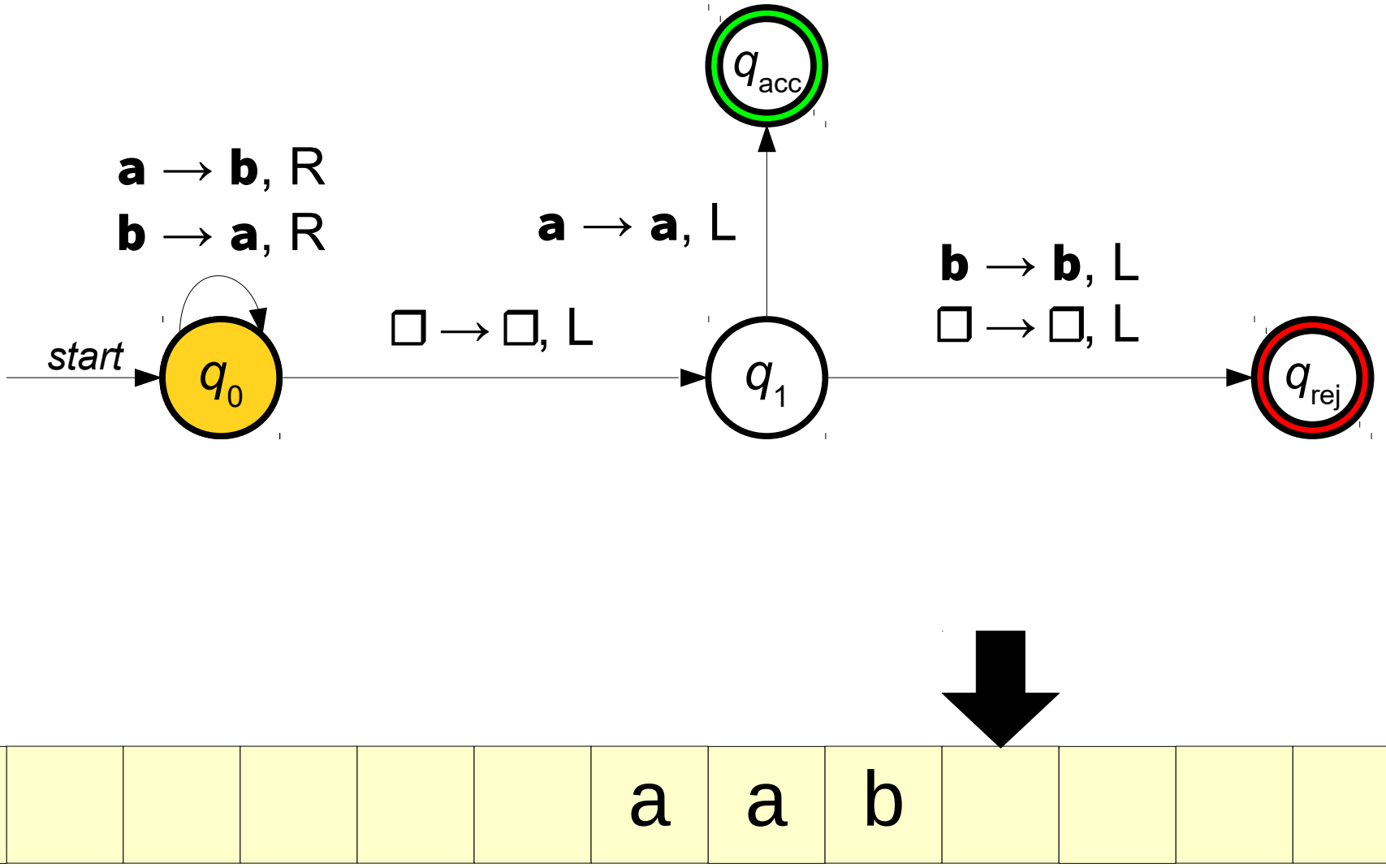
Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then ***A***, ***B***, ***C***, ***D***, or ***E***.

