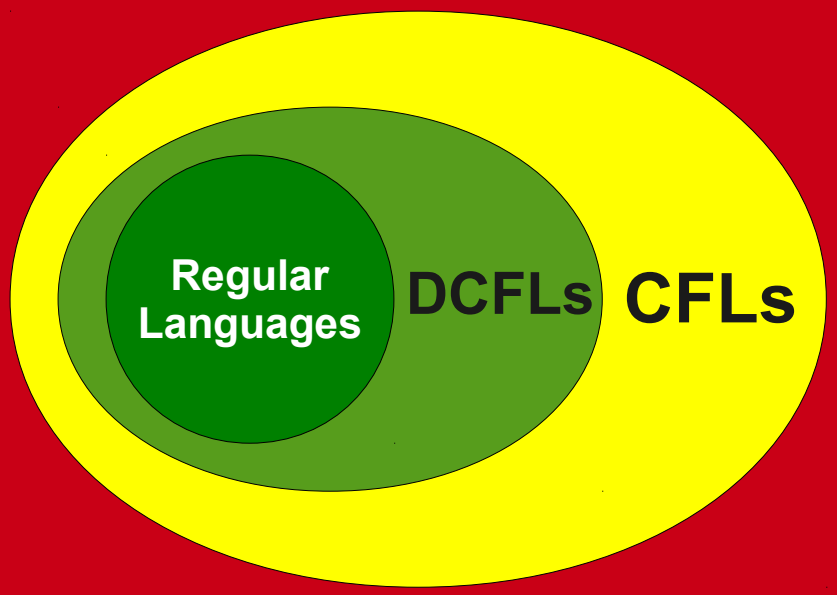
Beyond Context-Free Languages

Are some problems inherently harder than others?



What sorts of languages are out here?

All Languages

The Pumping Lemma for Regular Languages

- Let L be a regular language, so there is a DFA D for L.
- A sufficiently long string $w \in L$ must visit some state in D twice.
- This means w went through a loop in the D.
- By replicating the characters that went through the loop in the D, we can “pump” a portion of w to produce new strings in the language.

The Pumping Lemma Intuition

- The model of computation used has a finite description.
- For sufficiently long strings, the model of computation must repeat some step of the computation to recognize the string.
- Under the right circumstances, we can iterate this repeated step zero or more times to produce more and more strings.

Recall: Parse Trees

R → **a** | **b** | **c** | …

R → "**ε**"

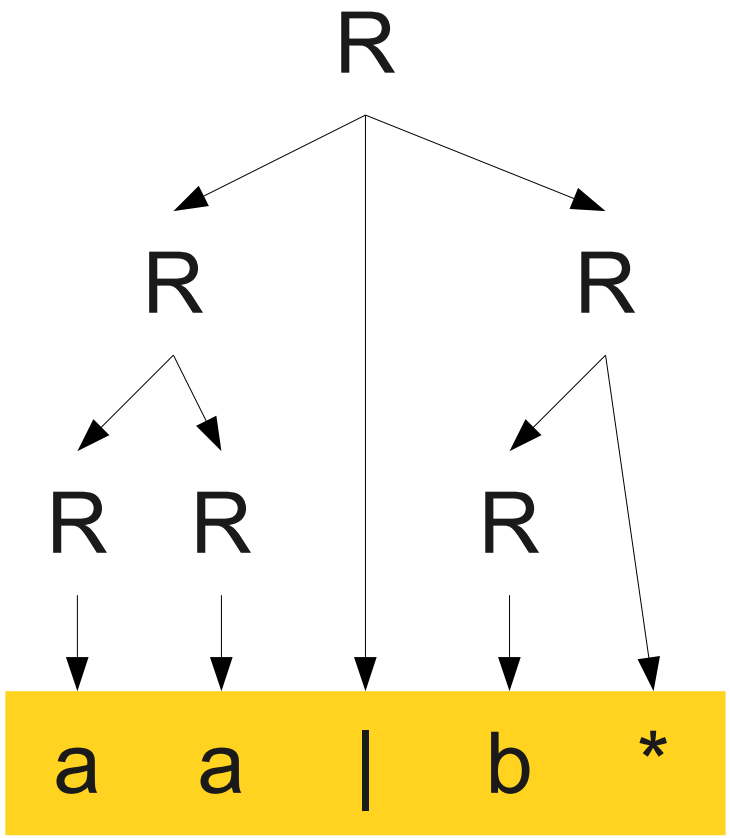
R → **∅**

R → **RR**

R → **R** "**|**" **R**

R → **R***

R → **(R)**



Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

Parse Trees Revisited

S

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

Parse Trees Revisited

S

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

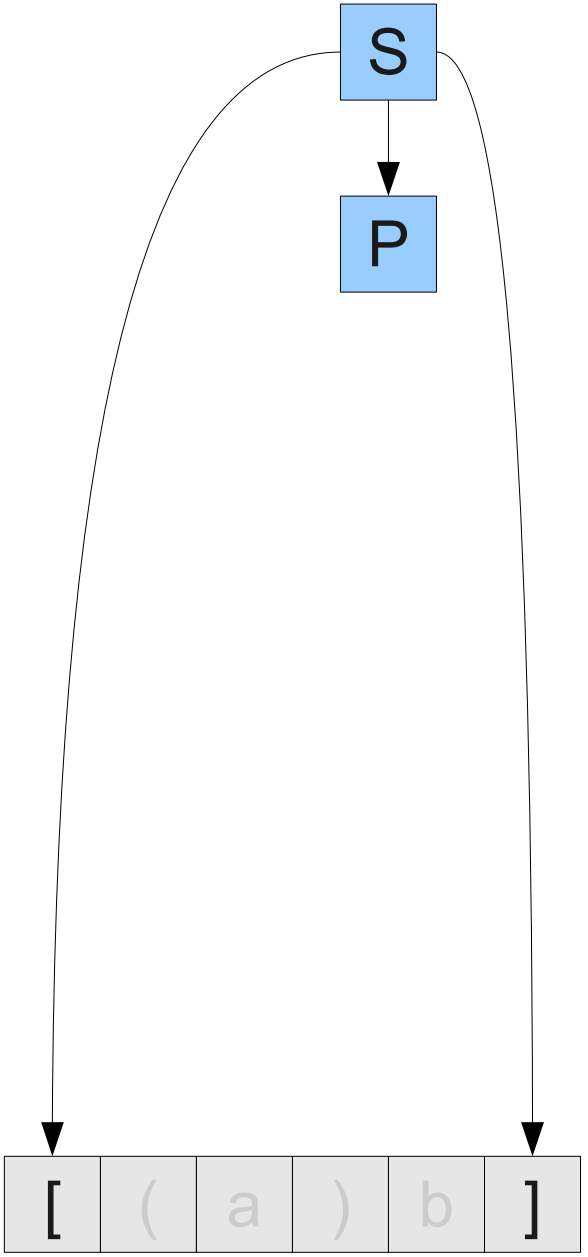
[	(	a	)	b	]

Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

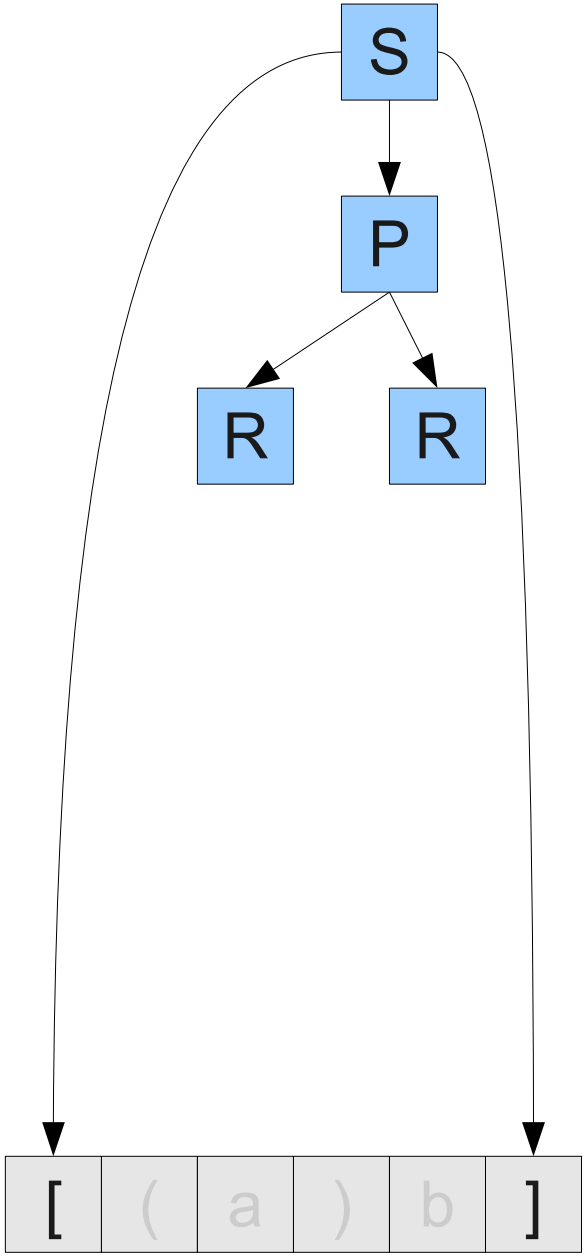


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

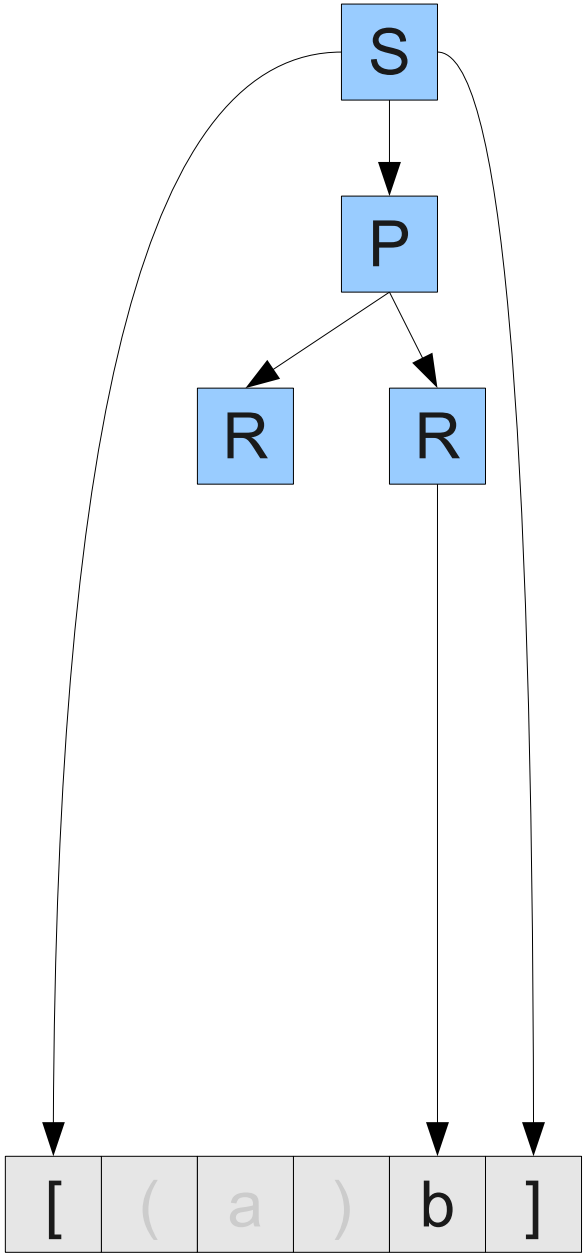


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

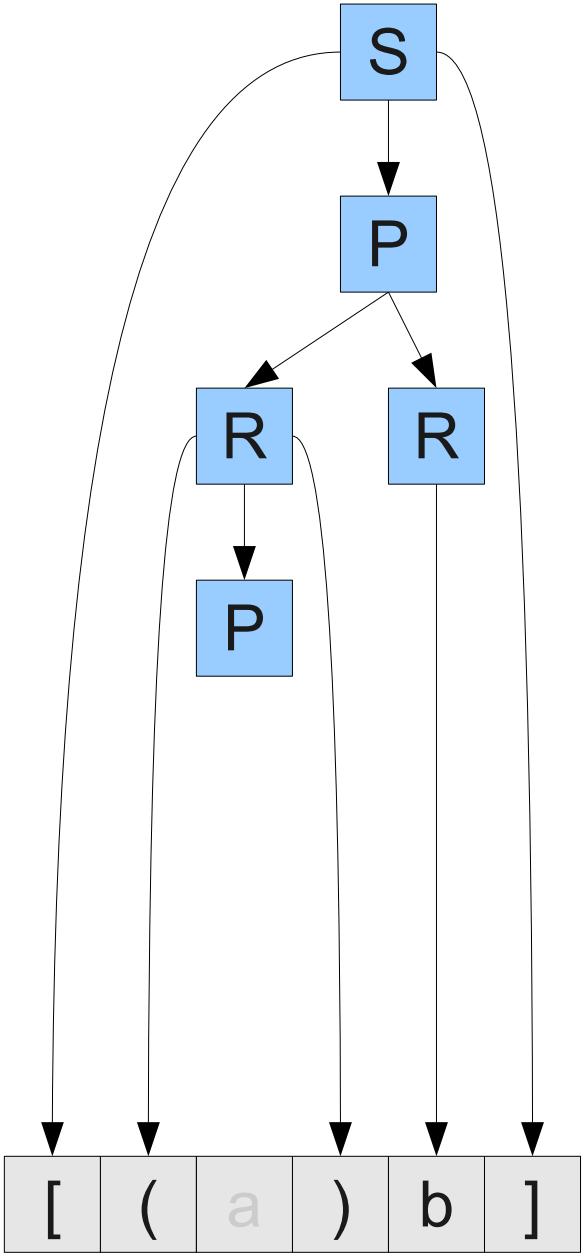


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

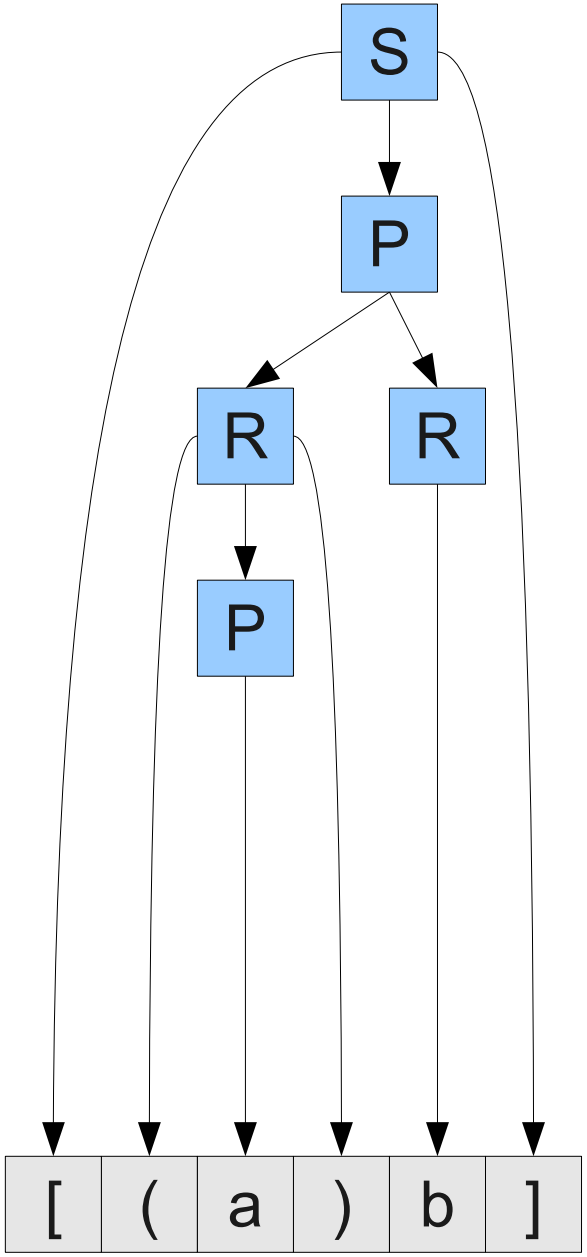


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

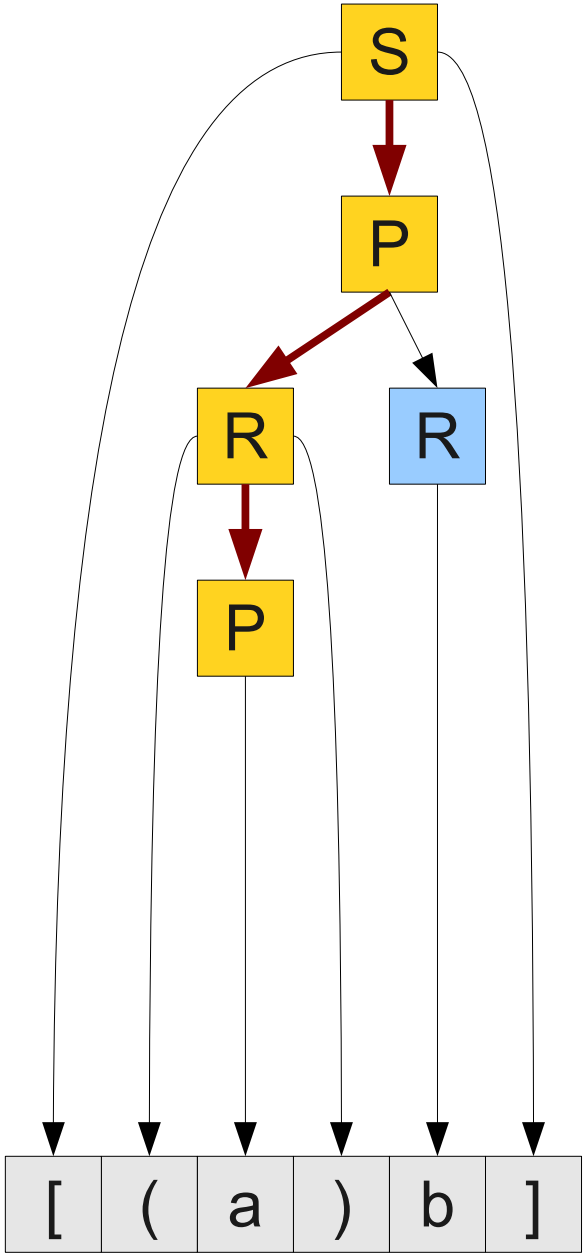


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

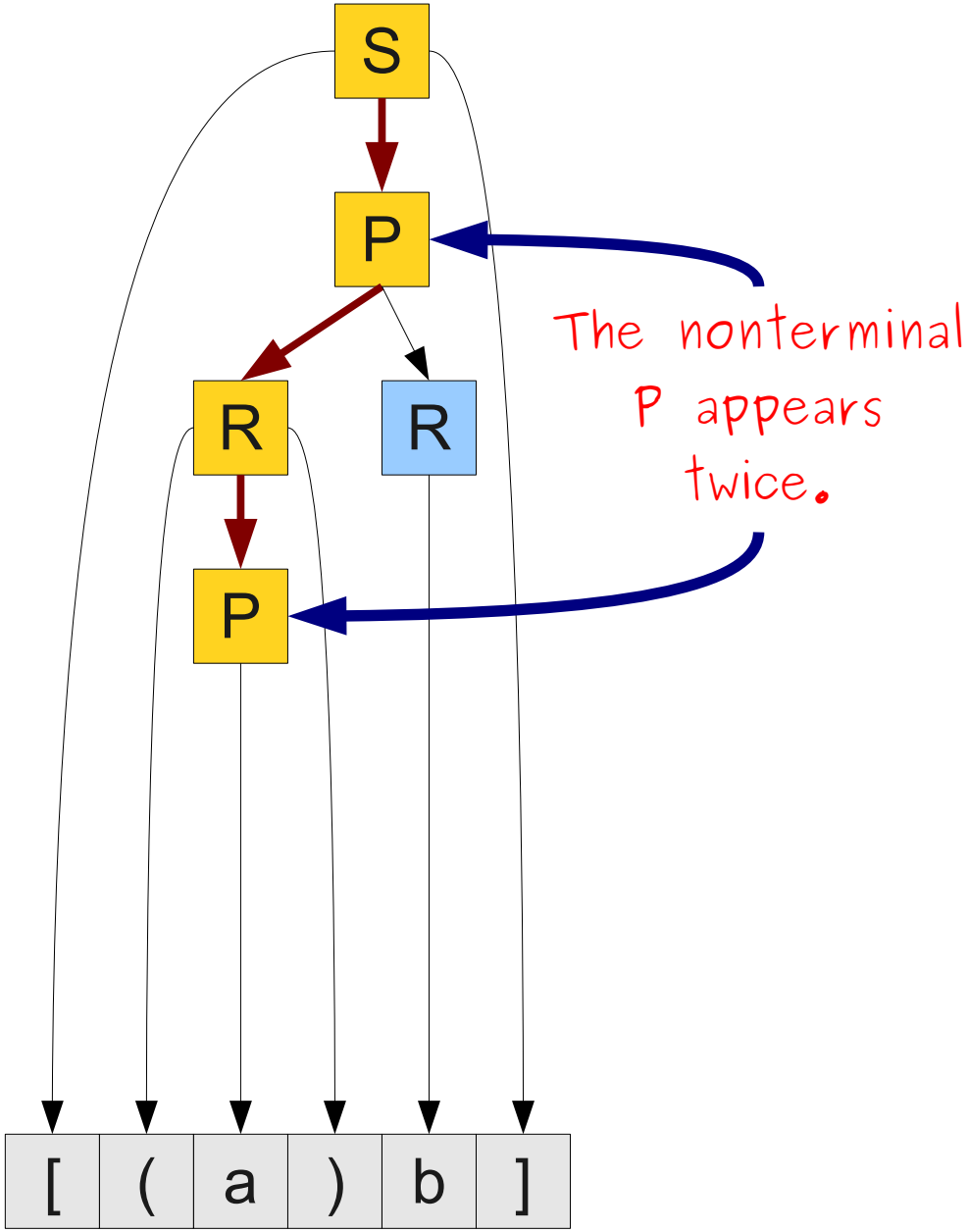


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

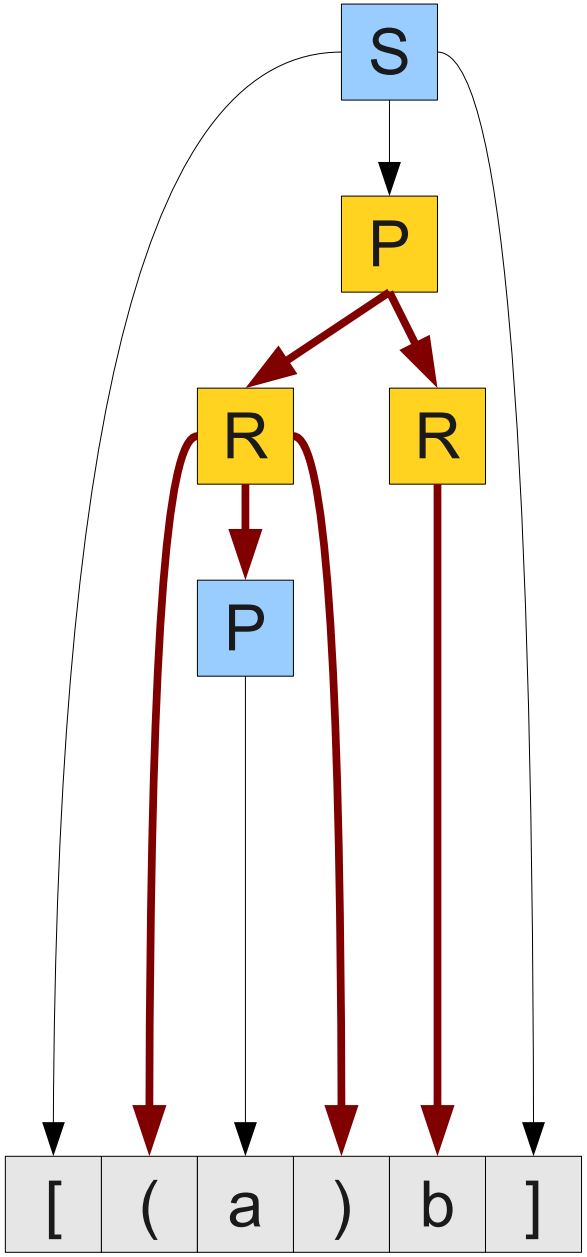


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

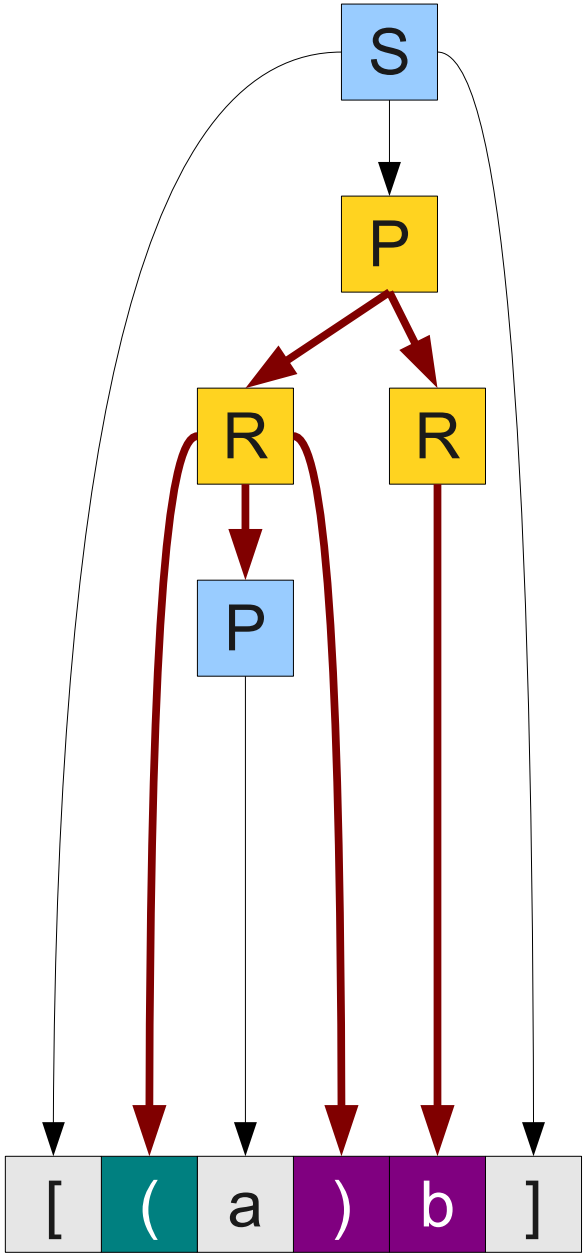


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

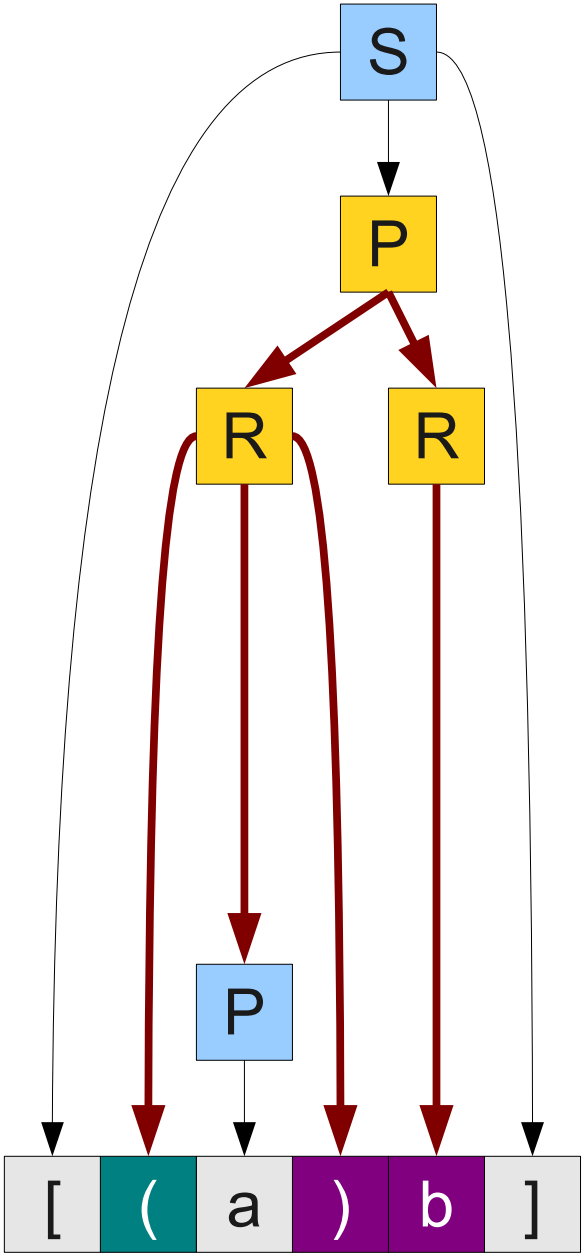


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

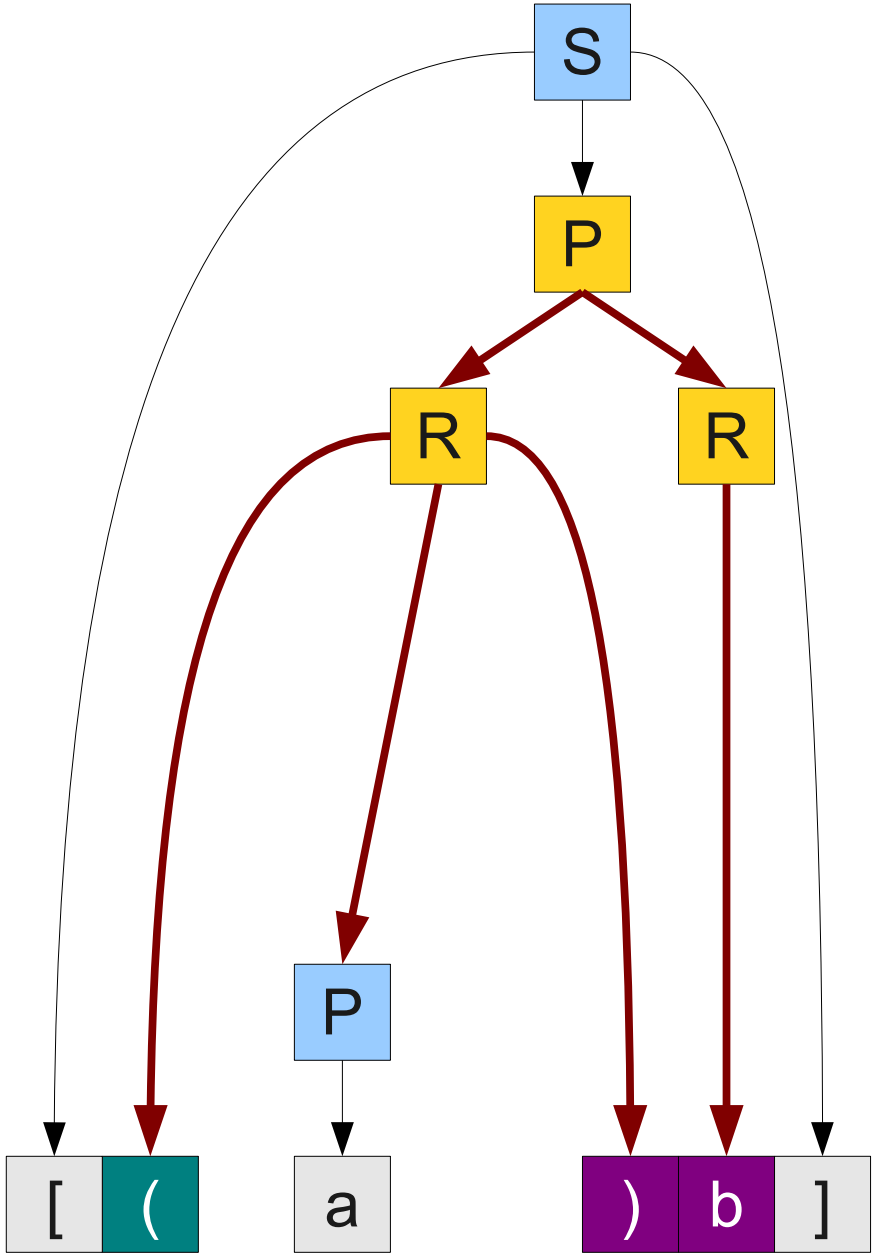


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

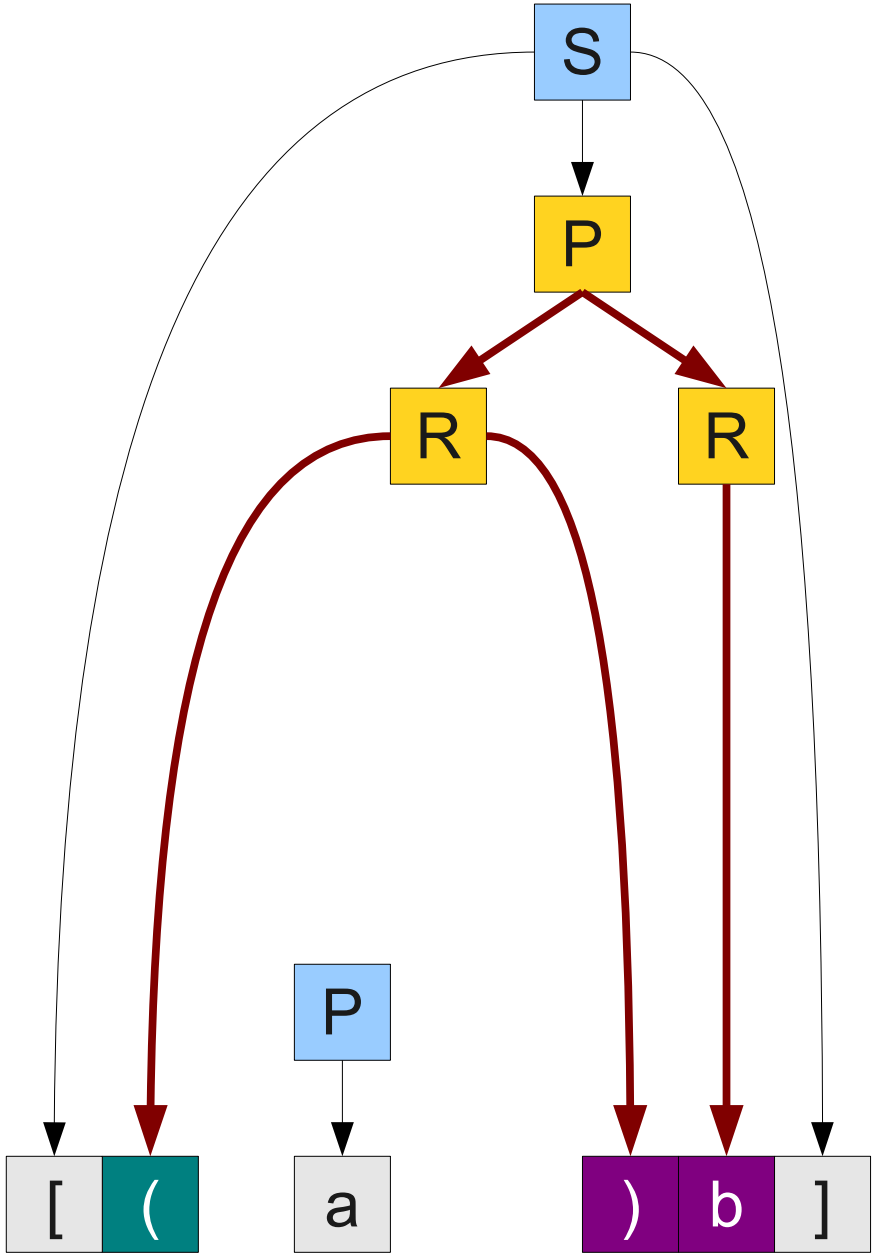


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

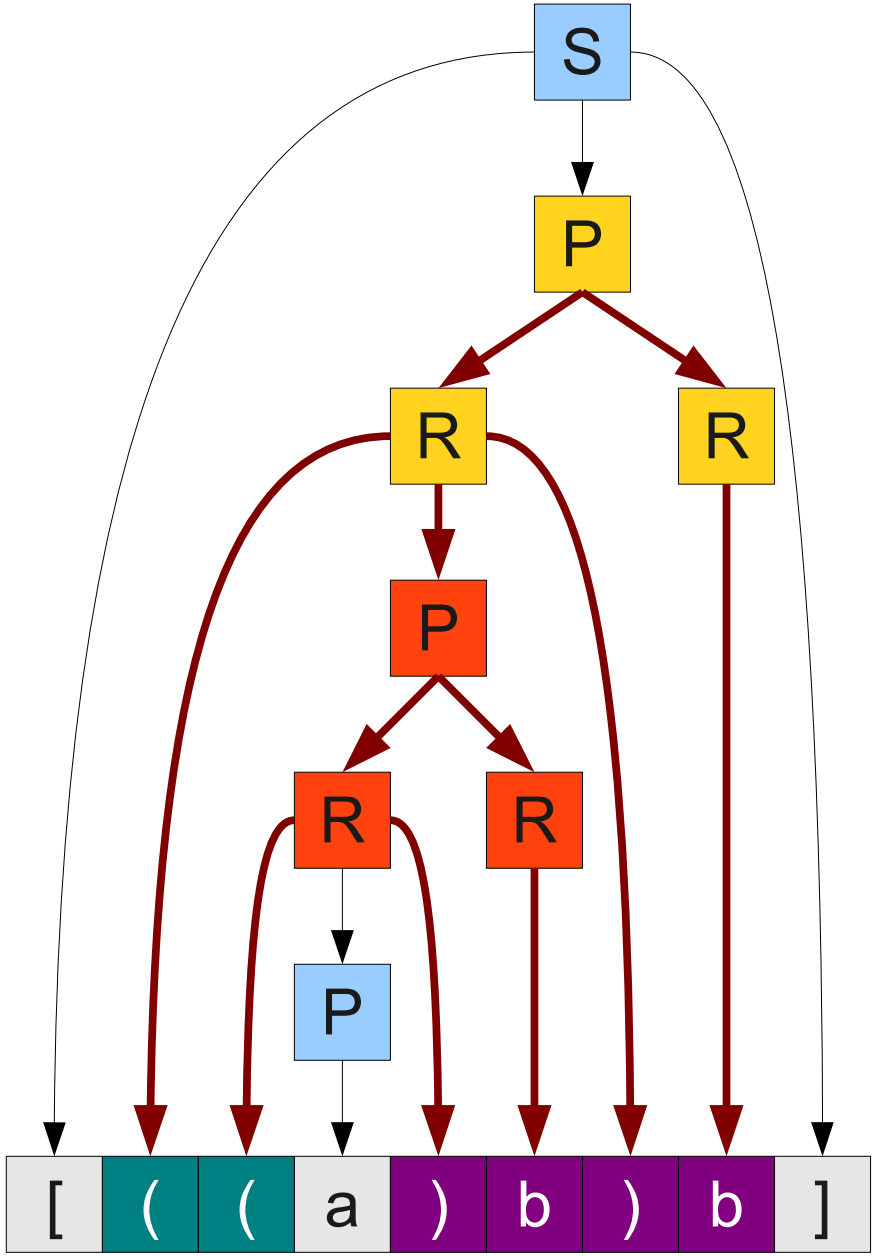


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

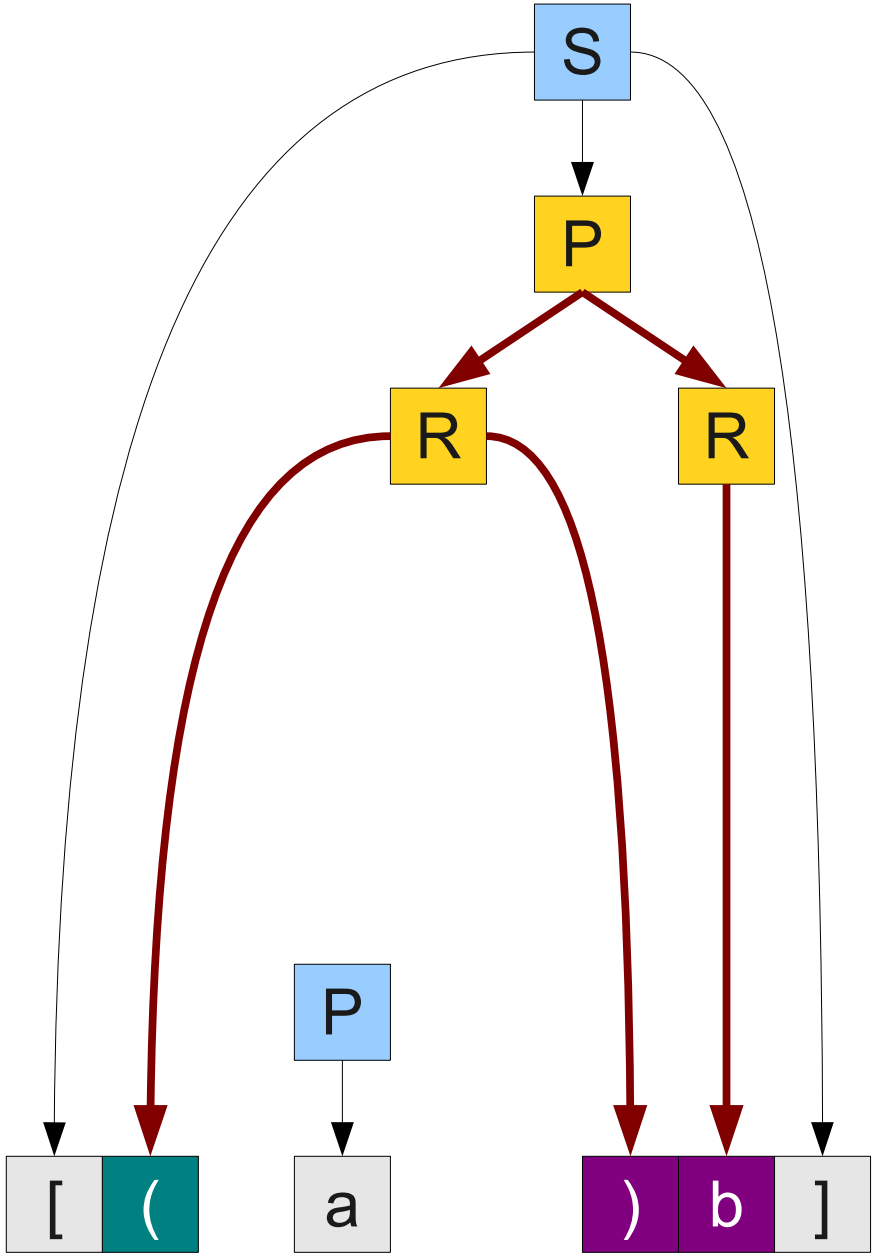


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

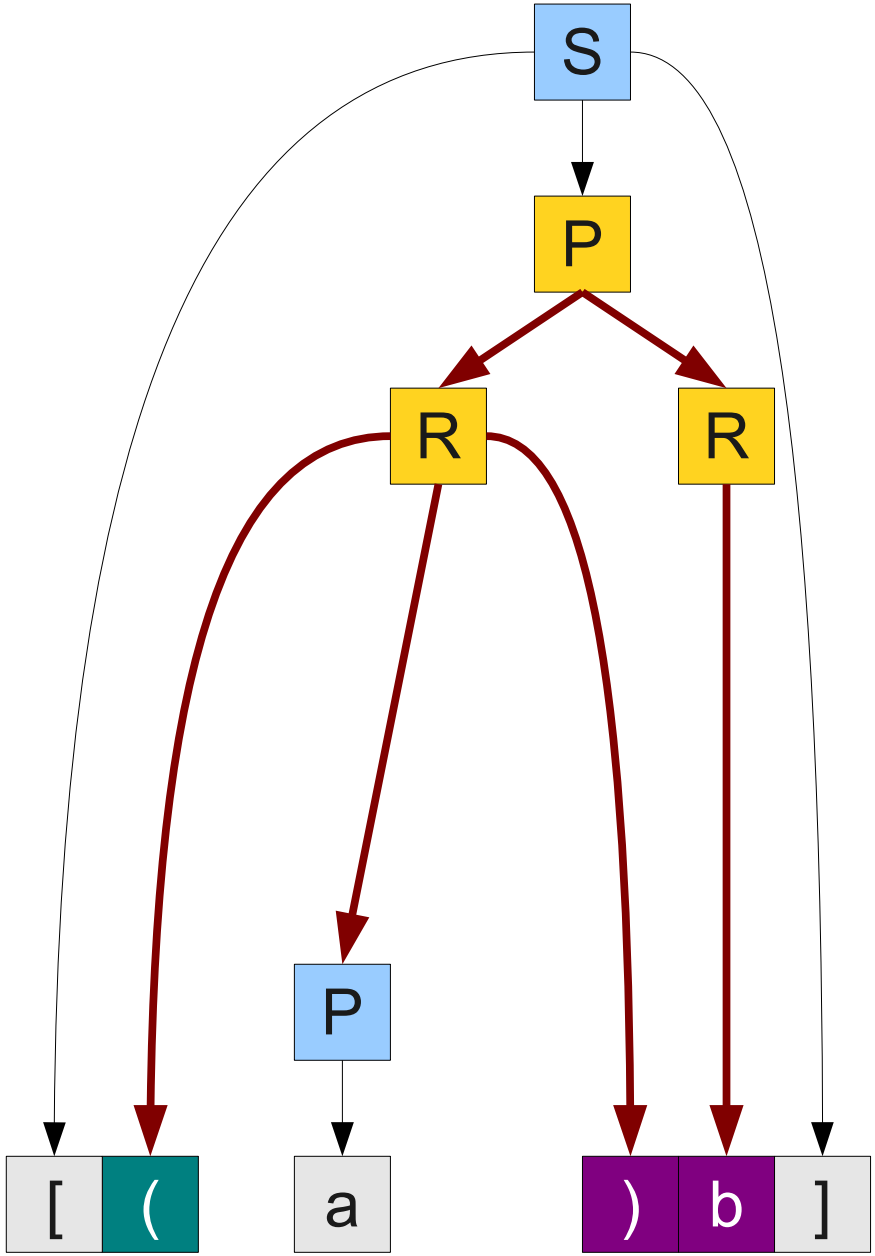


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

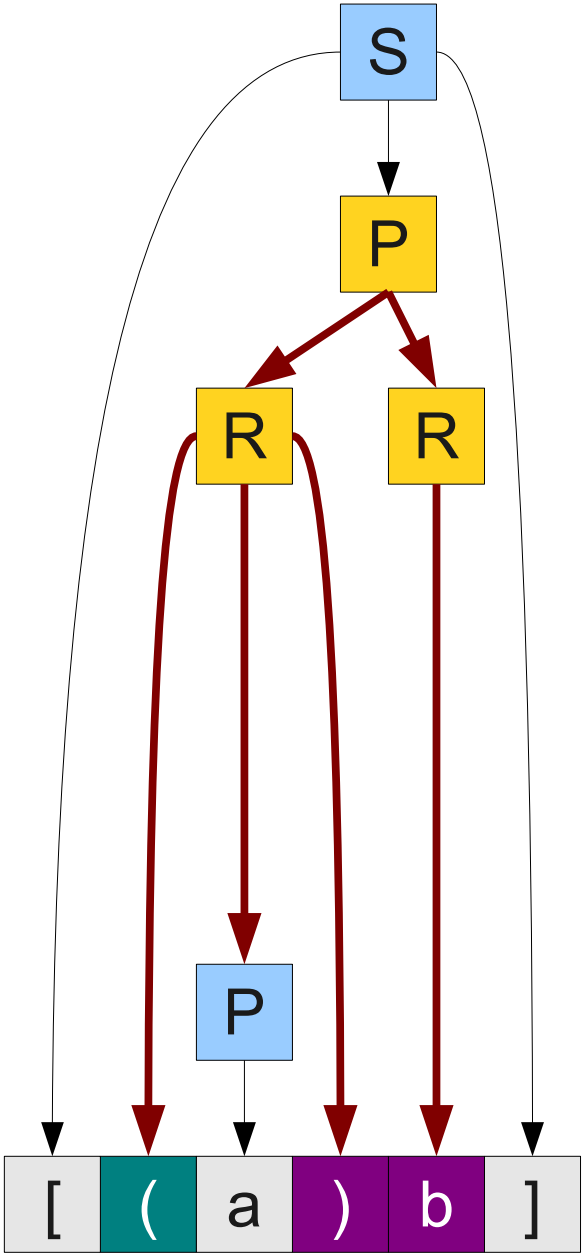


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

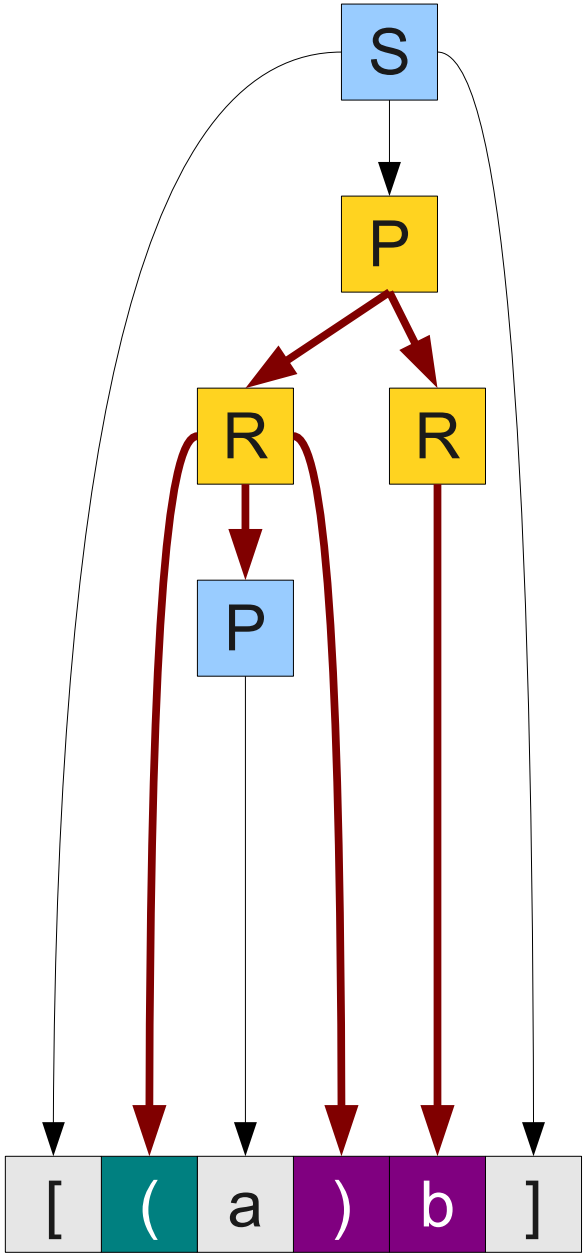


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

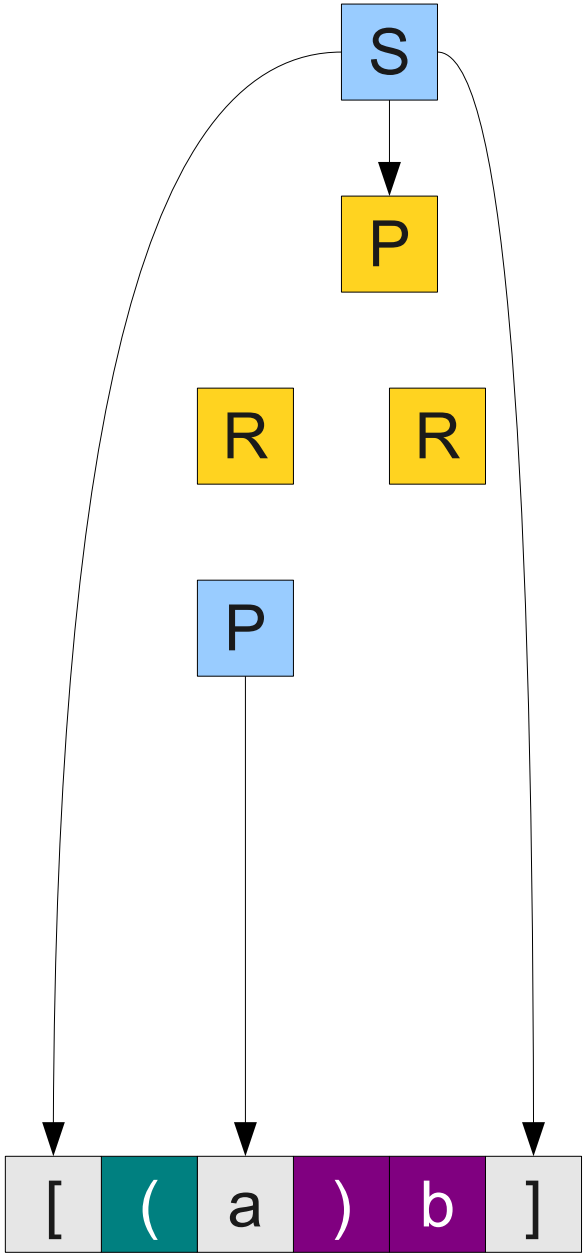


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

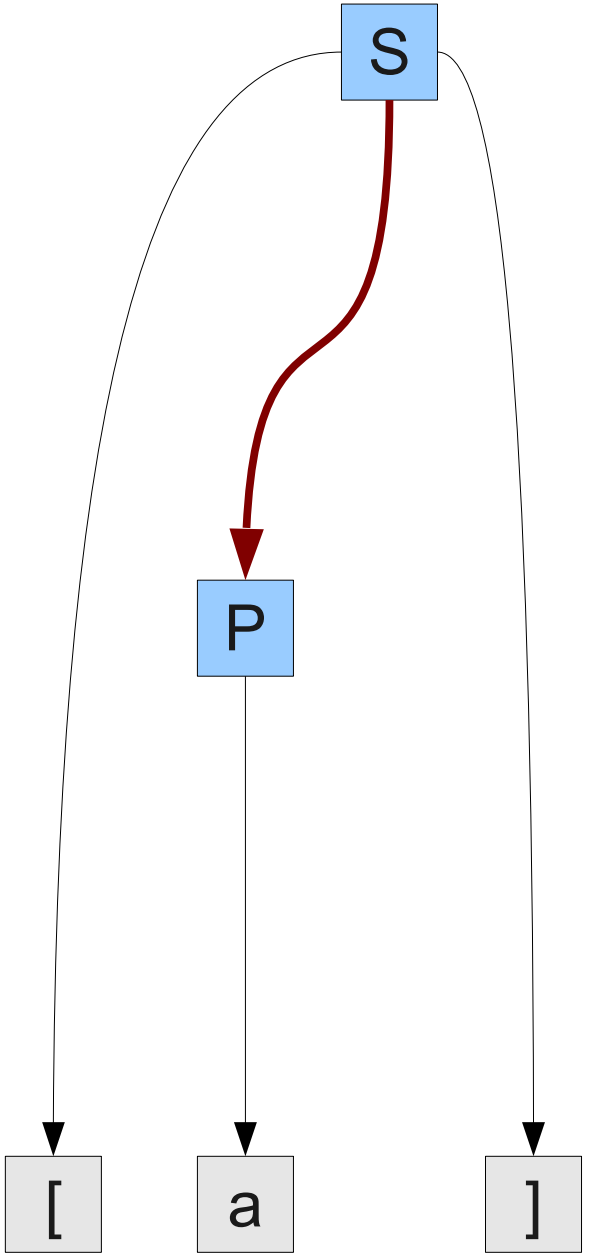


Parse Trees Revisited

S → **[P]**

P → **RR** | **a**

R → **(P)** | **b**

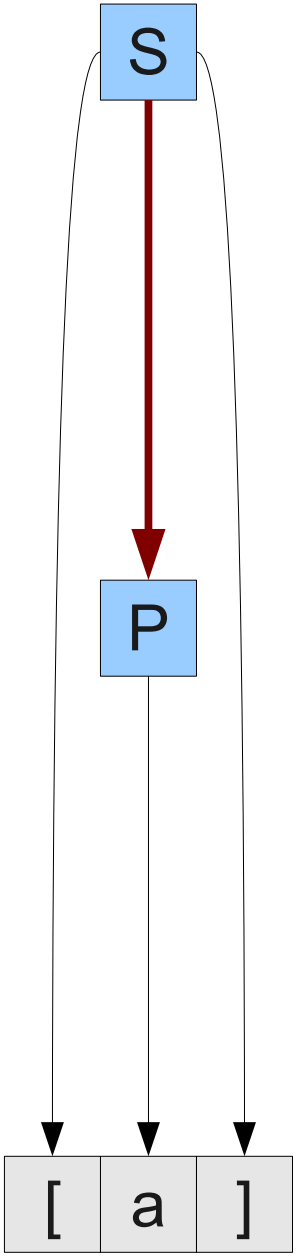


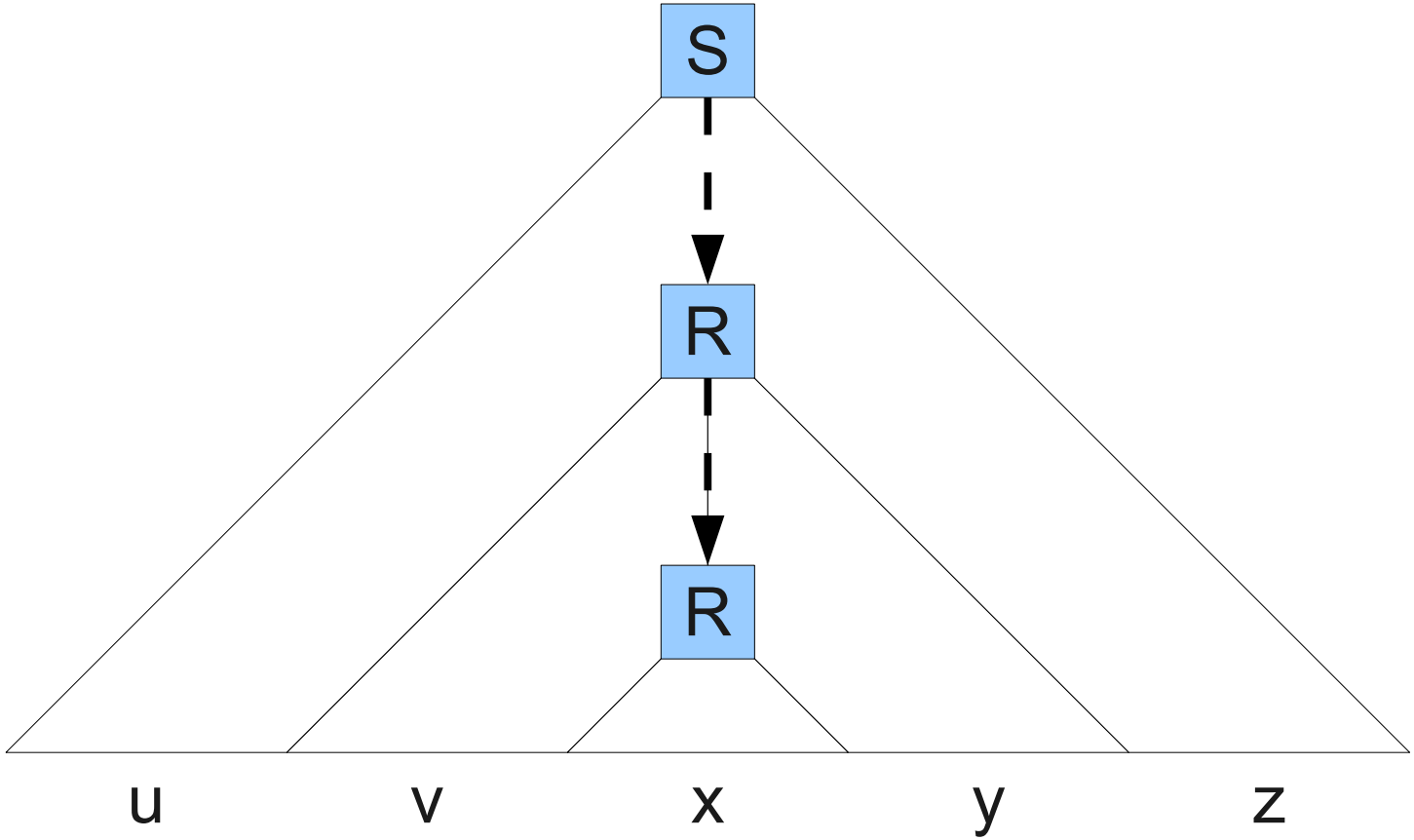
Parse Trees Revisited

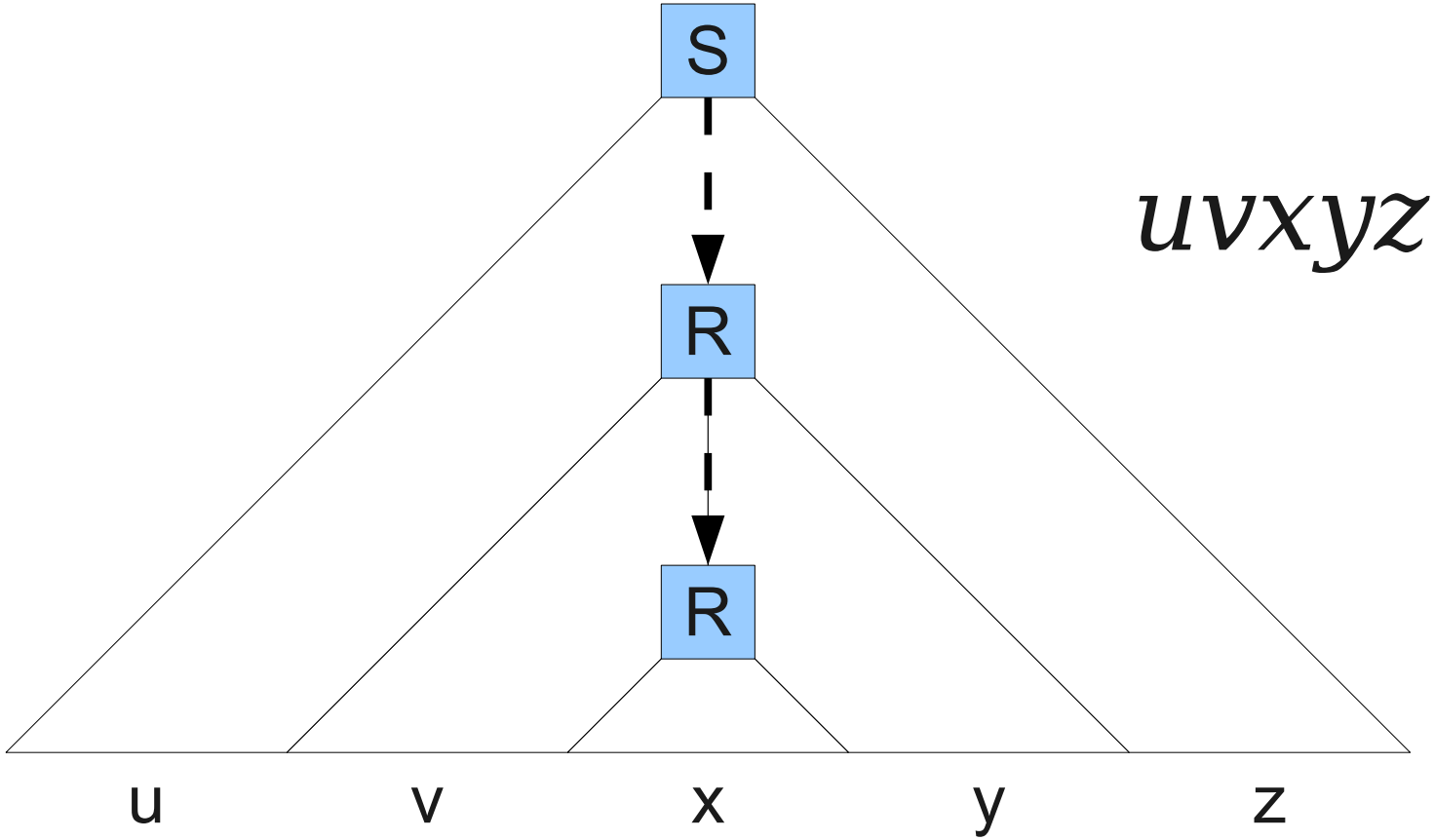
S → **[P]**

P → **RR** | **a**

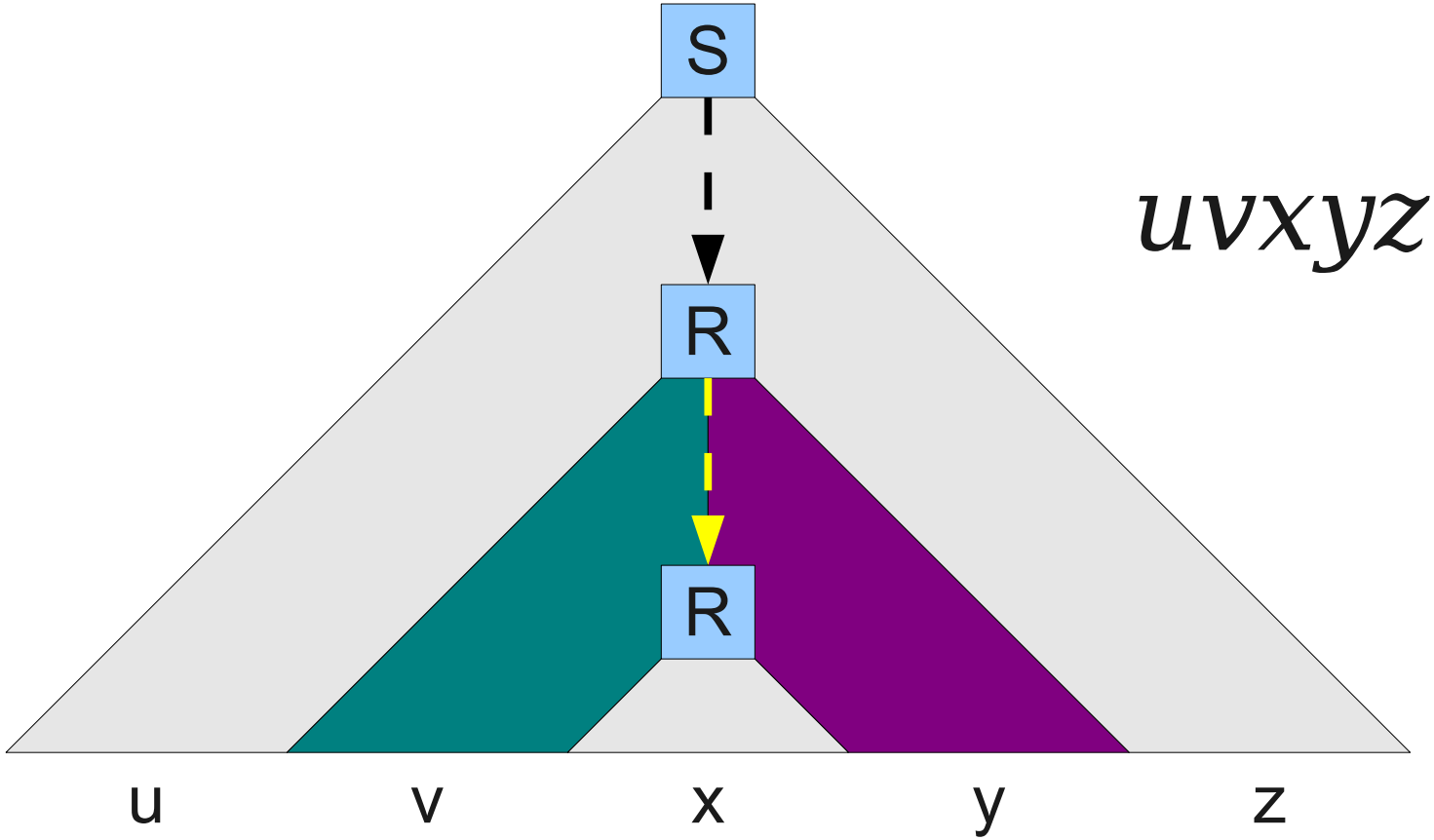
R → **(P)** | **b**



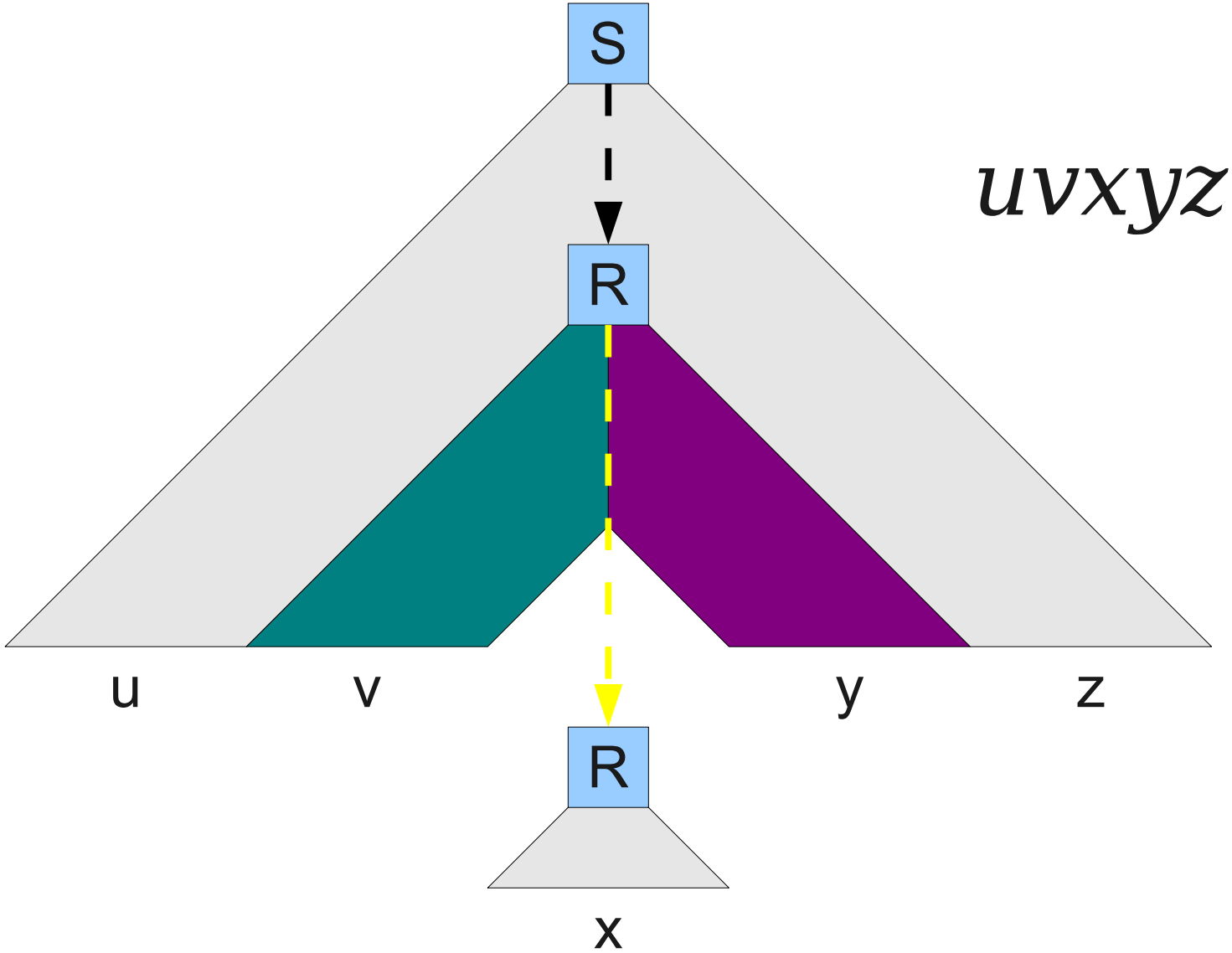

S
R
R
u
v
x
y
z



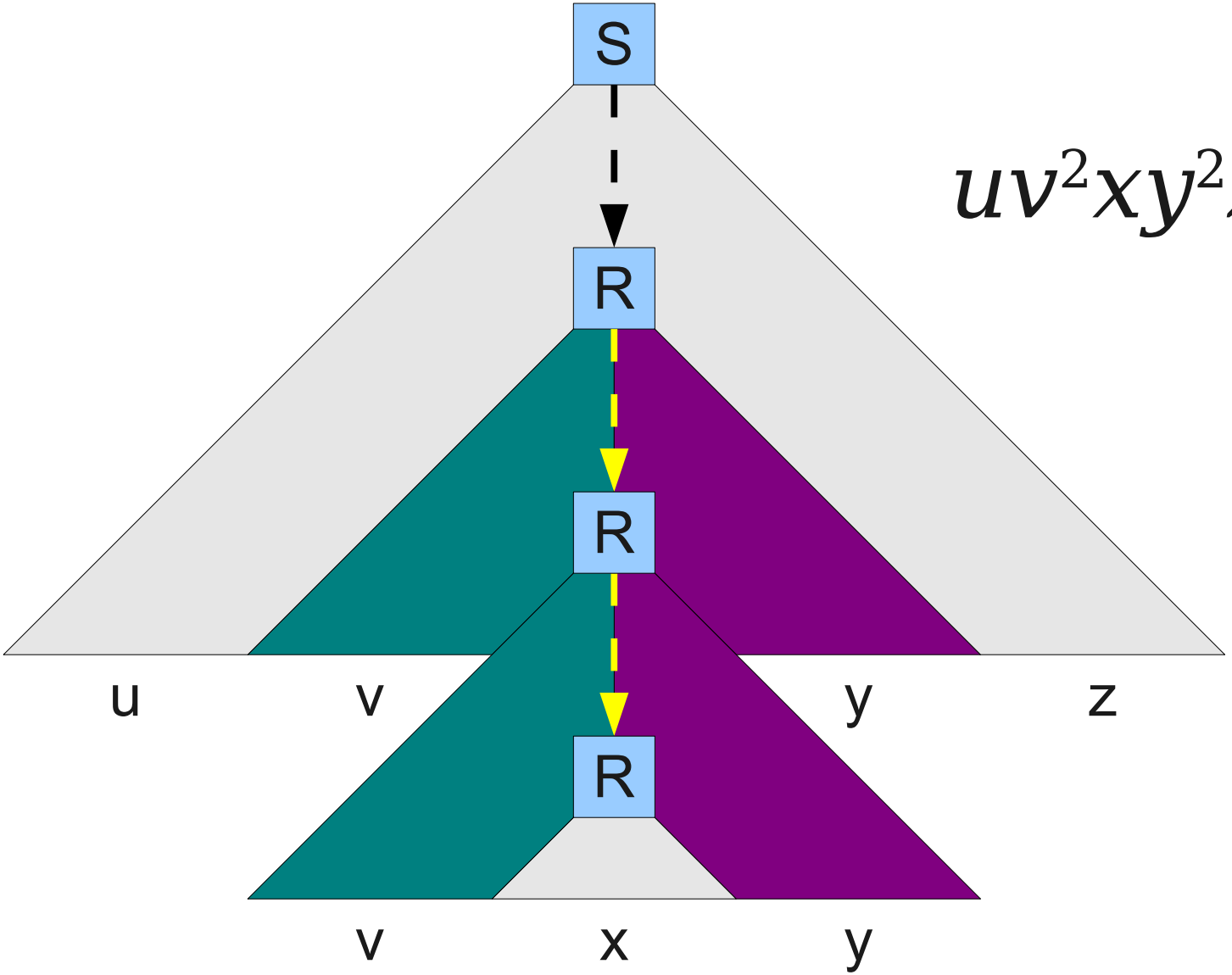
$uvxyz \in L$



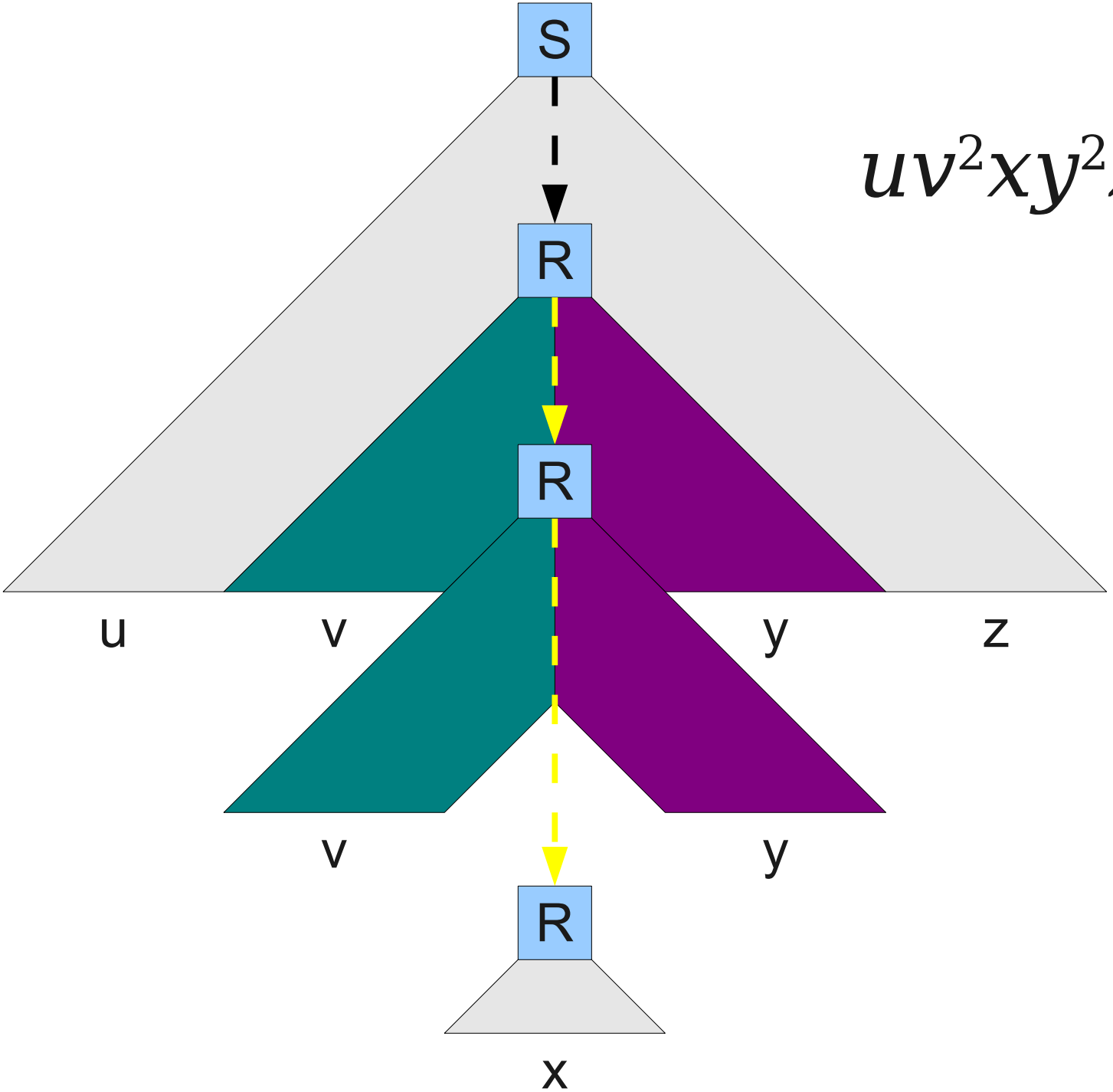
$uvxyz \in L$



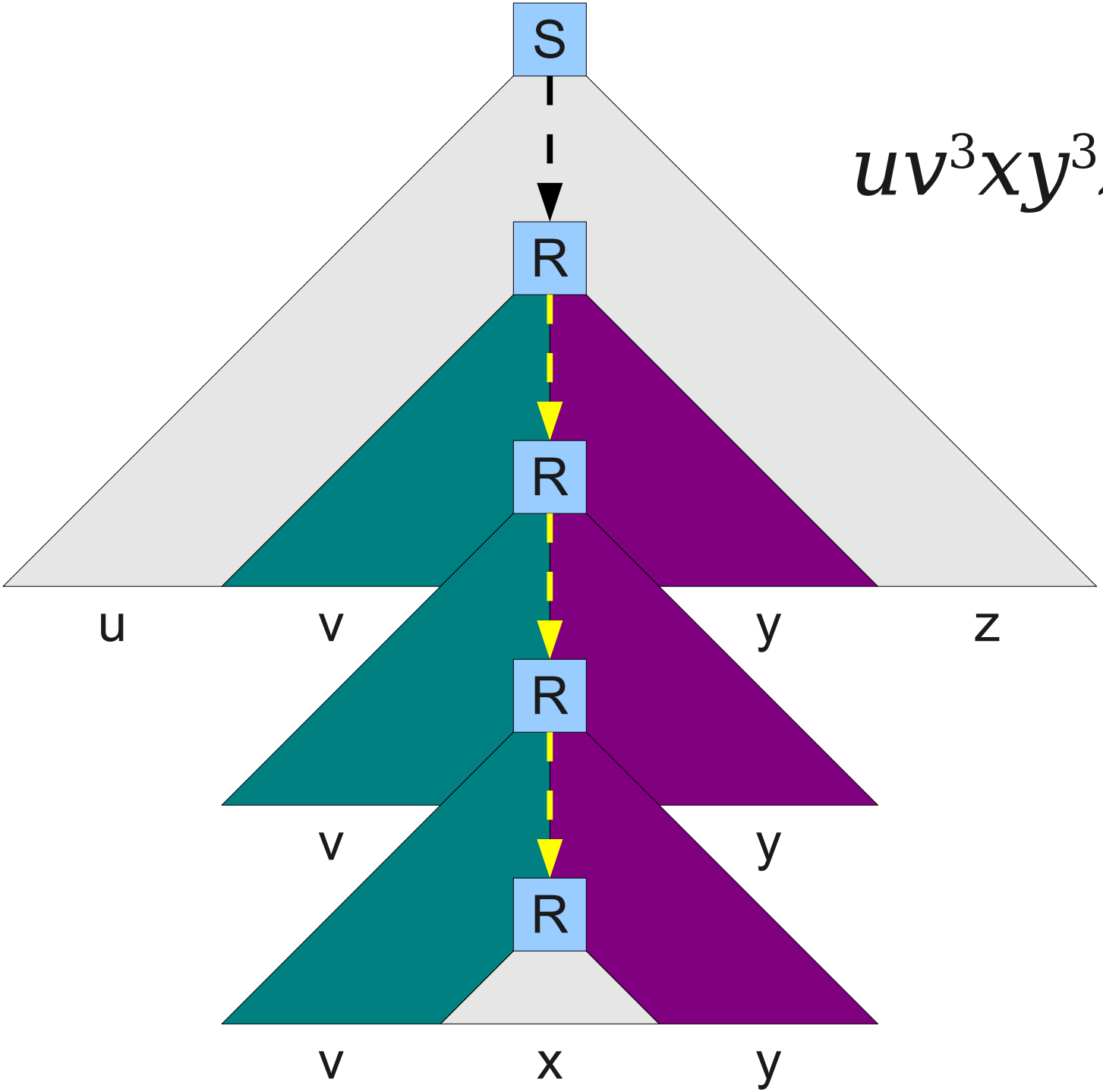
$uvxyz \in L$



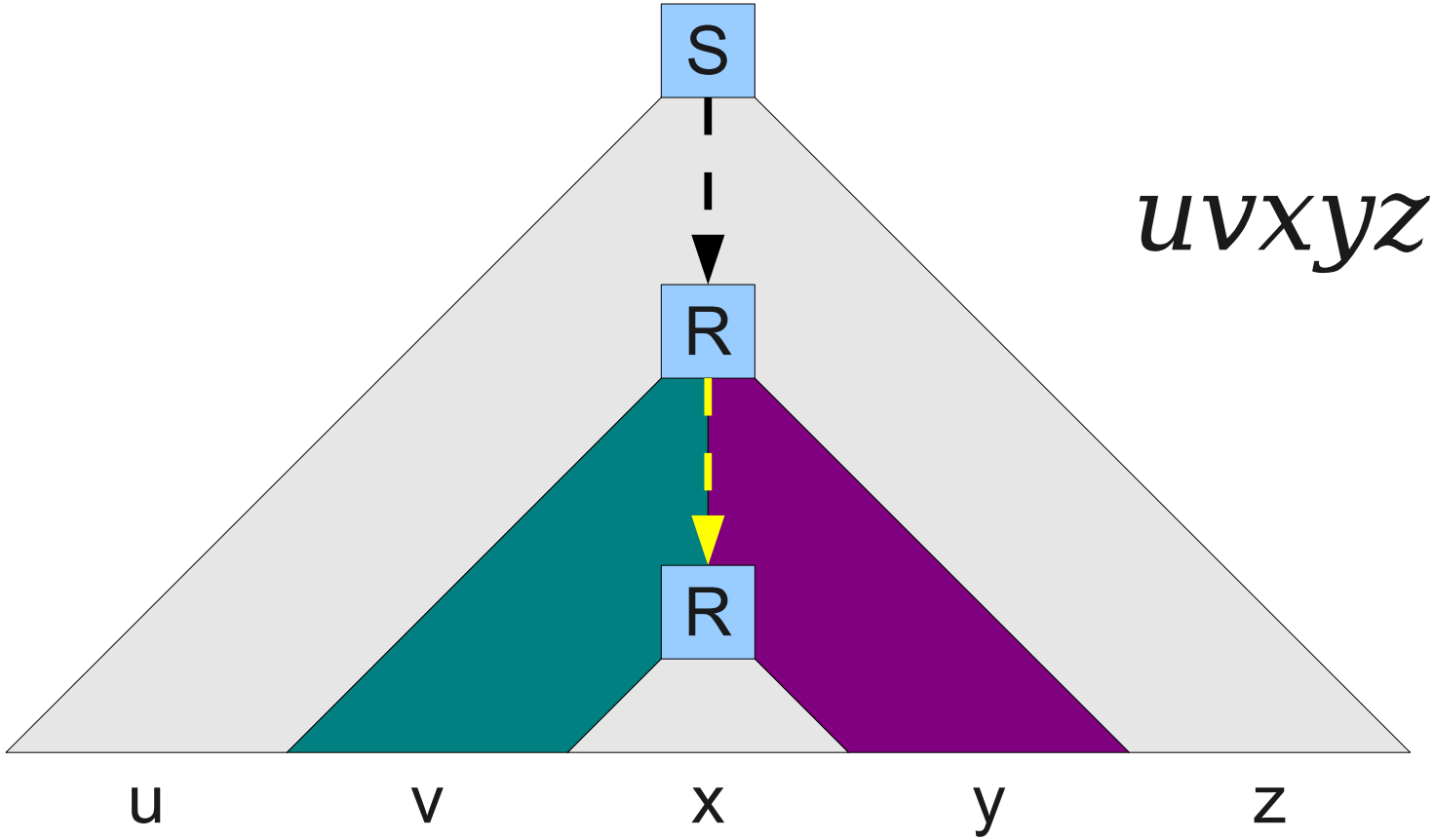
$$uv^2xy^2z \in L$$



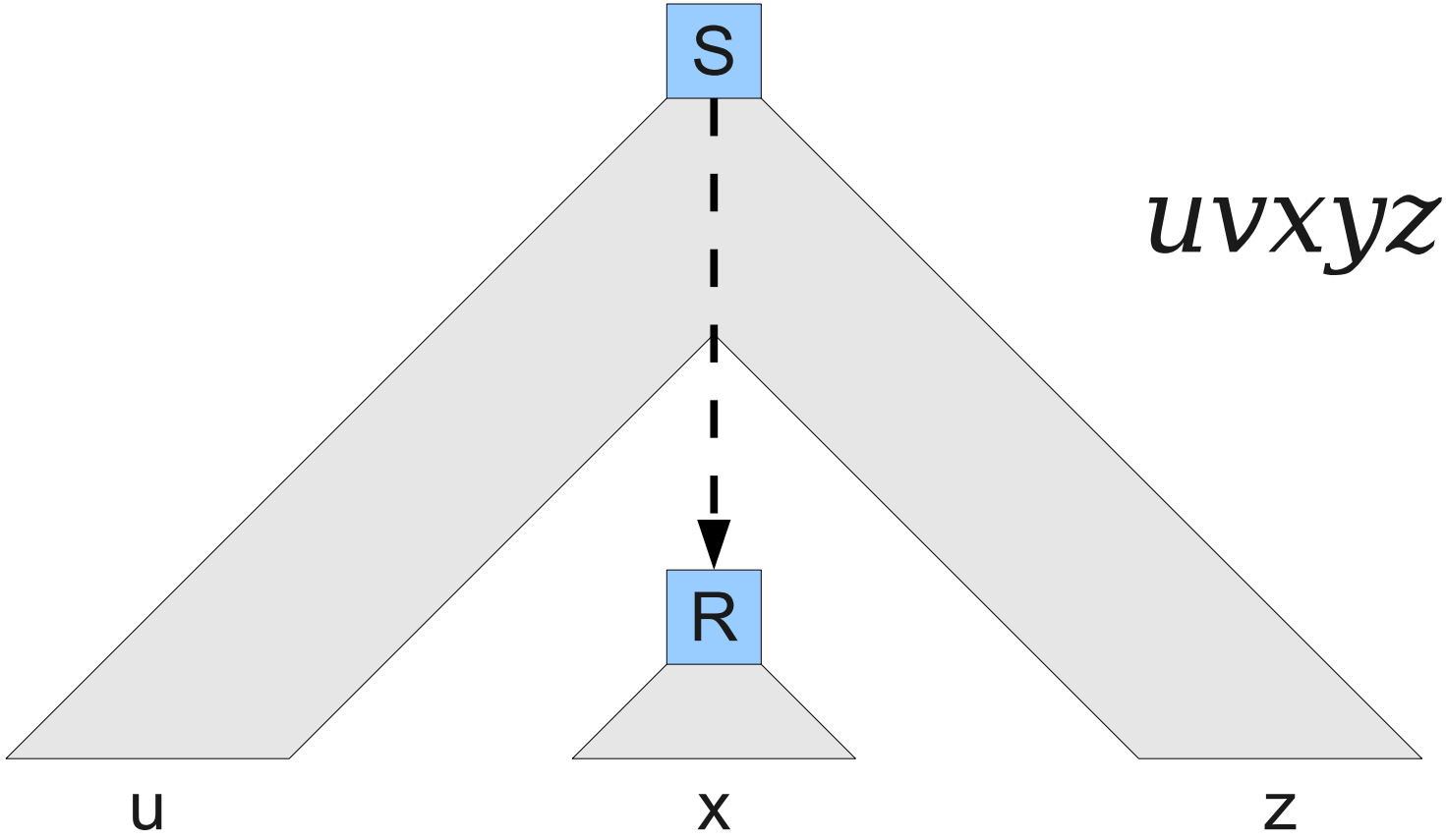
$$uv^2xy^2z \in L$$



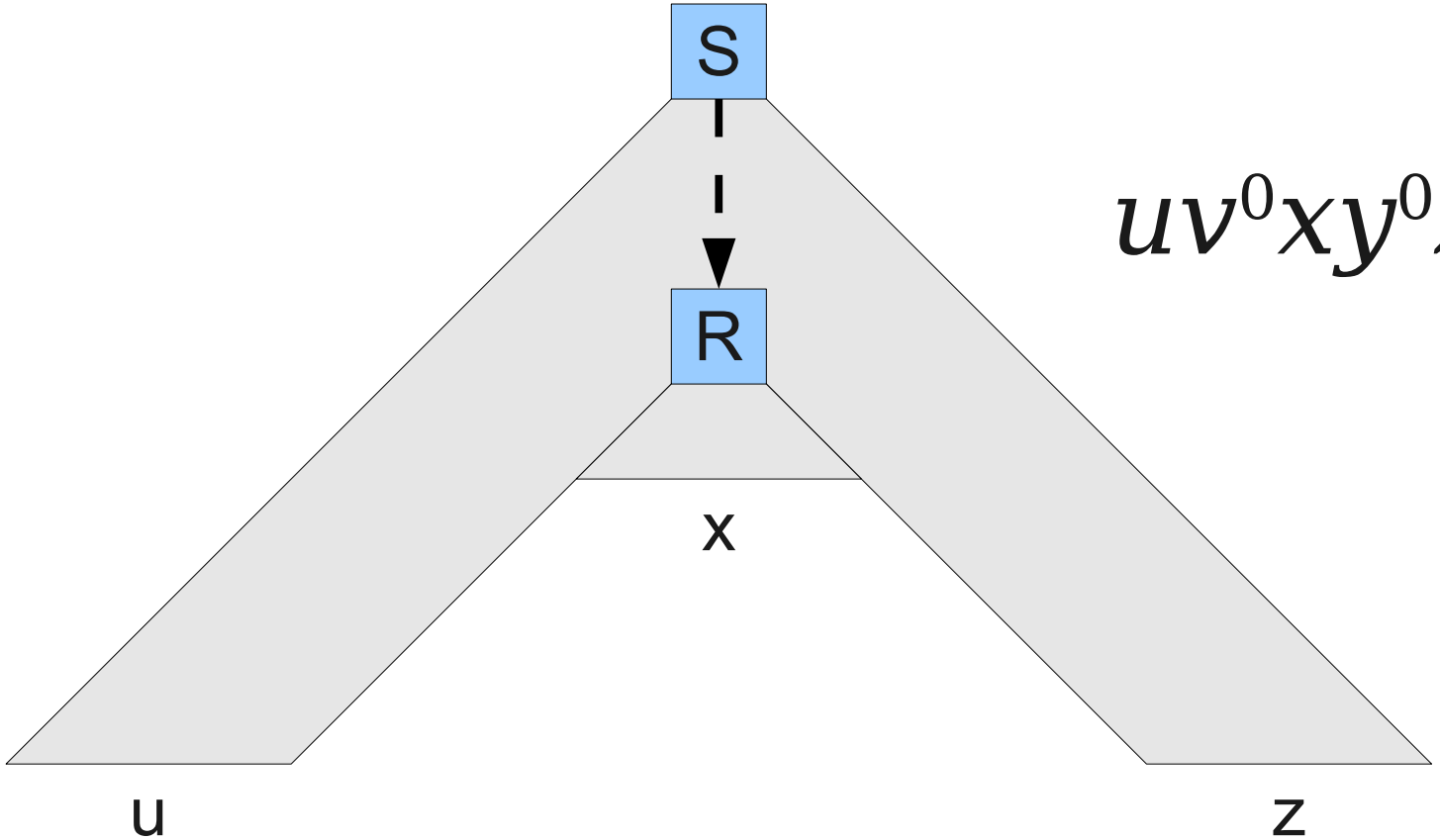
$$uv^3xy^3z \in L$$



$uvxyz \in L$



$$uvxyz \in L$$



$$uv^0xy^0z \in L$$

The Pumping Lemma for CFLs

For any context-free language L,

There exists a positive natural number n such that