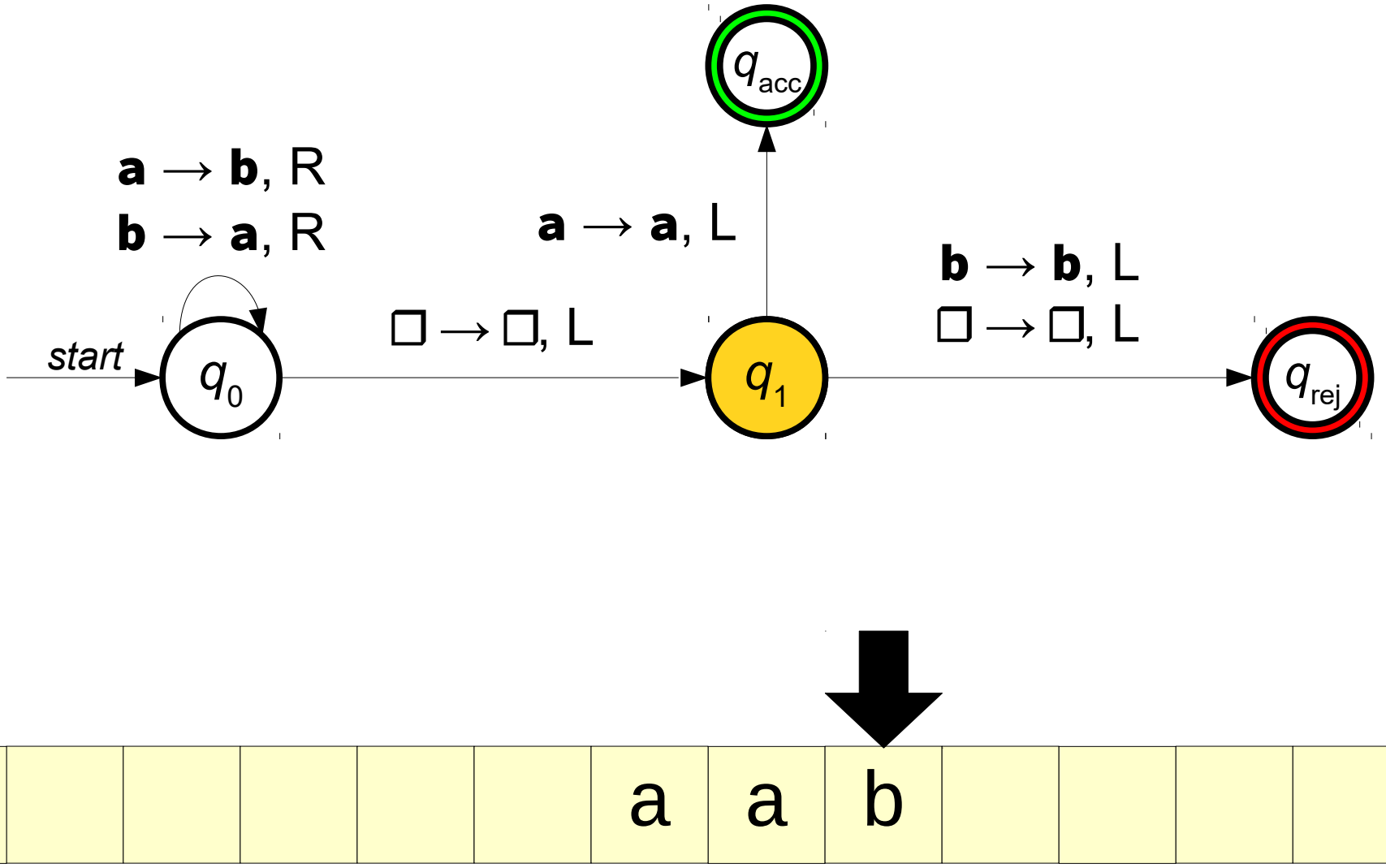

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$


...
b
b
a
...

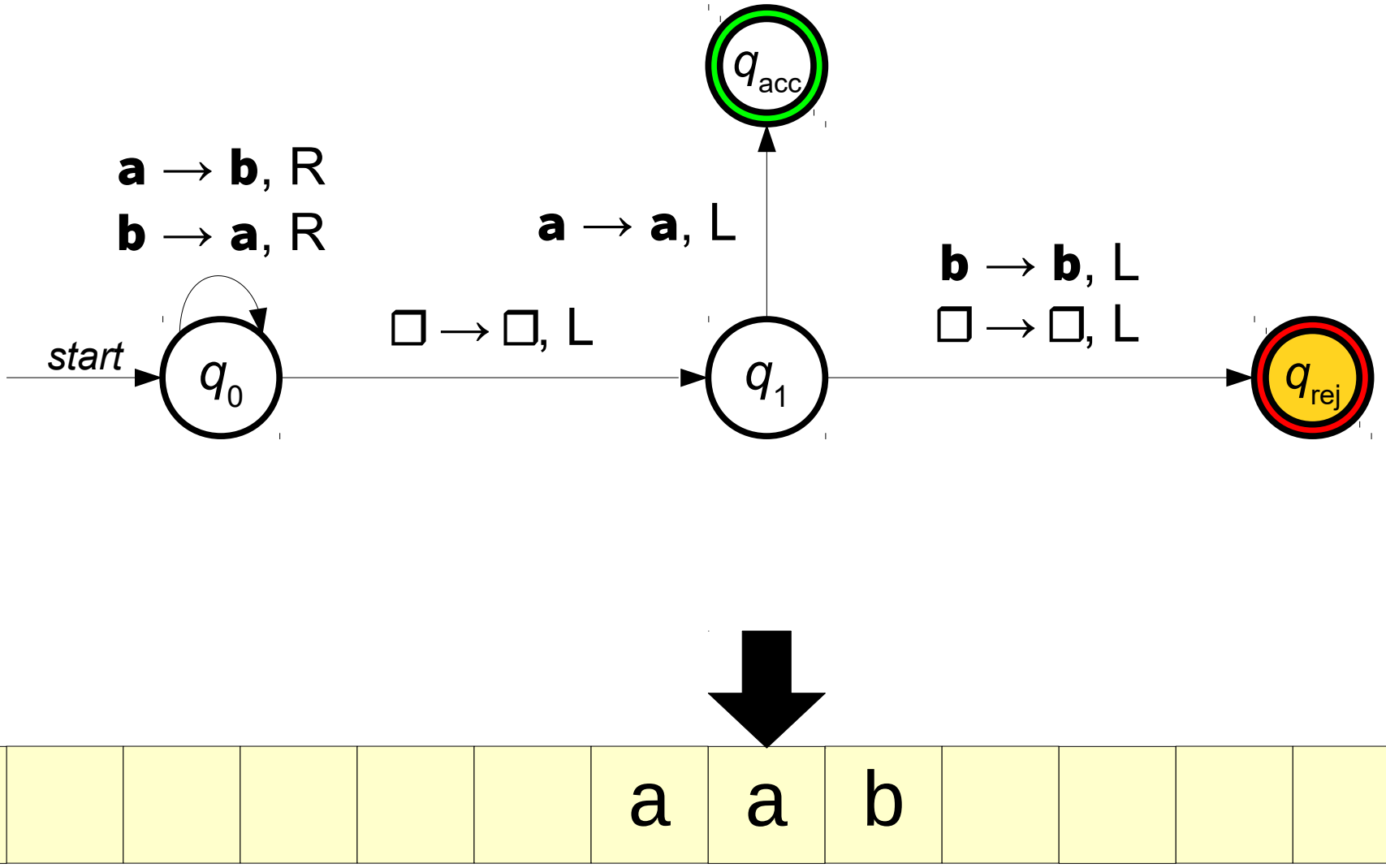

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
b
a
...