For any $w \in L$ with $|w| \geq n$,

There exists strings u, v, x, y, z such that

For any natural number i,

$w = uvxyz$,	w can be broken into five pieces,
$|vxy| \leq n$,	where the middle three pieces aren't too long,
$|vy| > 0$	where the 2nd and 4th pieces aren't both empty, and
$uv^ixy^iz \in L$	where the 2nd and 4th pieces can be replicated 0 or more times

The Pumping Lemma for CFLs

For any context-free language L,

There exists a positive natural number n such that

For any $w \in L$ with $|w| \geq n$,

There exists strings u, v, x, y, z such that

For any natural number i,

$w = uvxyz$, w can be broken into five pieces,

$|vxy| \leq n$, where the middle three pieces aren't too long,

$|vy| > 0$ where the 2nd and 4th pieces aren't both empty, and

$uv^ixy^iz \in L$ where the 2nd and 4th pieces can be replicated 0 or more times

Note that we pump both v and y at the same time, not just one or the other.

The Pumping Lemma for CFLs

For any context-free language L,

There exists a positive natural number n such that

For any $w \in L$ with $|w| \geq n$,

There exists strings u, v, x, y, z such that

For any natural number i,

$w = uvxyz$, w can be broken into five pieces,

$|vxy| \leq n$, where the middle three pieces aren't too long,

$|vy| > 0$ where the 2nd and 4th pieces aren't both empty, and

$uv^ixy^iz \in L$ where the 2nd and 4th pieces can be replicated 0 or more times

The two strings to pump, collectively, cannot be too long.

They also must be close to one another.

The Pumping Lemma for CFLs

For any context-free language L,

There exists a positive natural number n such that

For any $w \in L$ with $|w| \geq n$,

There exists strings u, v, x, y, z such that

For any natural number i,

$w = uvxyz$, w can be brok

$|vxy| \leq n$, where the middle three pieces aren't too long,

$|vy| > 0$ where the 2nd and 4th pieces aren't both empty, and

$uv^ixy^iz \in L$ where the 2nd and 4th pieces can be replicated 0 or more times

The pumping length is <u>not</u> simple; see Sipser for details.

The Pumping Lemma Game

$L = \{w \in \{0,1,2\}^* \mid w \text{ has the same number of } 0\text{s}, 1\text{s}, 2\text{s}\}$

ADVERSARY	YOU
Maliciously choose pumping length n.	Cleverly choose a string $w \in L$, $\lvert w \rvert \geq n$
Maliciously split $w = uvxyz$, with $\lvert vy \rvert > 0$ and $\lvert vxy \rvert \leq n$	Cleverly choose k such that $uv^kxy^kz \notin L$
Grrr! Aaaargh!	

The Pumping Lemma Game

$L = \{w \in \{0,1,2\}^* \mid w \text{ has the same number of } 0\text{s}, 1\text{s}, 2\text{s}\}$

ADVERSARY	YOU
Maliciously choose pumping length n.	Cleverly choose a string $w \in L$, $\lvert w \rvert \geq n$