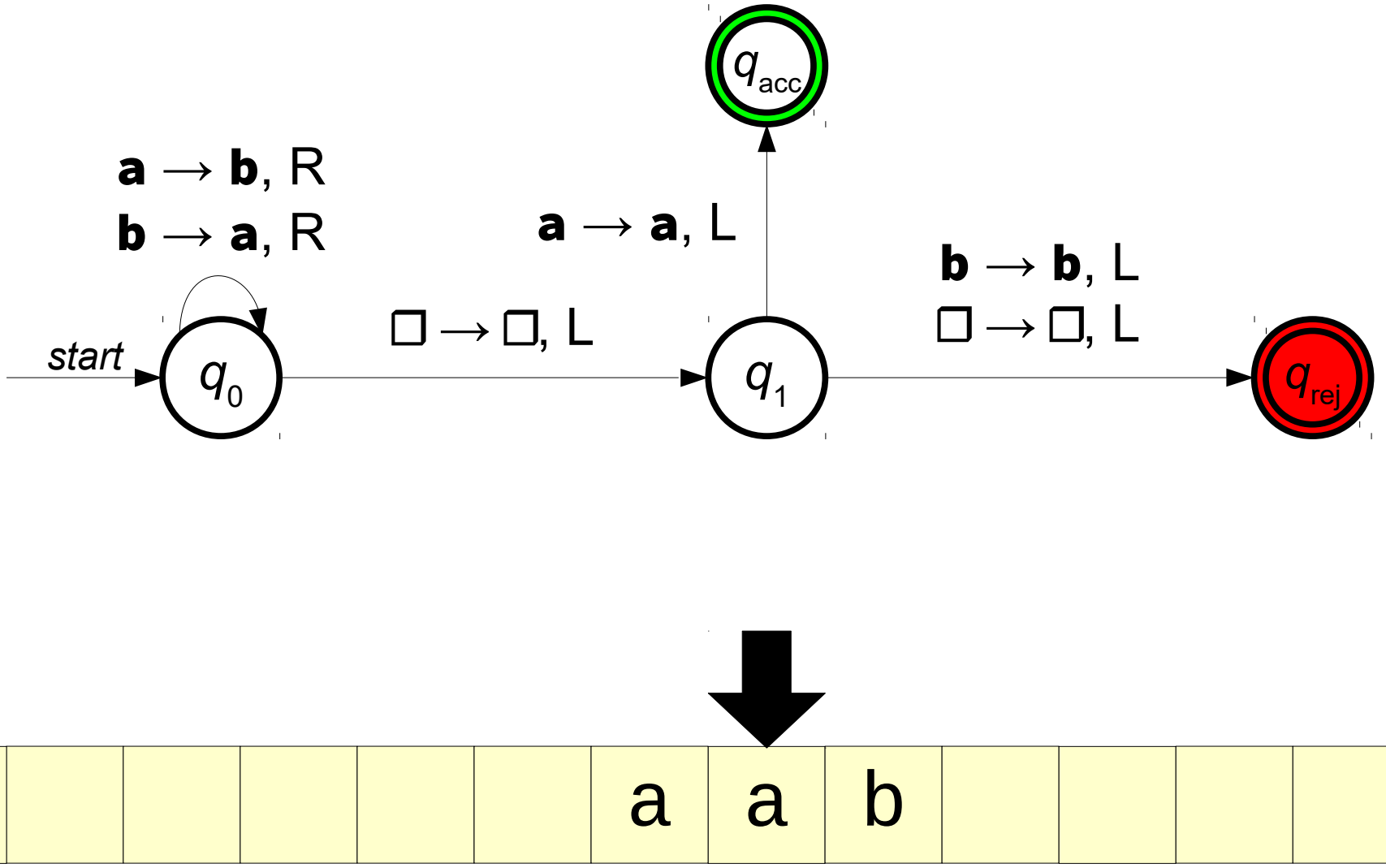

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
a
a
...

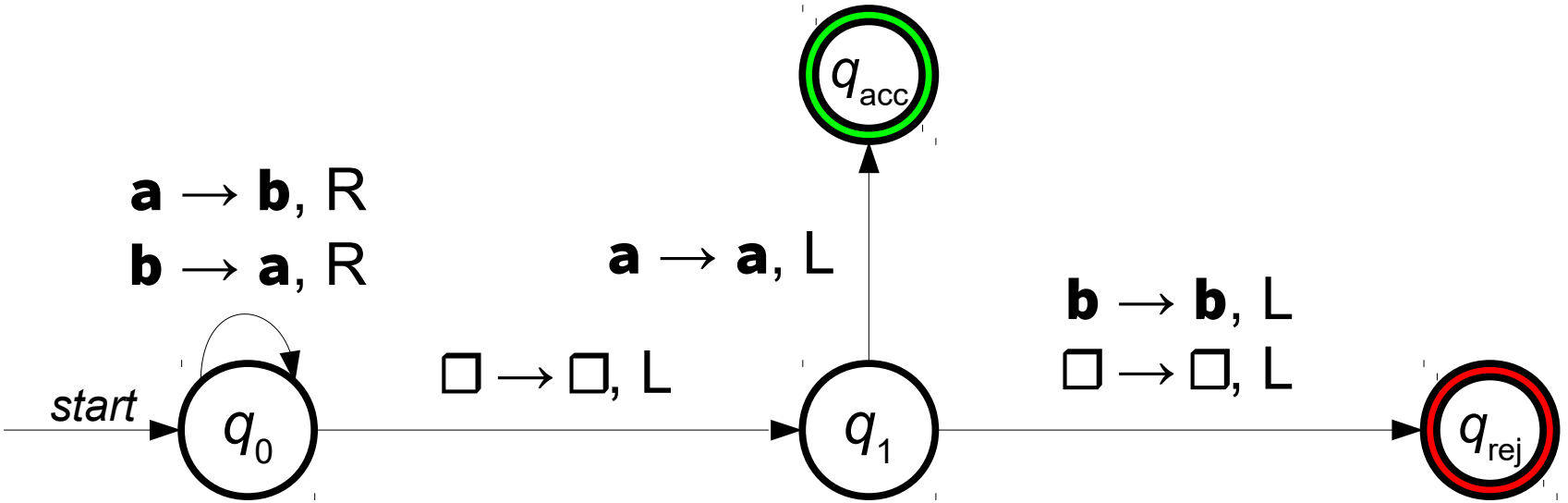

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
a
b
...


$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$


...
a
a
b
...


$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
a
b
...


$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
a
b
...



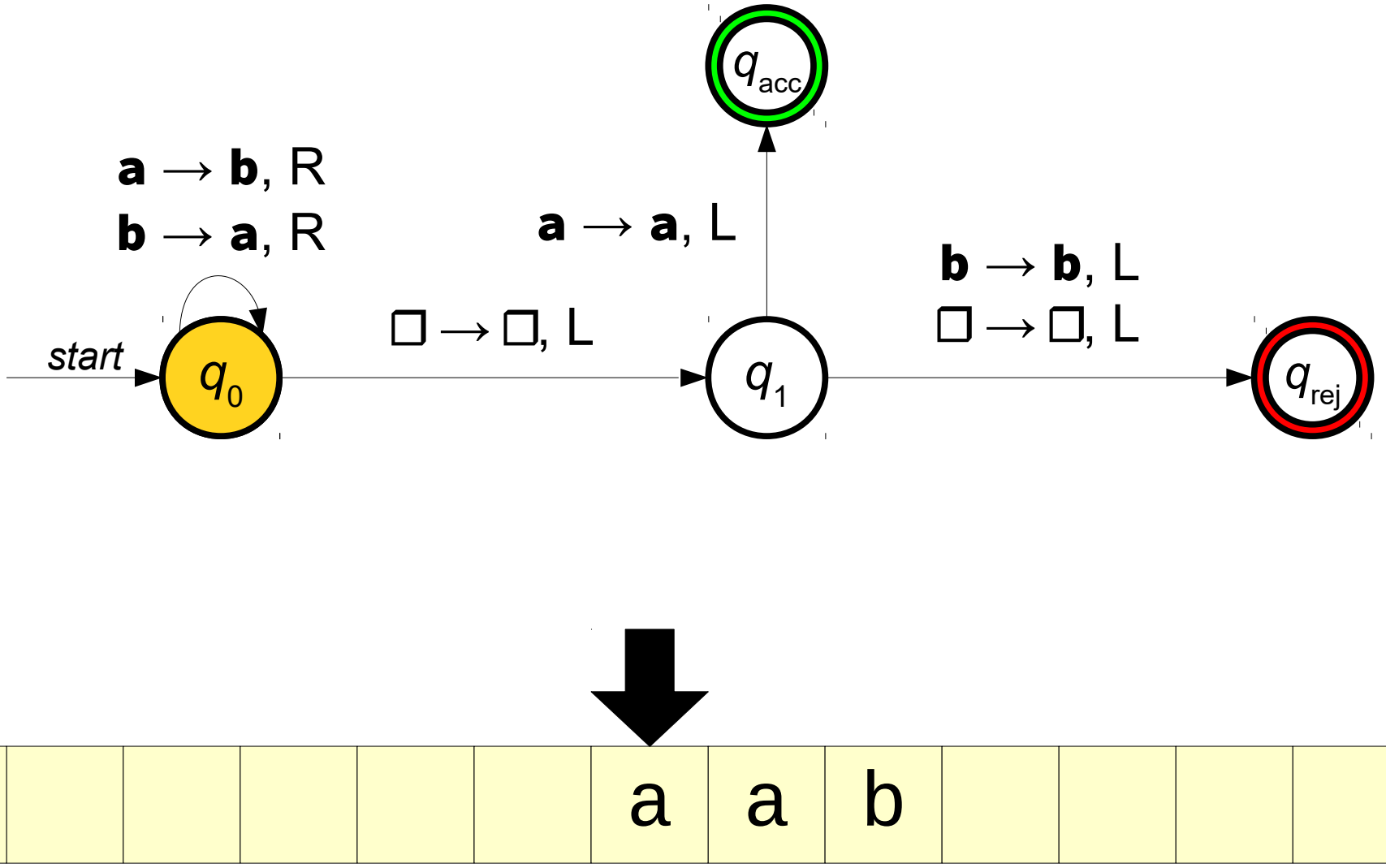

If $M$ is a Turing machine with input alphabet Σ, then the ***language of M***, denoted $\mathscr{L}(M)$, is the set

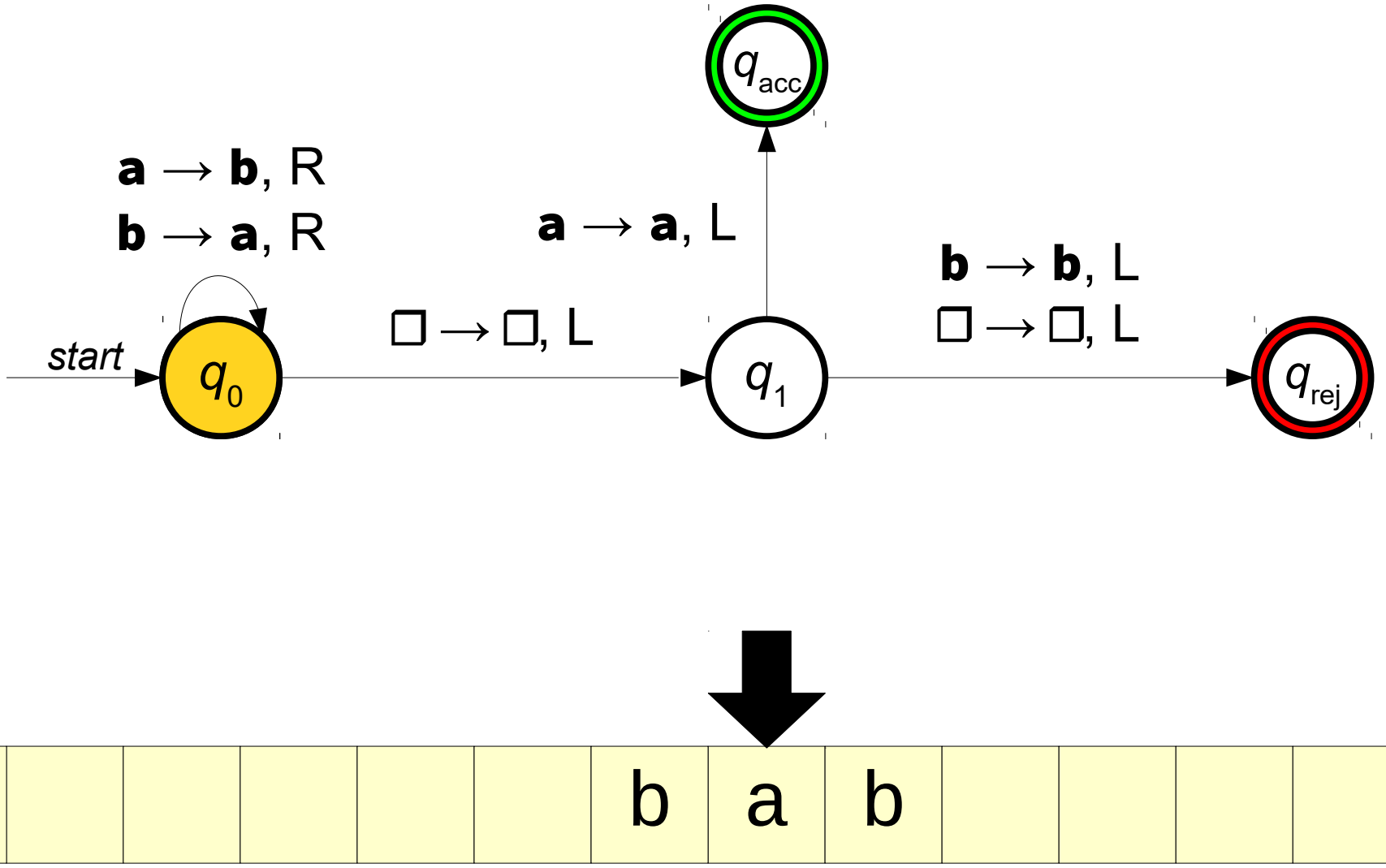
$$\mathscr{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