Maliciously split $w = uvxyz$, with $\lvert vy \rvert > 0$ and $\lvert vxy \rvert \leq n$	Cleverly choose k such that $uv^kxy^kz \notin L$
Grrr! Aaaargh!	

$0^n1^n2^n$

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Proofs using the pumping lemma for CFLs tend to be much harder than those for regular languages because there is no restriction on where in the string the portion that can be pumped can be. The string to pump must be very carefully constructed.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$).

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{0,1,2\}^* \mid w$ has the same # of 0s, 1s, 2s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = 0^n1^n2^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in 0^n, 1^n, or 2^n. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of 0 or 1 or 2 than the other two symbols.

Case 2: vxy either consists of 0s and 1s or of 1s and 2s (it cannot consist of all three symbols, because $|vxy| \leq n$).

Note how we chose w so that vxy can't span all three groups of symbols. This makes it impossible to pump all three groups at once.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no $\mathtt{2}$s in it, $uv^2xy^2z \notin L$ since it contains more $\mathtt{0}$s or $\mathtt{1}$s than $\mathtt{2}$s.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no $\mathtt{2}$s in it, $uv^2xy^2z \notin L$ since it contains more $\mathtt{0}$s or $\mathtt{1}$s than $\mathtt{2}$s. Similarly, if vxy has no $\mathtt{0}$s in it $uv^2xy^2z \notin L$ because it contains more $\mathtt{1}$s or $\mathtt{2}$s than $\mathtt{0}$s.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no $\mathtt{2}$s in it, $uv^2xy^2z \notin L$ since it contains more $\mathtt{0}$s or $\mathtt{1}$s than $\mathtt{2}$s. Similarly, if vxy has no $\mathtt{0}$s in it $uv^2xy^2z \notin L$ because it contains more $\mathtt{1}$s or $\mathtt{2}$s than $\mathtt{0}$s.

In either case, we contradict the pumping lemma.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no $\mathtt{2}$s in it, $uv^2xy^2z \notin L$ since it contains more $\mathtt{0}$s or $\mathtt{1}$s than $\mathtt{2}$s. Similarly, if vxy has no $\mathtt{0}$s in it $uv^2xy^2z \notin L$ because it contains more $\mathtt{1}$s or $\mathtt{2}$s than $\mathtt{0}$s.

In either case, we contradict the pumping lemma. Thus our assumption must have been wrong, so L is not a CFL.

For any context-free language L,
There exists a positive natural number n such that
For any $w \in L$ with $|w| \geq n$,
There exists strings u, v, x, y, z such that
For any natural number i,
$w = uvxyz$,
$|vxy| \leq n$,
$|vy| > 0$
$uv^ixy^iz \in L$