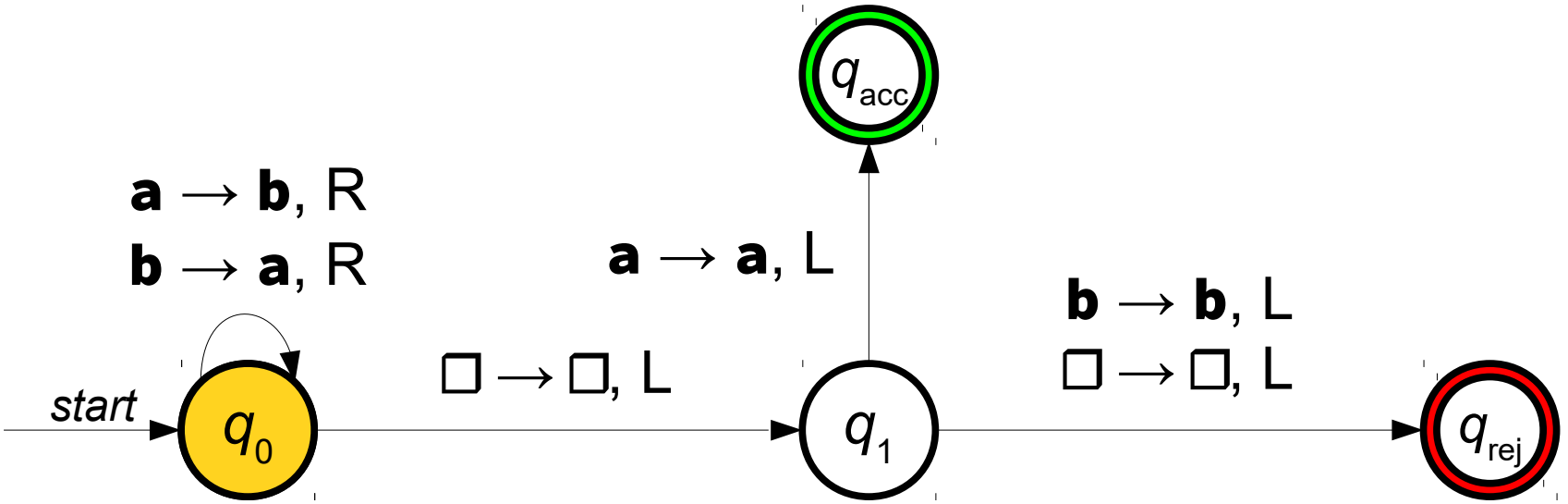
Let $M$ be the above TM, and assume its input alphabet is {**a**, **b**}. What is $\mathscr{L}(M)$?

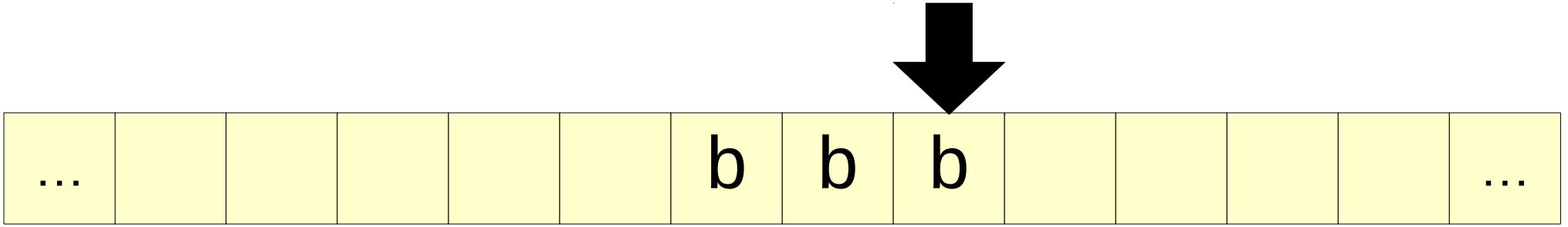
*A.* { $w \in$ {**a**, **b**}* | $w$ ends in **a** }
*B.* { $w \in$ {**a**, **b**}* | $w$ ends in **b** }
*C.* Ø
*D.* None of these, or two or more of these.