Theorem: $L = \{w \in \{\mathtt{0},\mathtt{1},\mathtt{2}\}^* \mid w$ has the same # of $\mathtt{0}$s, $\mathtt{1}$s, $\mathtt{2}$s$\}$ is not a CFL.

Proof: By contradiction; assume L is a CFL. Let n be the pumping length guaranteed by the pumping lemma. Let $w = \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n$. Thus $w \in L$ and $|w| = 3n \geq n$. Therefore we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any $i \in \mathbb{N}$, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in $\mathtt{0}^n$, $\mathtt{1}^n$, or $\mathtt{2}^n$. In that case, the string $uv^2xy^2z \notin L$, because this string has more copies of $\mathtt{0}$ or $\mathtt{1}$ or $\mathtt{2}$ than the other two symbols.

Case 2: vxy either consists of $\mathtt{0}$s and $\mathtt{1}$s or of $\mathtt{1}$s and $\mathtt{2}$s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no $\mathtt{2}$s in it, $uv^2xy^2z \notin L$ since it contains more $\mathtt{0}$s or $\mathtt{1}$s than $\mathtt{2}$s. Similarly, if vxy has no $\mathtt{0}$s in it $uv^2xy^2z \notin L$ because it contains more $\mathtt{1}$s or $\mathtt{2}$s than $\mathtt{0}$s.

In either case, we contradict the pumping lemma. Thus our assumption must have been wrong, so L is not a CFL. ■

Using the Pumping Lemma

- Keep the following in mind when using the context-free pumping lemma when $w = uvxyz$:
  - Both v and y must be pumped at the same time.
  - v and y need not be contiguous in the string.
  - One of v and y may be empty.
  - vxy may be anywhere in the string.
- I **strongly suggest** reading through Sipser to get a better sense for how these proofs work.

Non-CFLs

- Regular languages cannot count once:

  $\{ \mathtt{0}^n\mathtt{1}^n \mid n \in \mathbb{N} \}$ is not regular.

- CFLs cannot count *twice*:
  - $\{ \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n \mid n \in \mathbb{N} \}$ is not context-free.
- A finite number of states cannot count arbitrarily high.
- With a single stack and finite states, cannot track two arbitrary quantities.

(Non) Closure Properties of CFLs

(Non) Closure Properties of CFLs

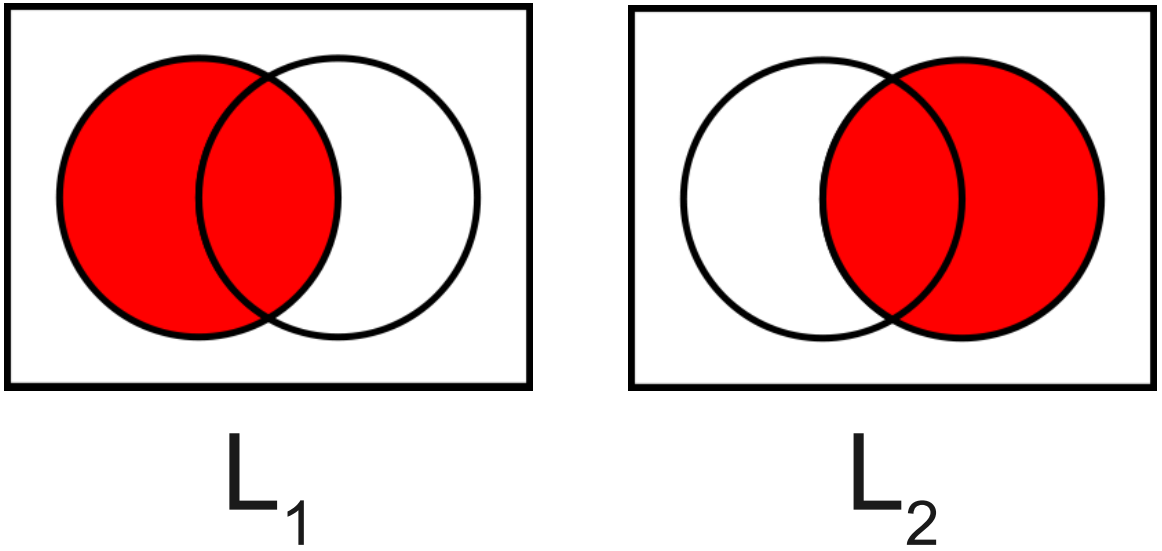
- Now that we have a single non-context-free language, we can prove that CFLs are not closed under certain operations.
- Let $L_1 = \{ \mathtt{0}^n\mathtt{1}^n\mathtt{2}^m \mid n, m \in \mathbb{N} \}$
- Let $L_2 = \{ \mathtt{0}^m\mathtt{1}^n\mathtt{2}^n \mid n, m \in \mathbb{N} \}$
- Both of these languages are context-free.
  - Can either find an explicit CFG, or note that these languages are the concatenation of two CFLs.
- But $L_1 \cap L_2 = \{ \mathtt{0}^n\mathtt{1}^n\mathtt{2}^n \mid n \in \mathbb{N} \}$, which is not a CFL.
- **Context-free languages are not closed under intersection**.