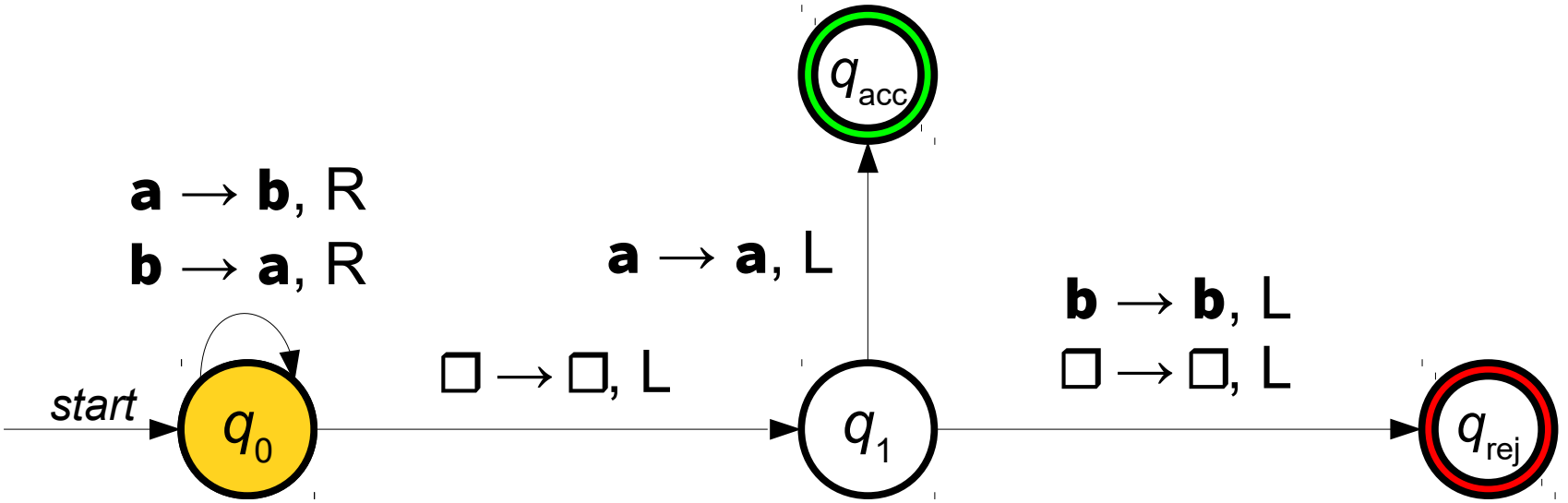
Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then ***A***, ***B***, ***C***, or ***D***.

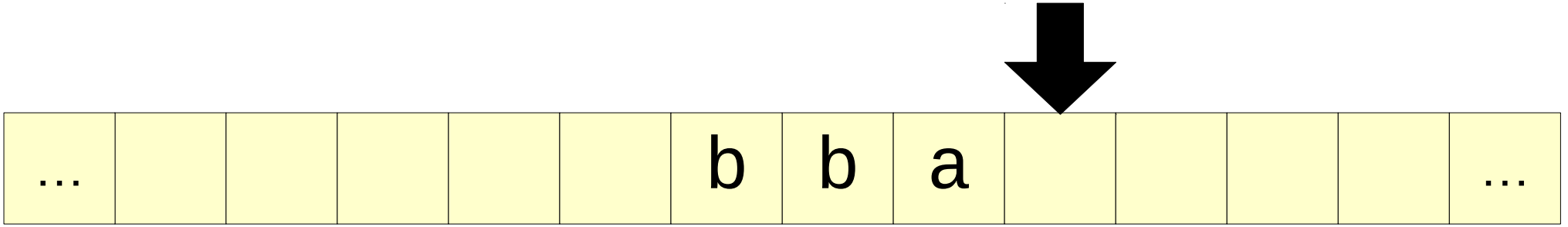

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
a
a
b
...

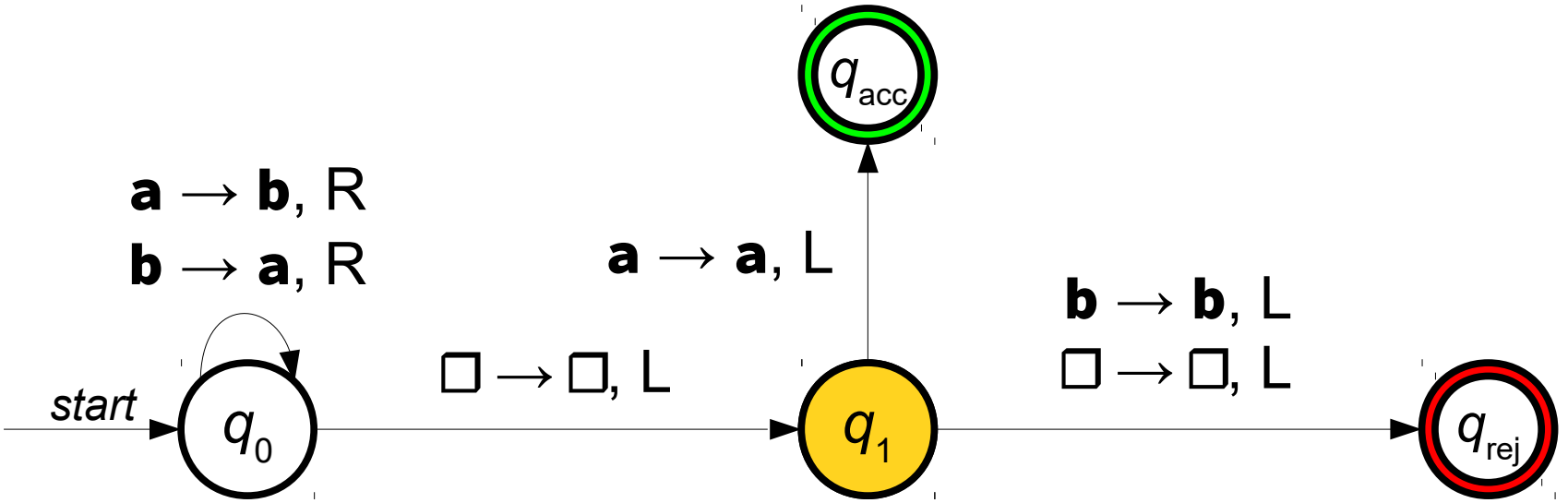

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
b
a
b
...