(Non) Closure under Complement

- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.

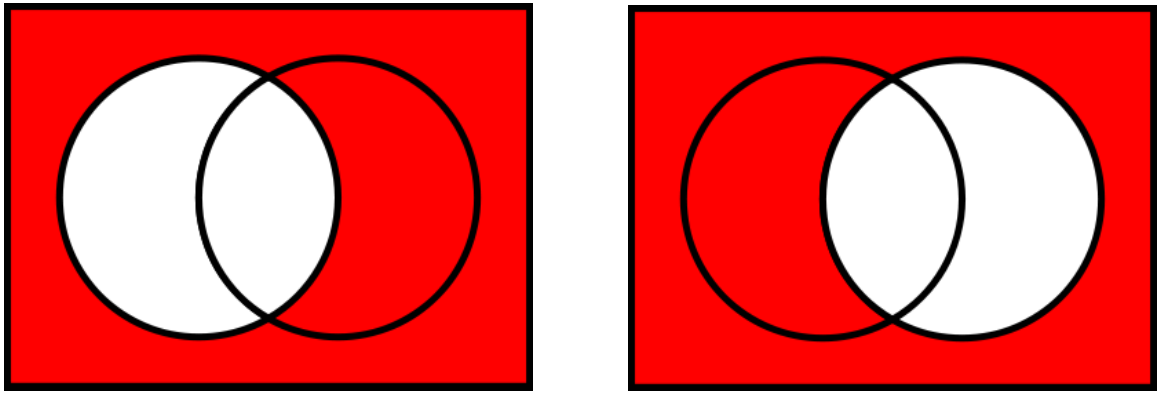
(Non) Closure under Complement

- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



(Non) Closure under Complement

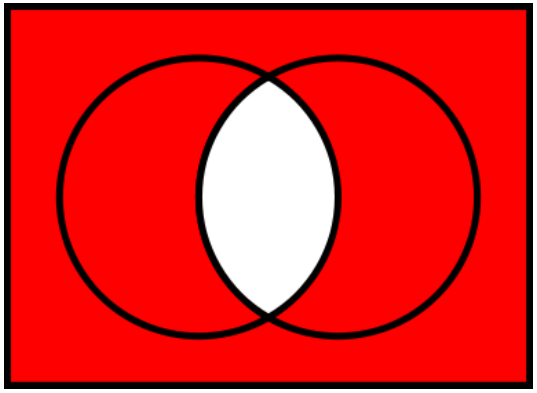
- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



$\overline{L}_1$ $\overline{L}_2$

(Non) Closure under Complement

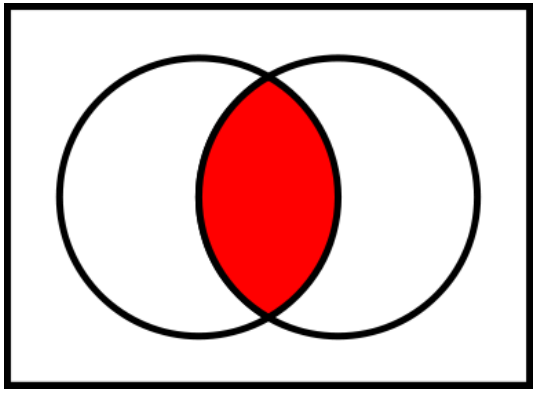
- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



$\overline{L}_1 \cup \overline{L}_2$

(Non) Closure under Complement

- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



$$\overline{\overline{L_1} \cup \overline{L_2}}$$

(Non) Closure under Subtraction

- ***Theorem:*** If L_1 and L_2 are regular, $L_1 - L_2$ is regular as well.
- However, if L_1 and L_2 are context-free, $L_1 - L_2$ may not be context-free.
- Intuition: We can construct the complement from the difference.
  - Σ^* is context-free because it is regular.
  - But $\Sigma^* - L = \overline{L}$, which may not be context-free.

Summary of CFLs

- CFLs are strictly more powerful than the regular languages.
- CFLs can be described by CFGs (generation) or PDAs (recognition).
- CFGs encompass two classes of languages – deterministic and nondeterministic CFLs.
- Context-free languages have a pumping lemma just as regular languages do.

Interlude for Announcements

Problem Session

- Weekly problem session meets tonight at 7PM in 380-380X.
  - Covers CFLs and their limits.
- Optional, but highly recommended!

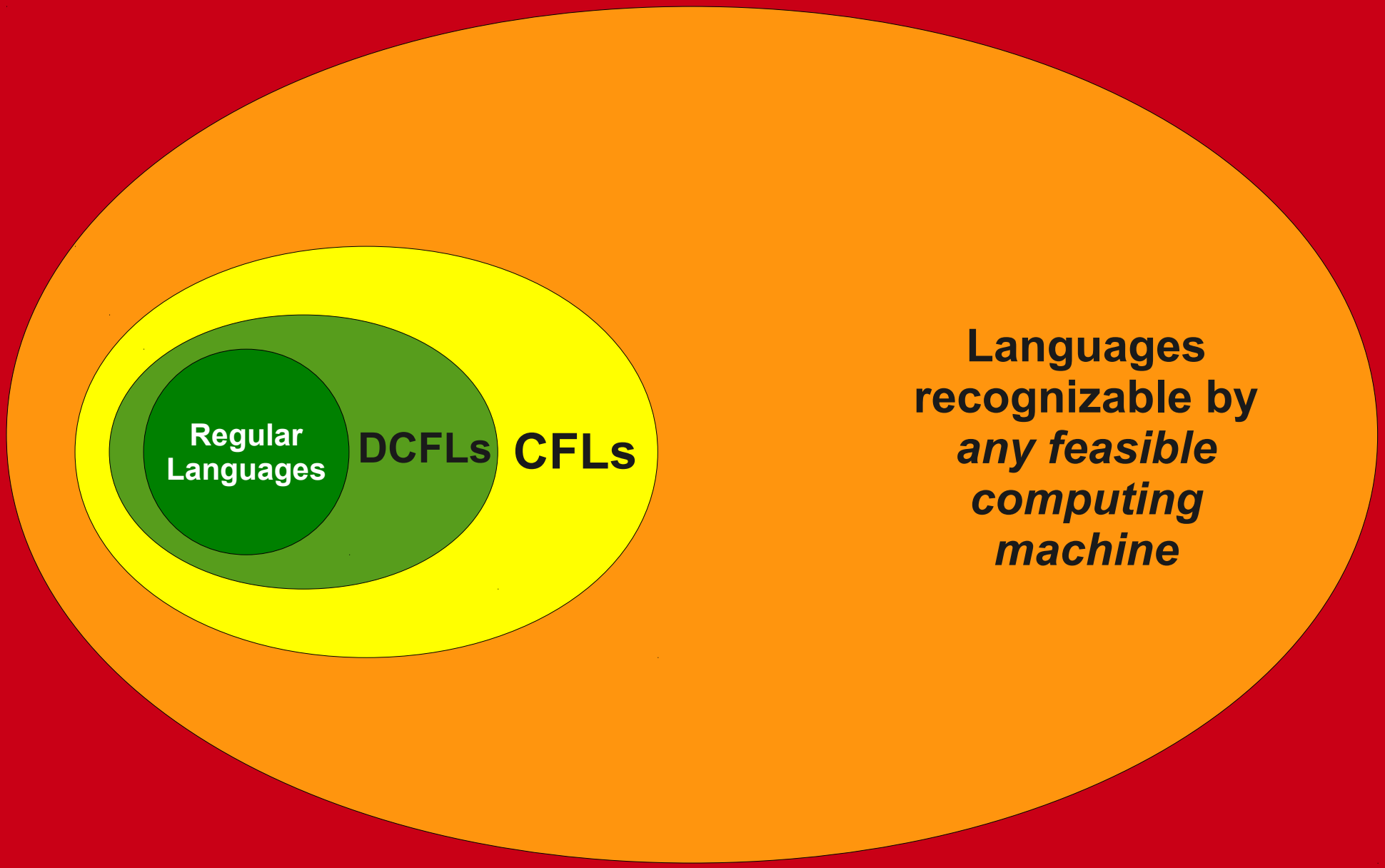
Midterm and Problem Set 4 Graded

Will be distributed at end of lecture.
After that, pick up at my office (Gates 178).

Beyond CFLs

Computability Theory

- **Finite automata** represent computers with bounded memory.
  - They accept precisely the **regular languages**.
- **Pushdown automata** represent computers with a weak infinite memory.
  - They accept precisely the **context-free languages**.
- Regular and context-free languages are comparatively weak.



All Languages

That same drawing, to scale.

All Languages

Defining Computability

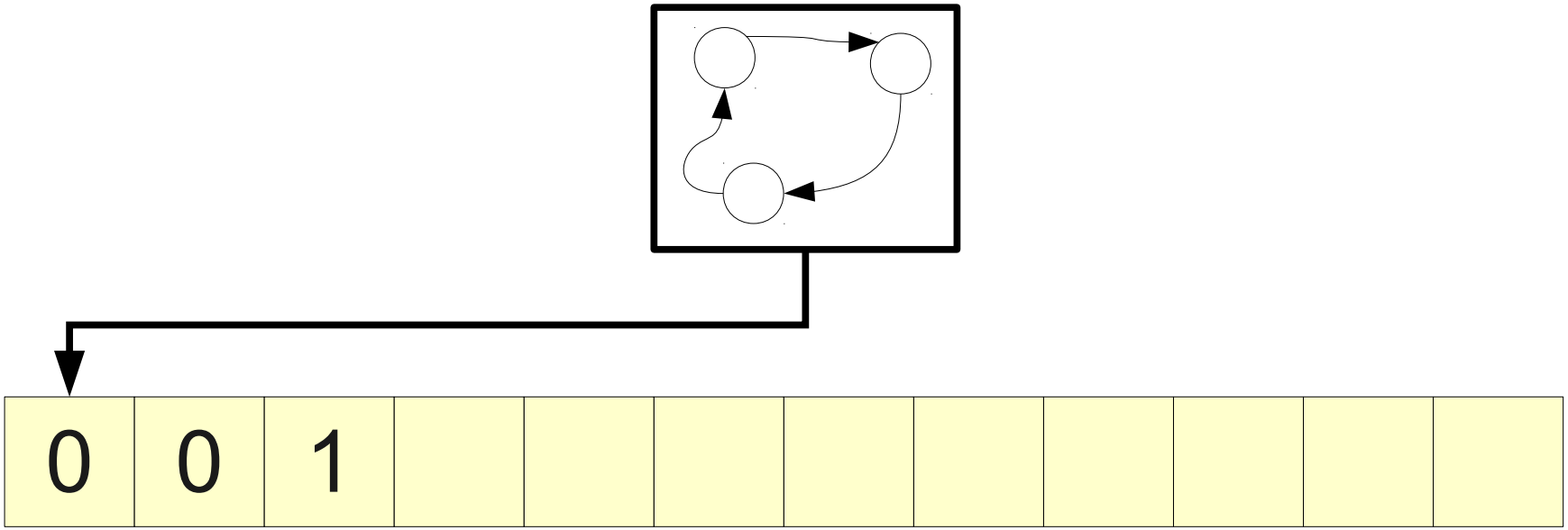
- In order to talk about what languages we could ever hope to recognize with a computer, we need to formalize our model of computation with an automaton.
- The standard automaton for this job is the **Turing machine**, named after Alan Turing, the "Father of Computer Science."

A Better Memory Device

- The pushdown automaton used a (potentially infinite) stack as its memory device.
- This severely limits how the memory can be used:
  - Accessing old data only possible after discarding old data.
  - Can't keep track of multiple unbounded quantities.

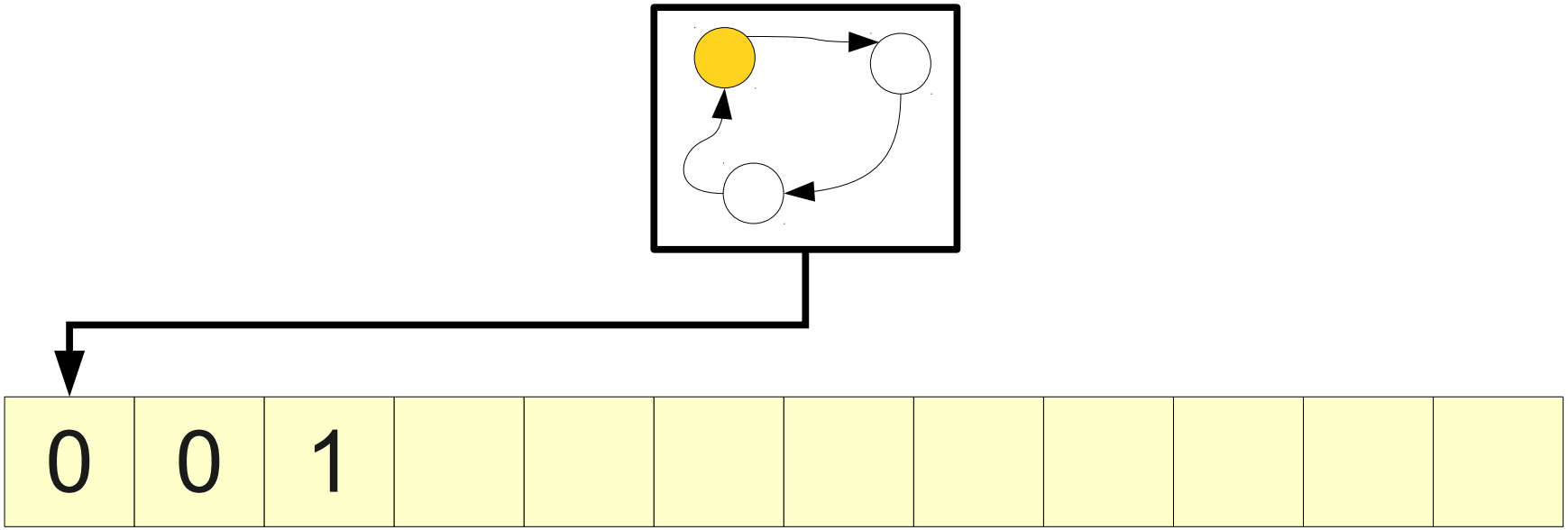
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



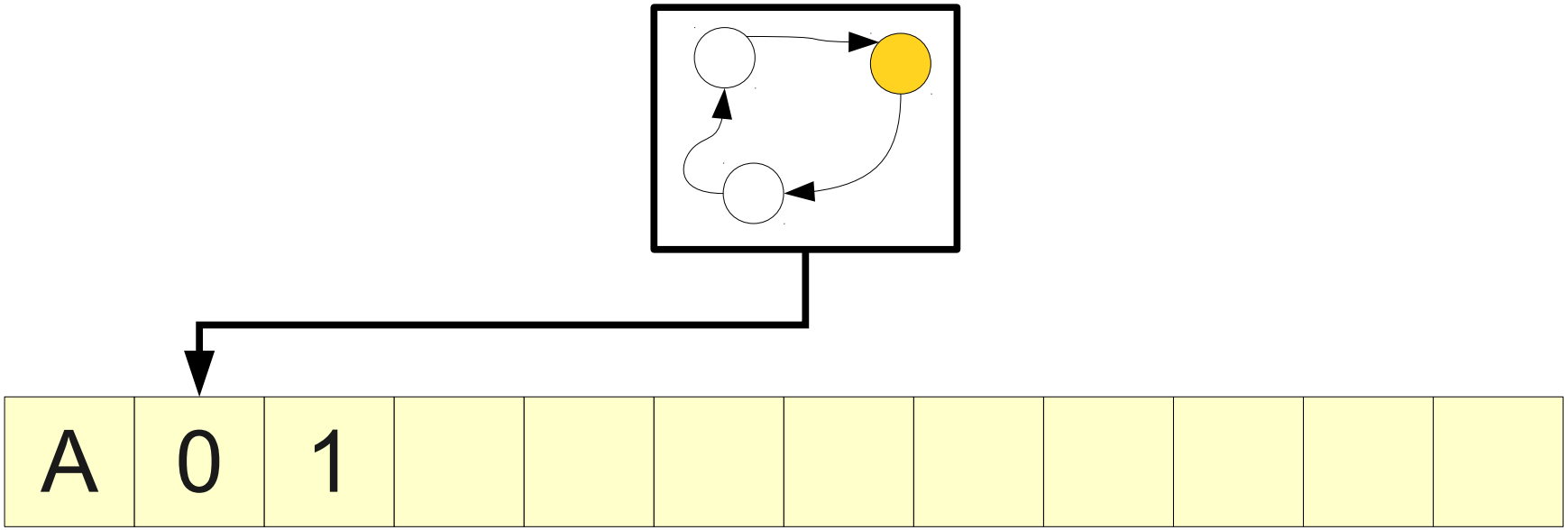
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



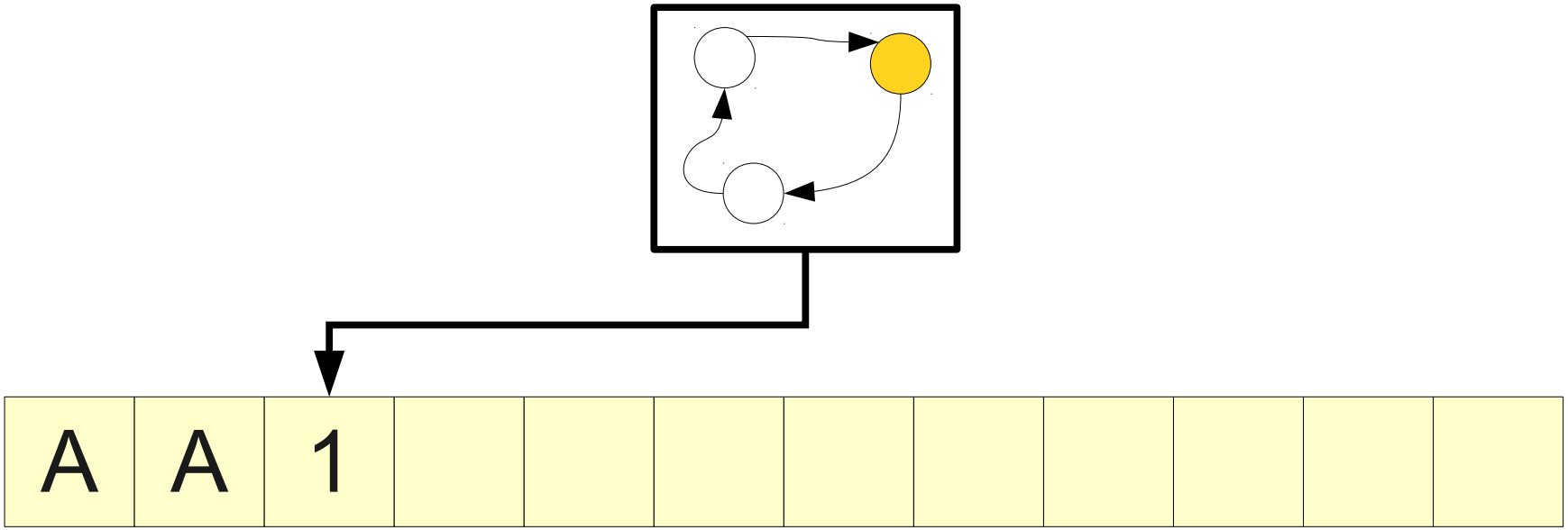
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



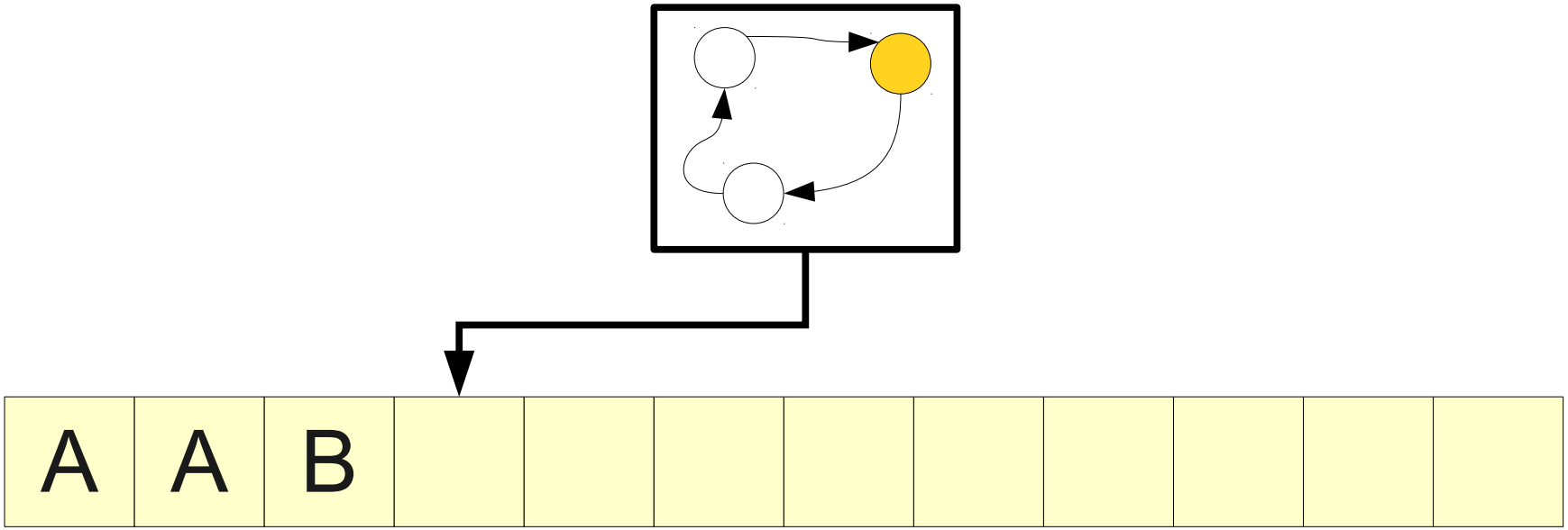
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



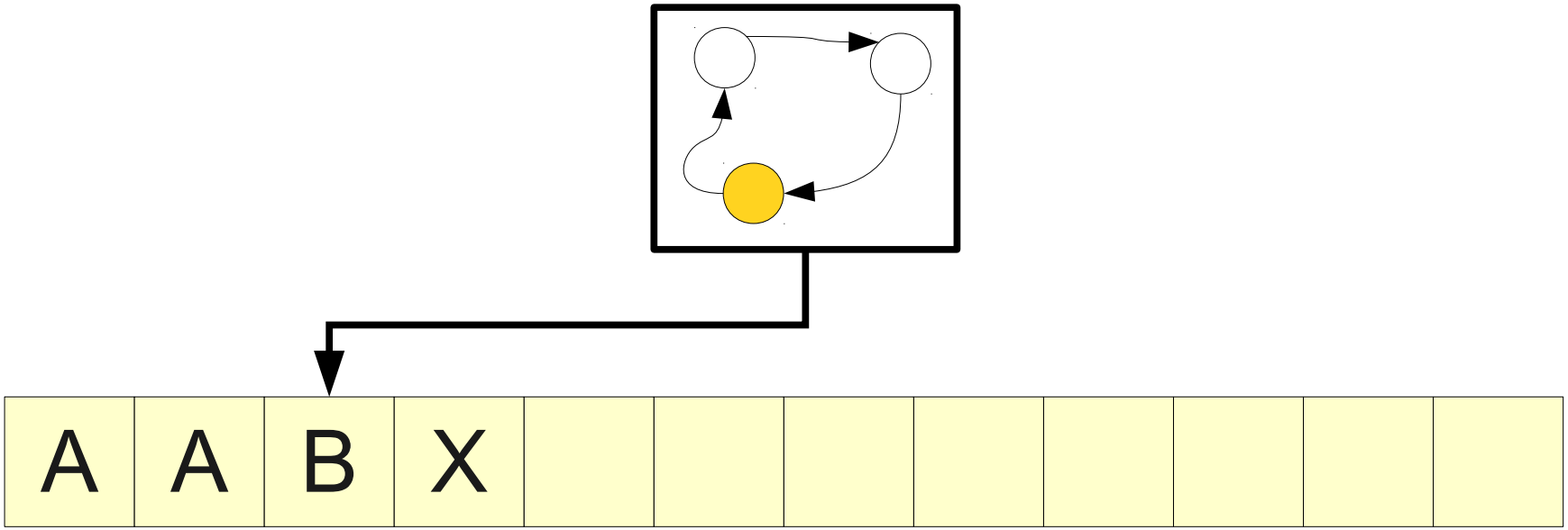
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



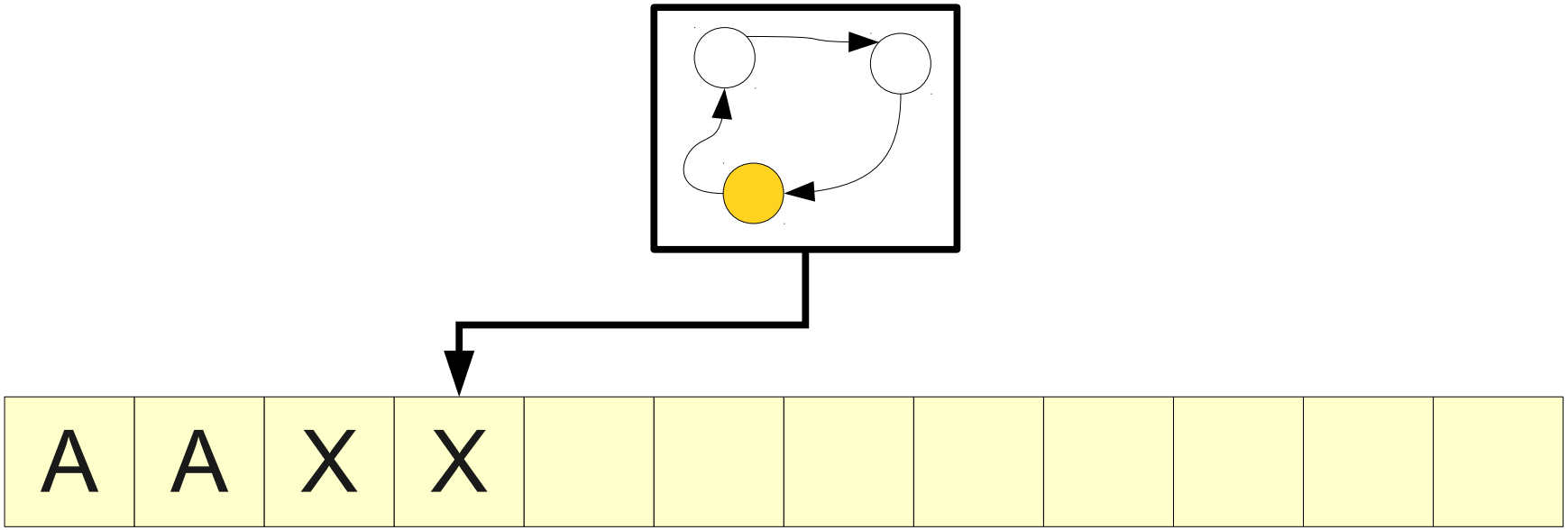
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



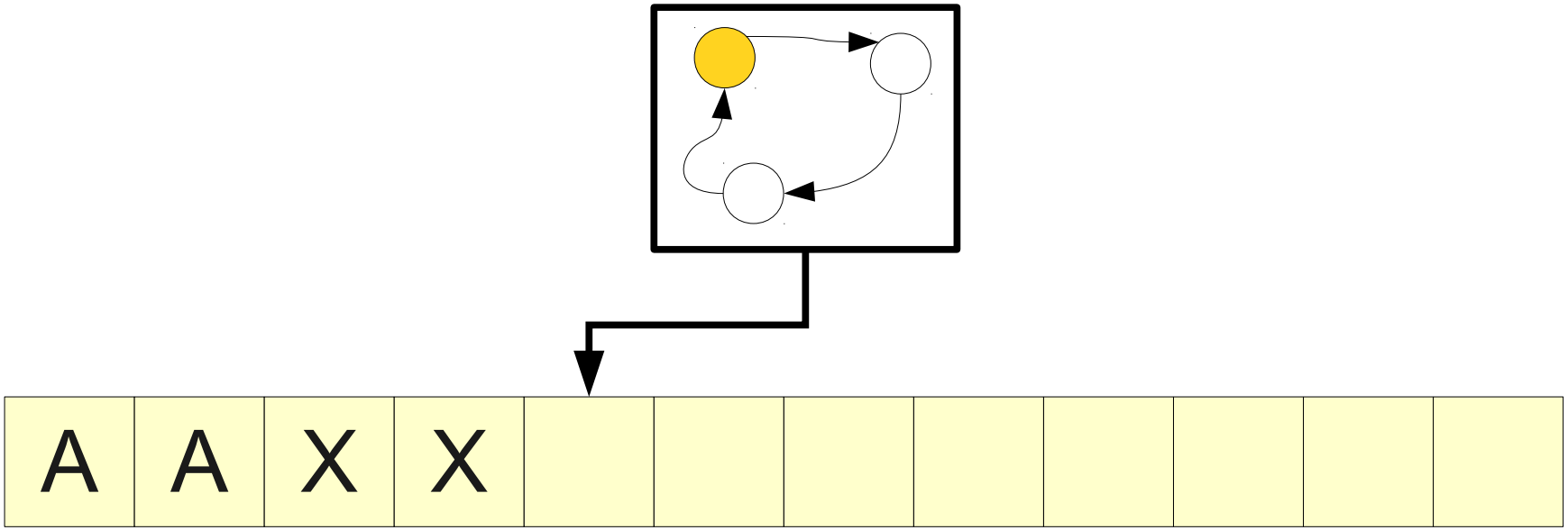
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



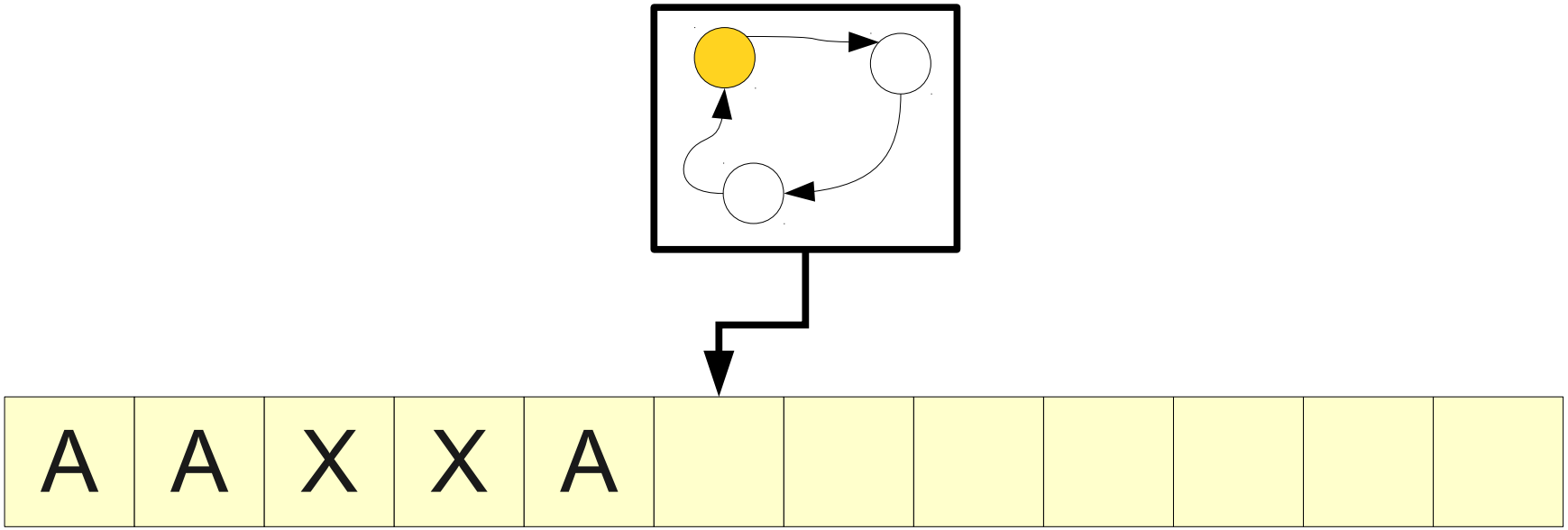
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



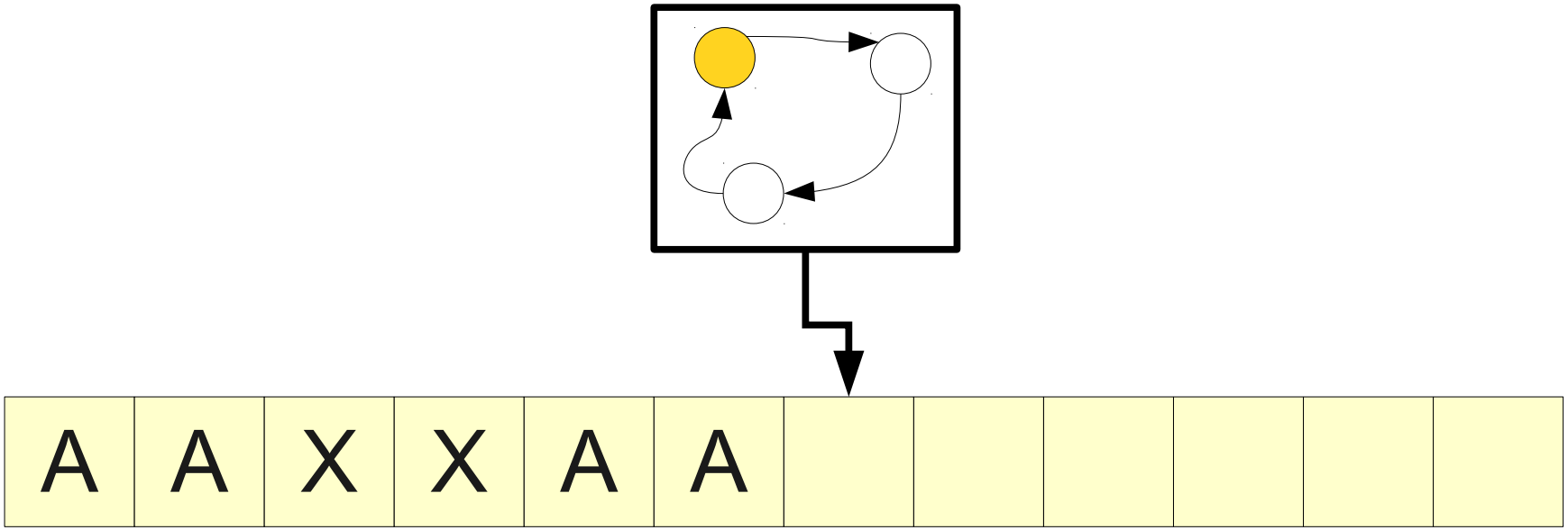
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



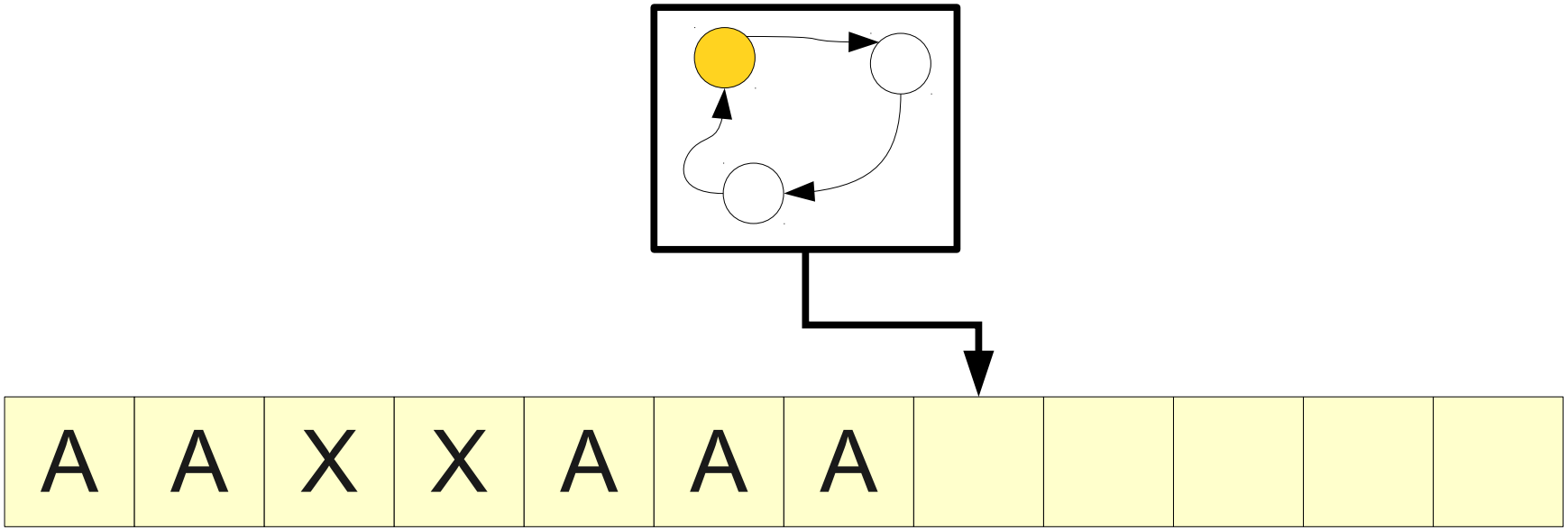
A Better Memory Device

- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



A Better Memory Device

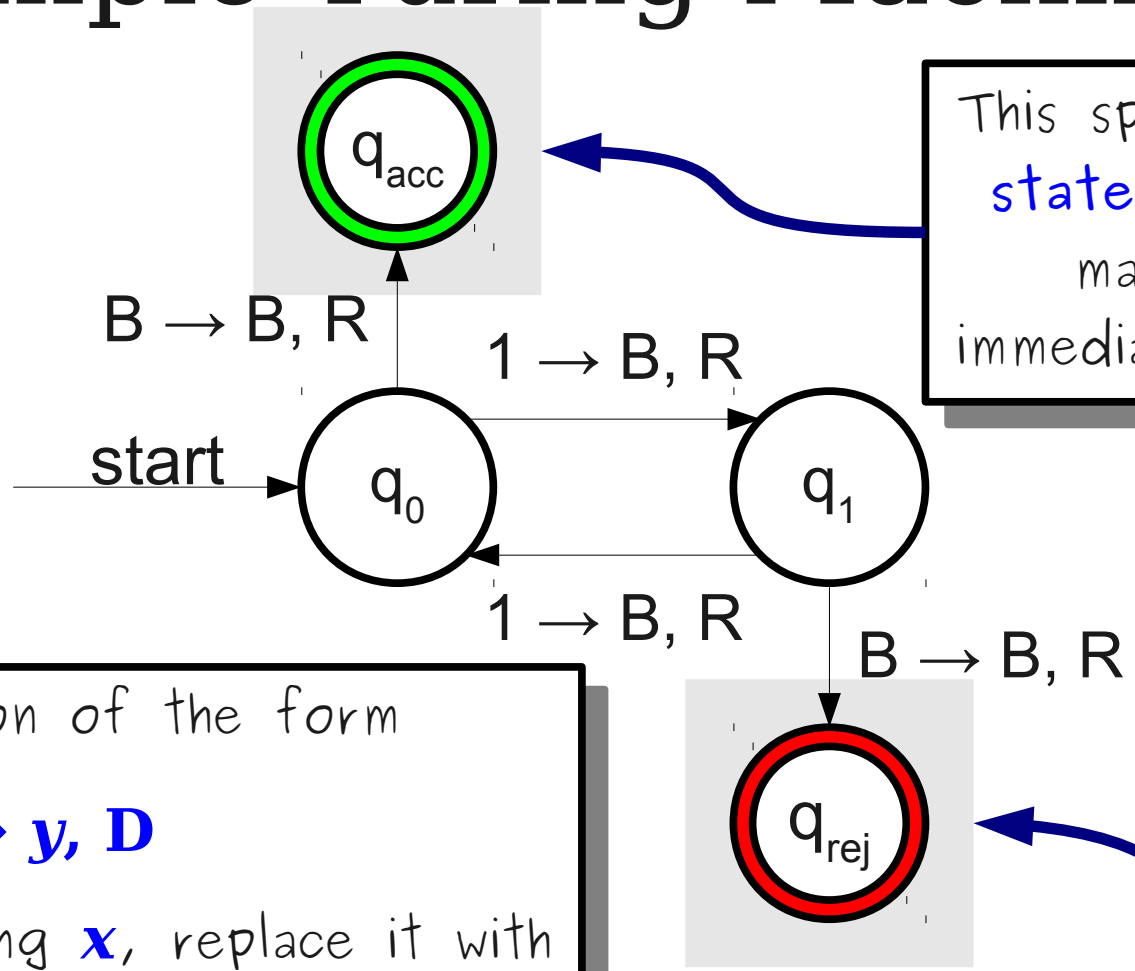
- A **Turing machine** is a finite automaton equipped with an **infinite tape** as its memory.
- The tape begins with the input to the machine written on it, followed by infinitely many blank cells.
- The machine has a **tape head** that can read and write a single memory cell at a time.