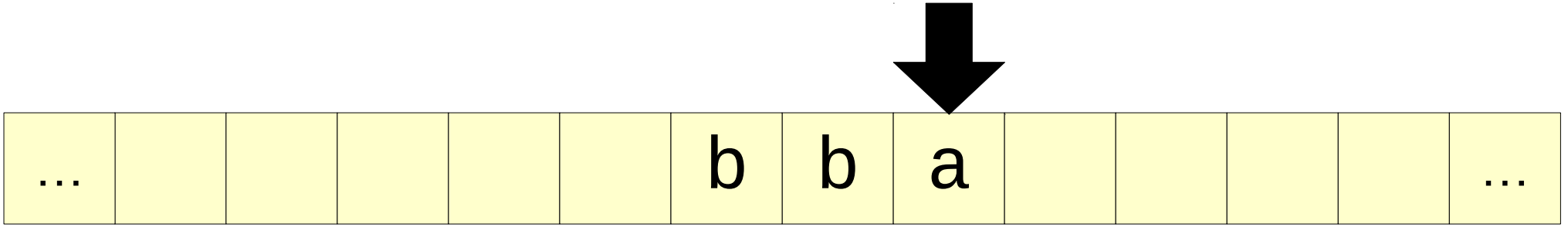

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$

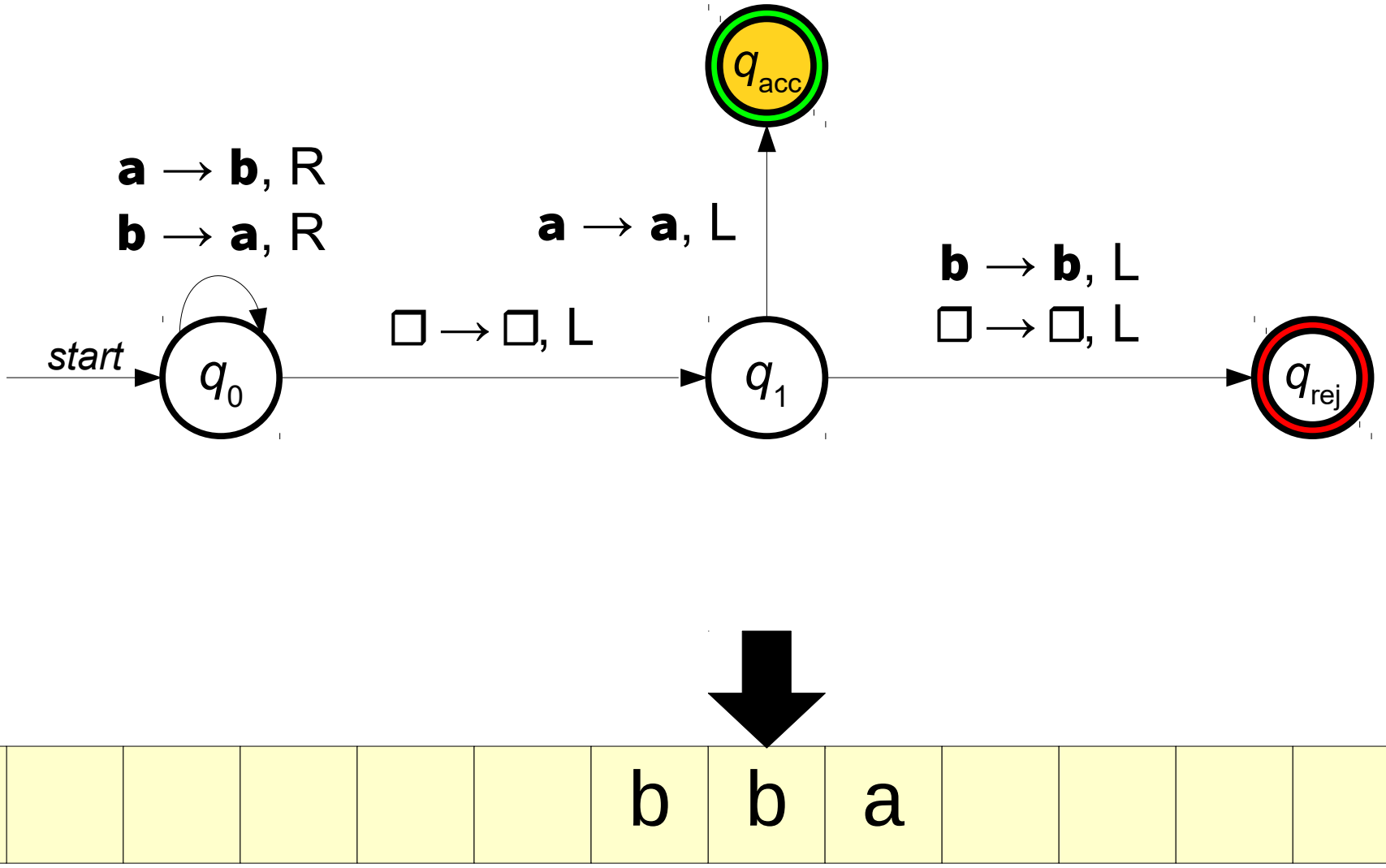

...
b
b
b
...

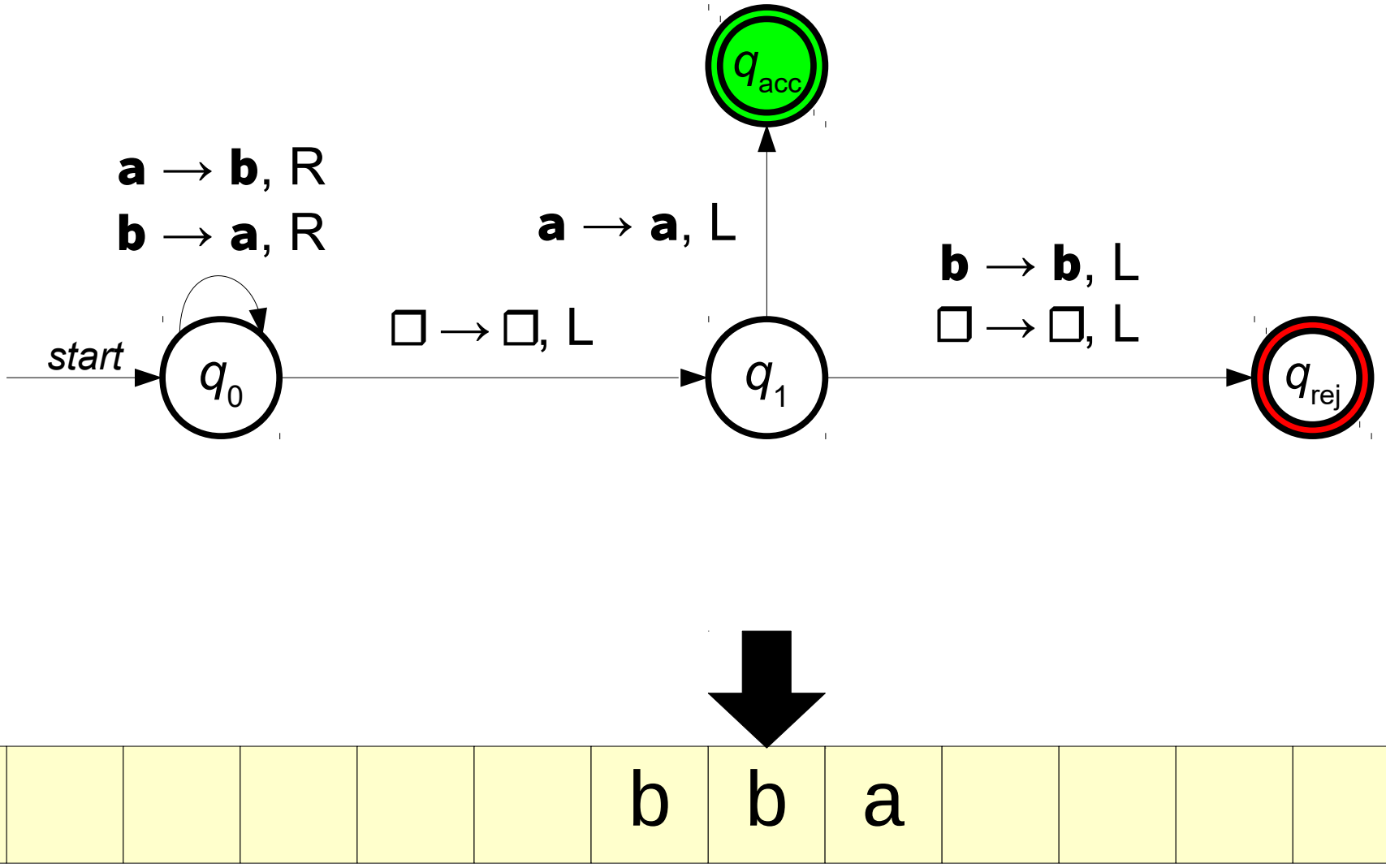

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$

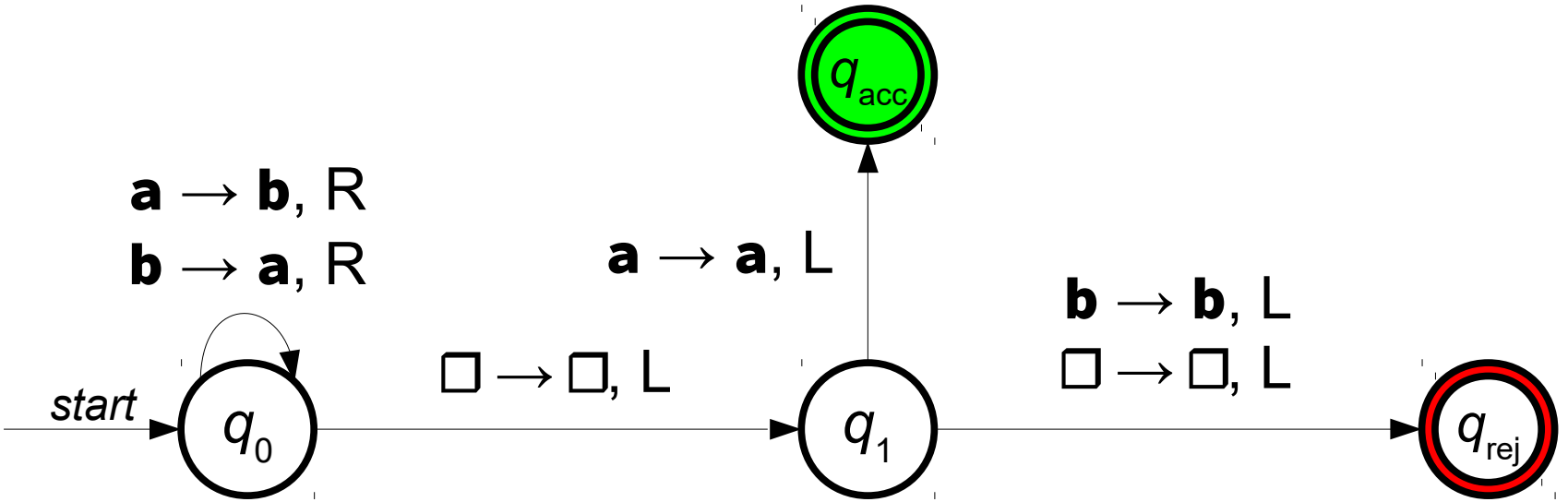

...
b
b
a
...

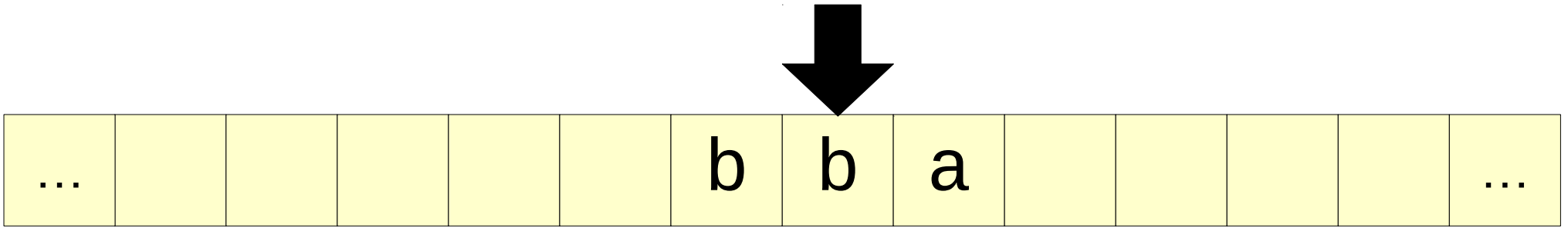

$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$


...
b
b
a
...


$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
b
b
a
...


$q_{acc}$
**a** → **b**, R
**b** → **a**, R
**a** → **a**, L
**b** → **b**, L
□ → □, L
□ → □, L
start
$q_0$
$q_1$
$q_{rej}$
...
b
b
a
...




Although the tape ends with **bba** written on it, the original input string was **aab**. This shows that the TM accepts **aab**, not **bba**.

So $\mathscr{L}(M) = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ ends in } \mathbf{b} \}$