The Turing Machine

- A Turing machine consists of three parts:
  - A **finite-state control** used to determine which actions to take,
  - an **infinite tape** serving as both input and scratch space, and
  - a **tape head** that can read and write the tape and move left or right.
- At each step, the Turing machine
  - Replaces the contents of the current cell with a new symbol (which could optionally be the same symbol as before),
  - Changes state, and
  - Moves the tape head to the left or to the right.

A Simple Turing Machine



This special **accept state** causes the machine to immediately accept.

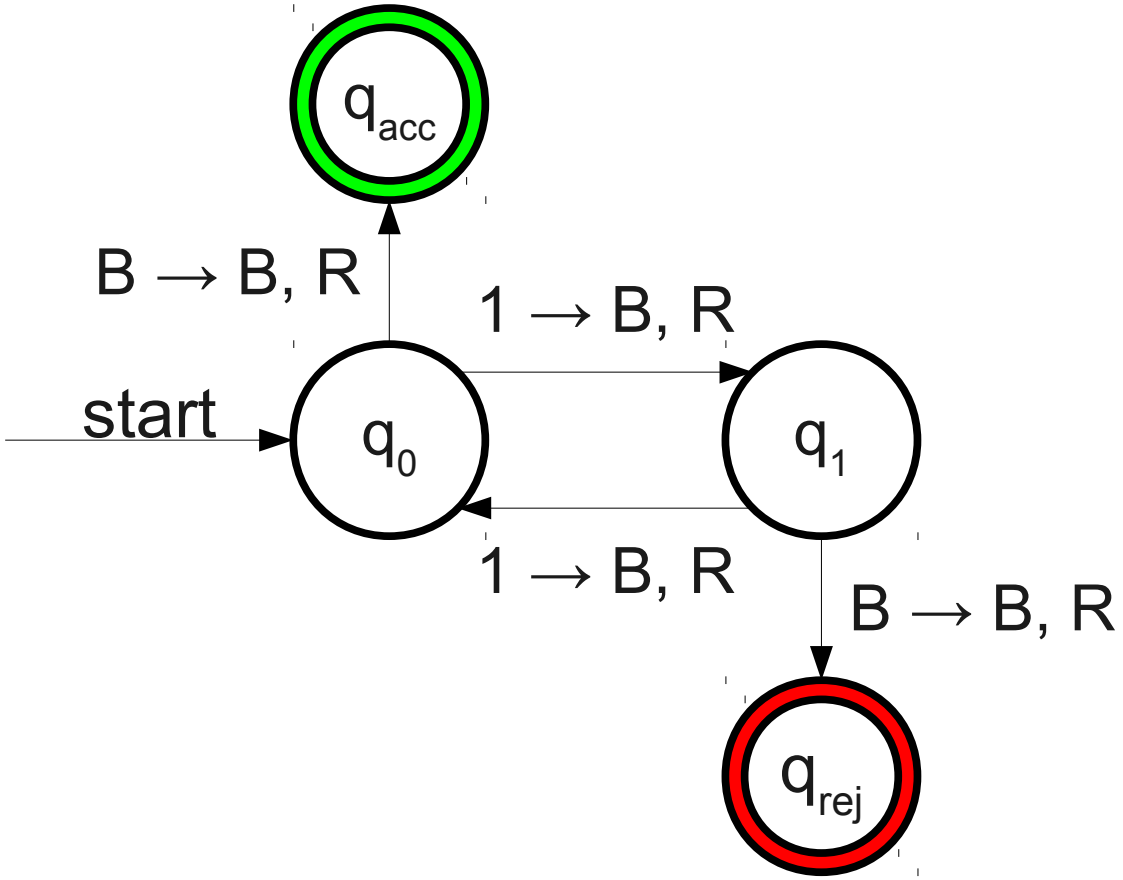
Each transition of the form

$x \rightarrow y$, D

means "upon reading **x**, replace it with symbol **y** and move the tape head in direction **D** (which is either **L** or **R**). The letter **B** represents the **blank symbol**.

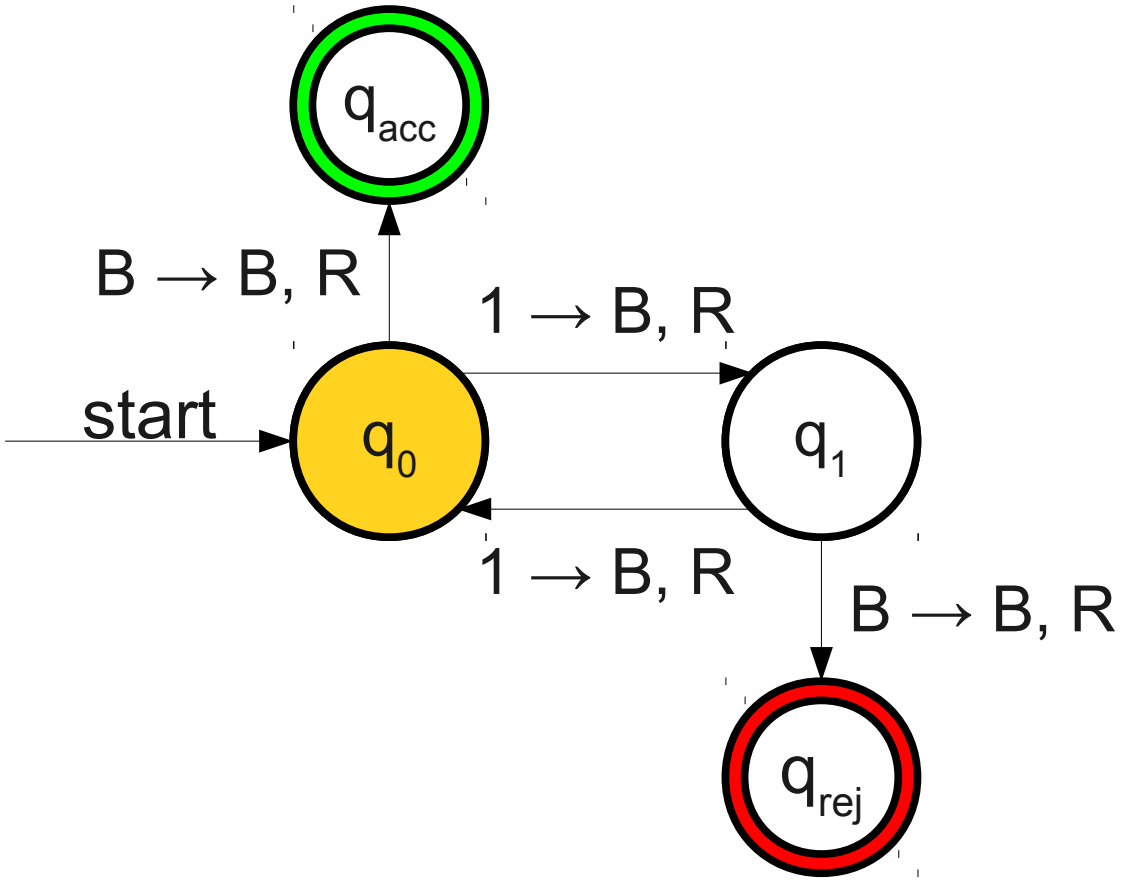
This special **reject state** causes the machine to immediately reject.

A Simple Turing Machine



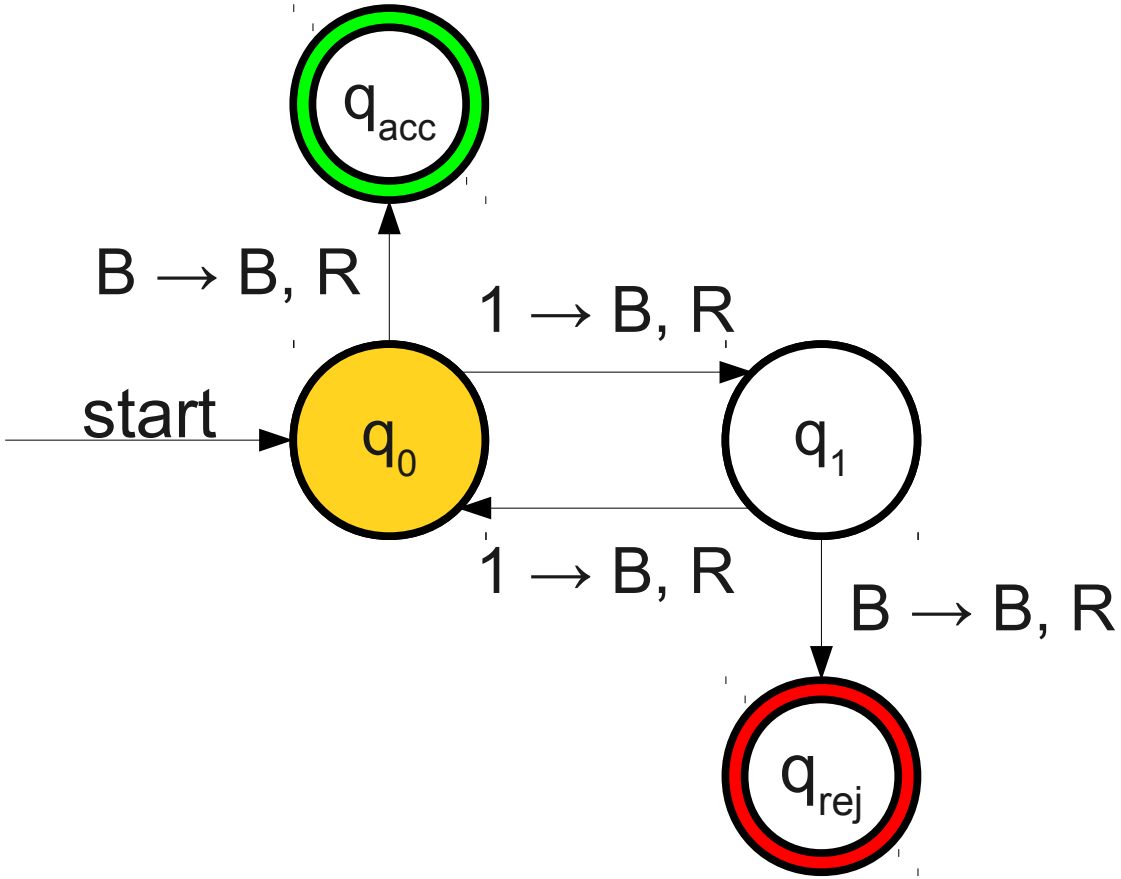
1	1	1	1								

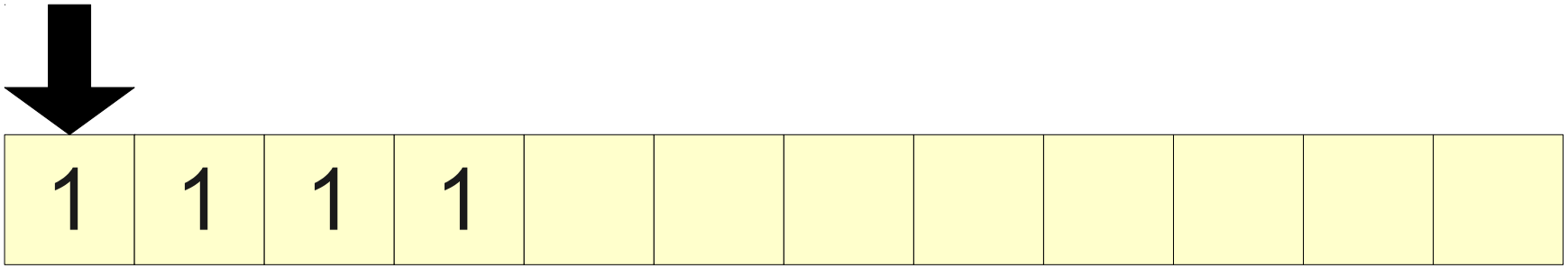
A Simple Turing Machine



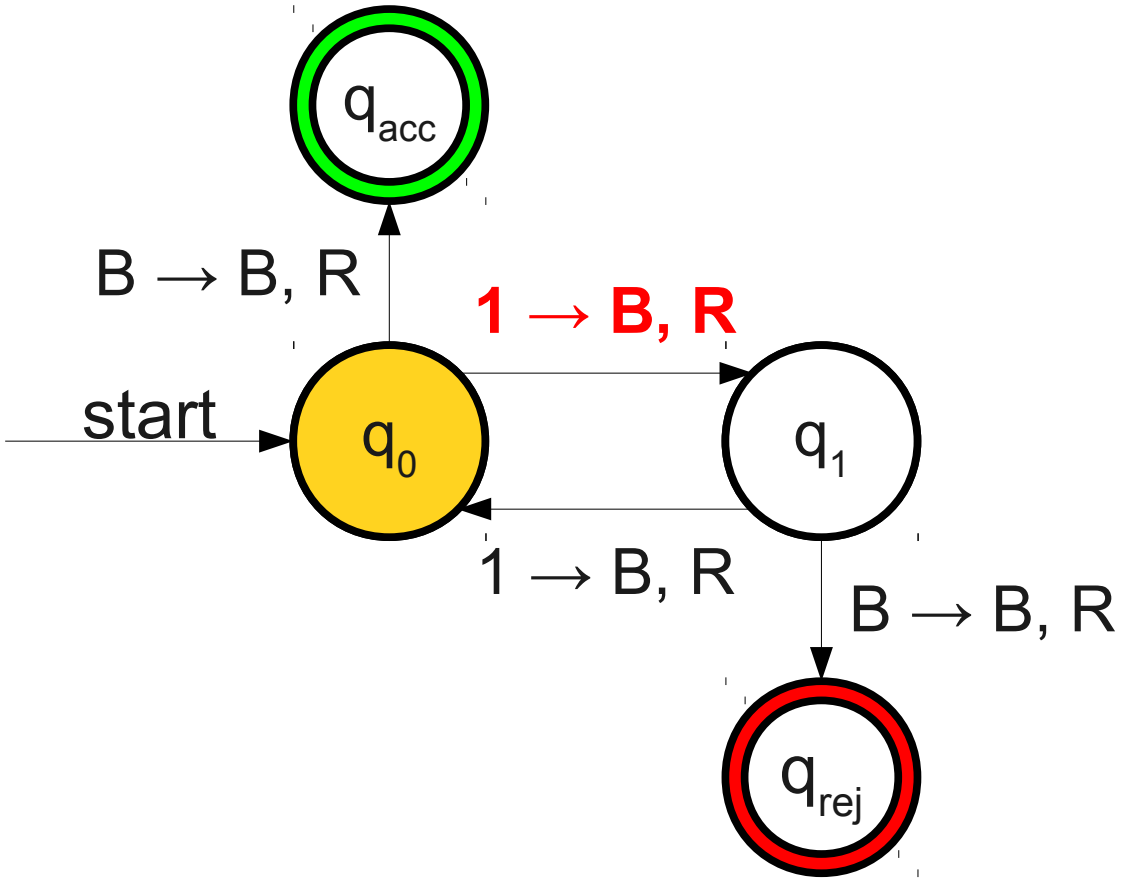
1	1	1	1								

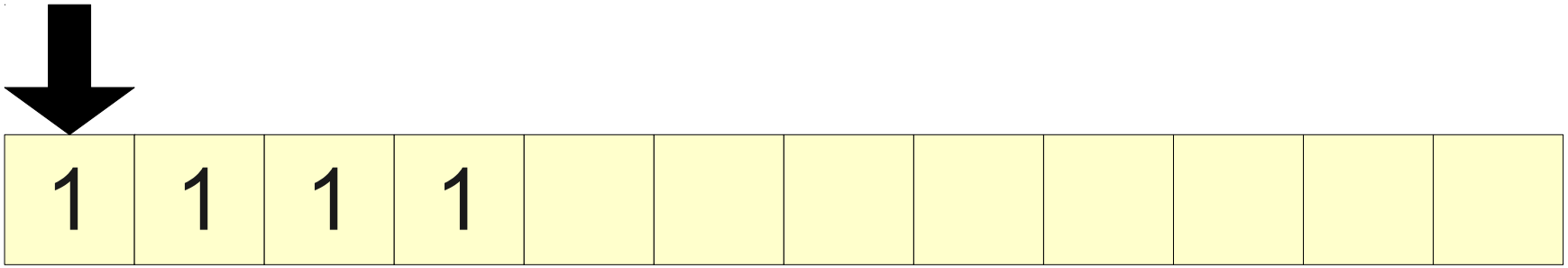
A Simple Turing Machine



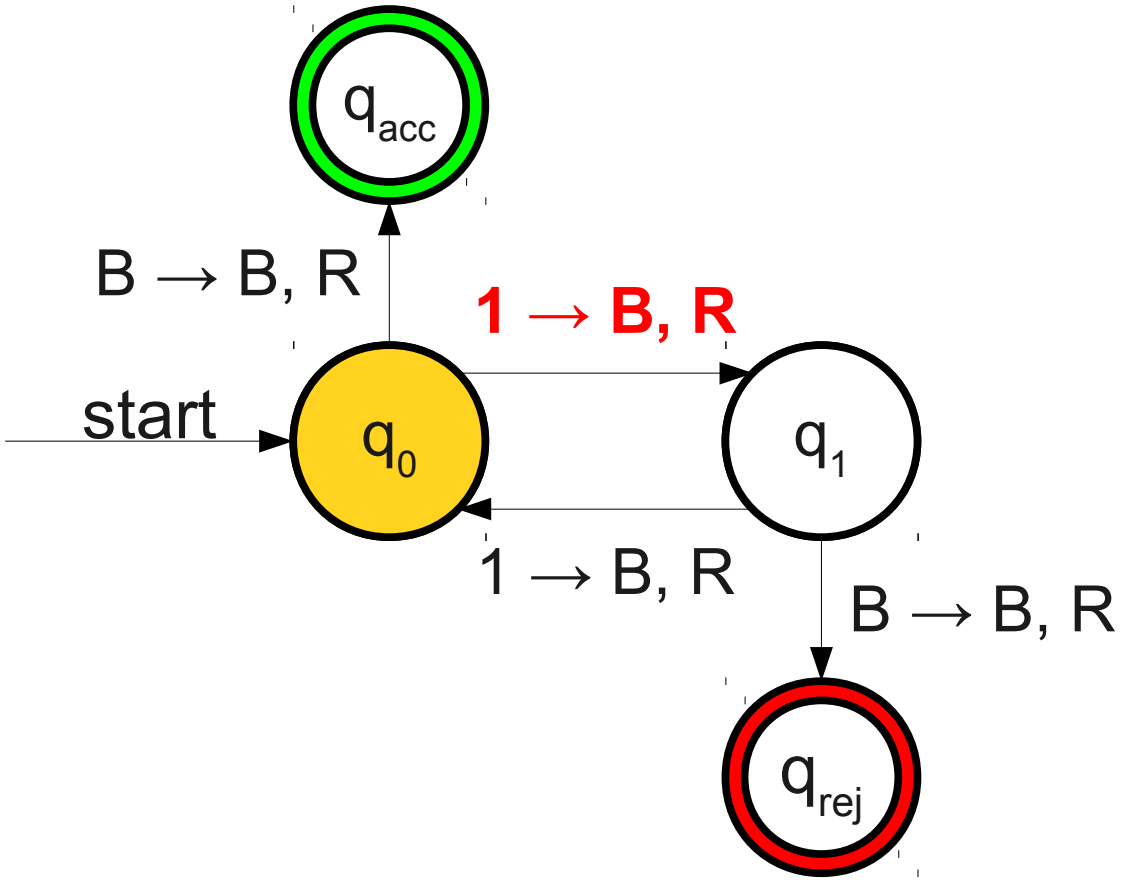


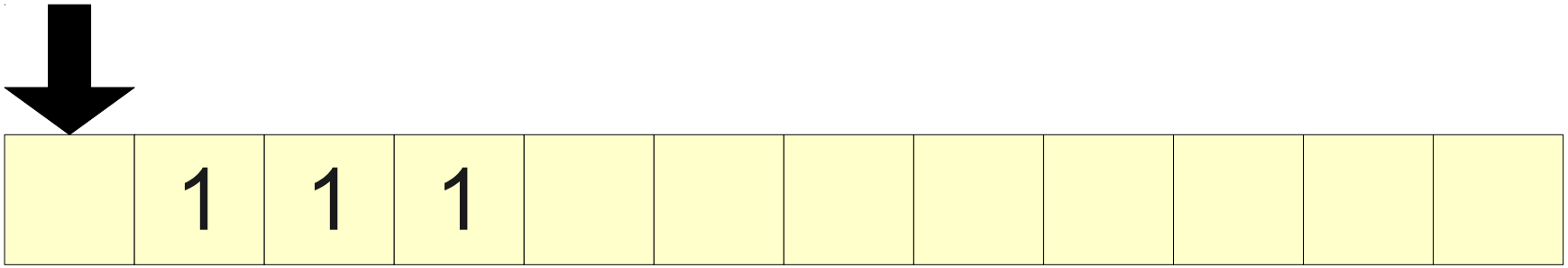
A Simple Turing Machine



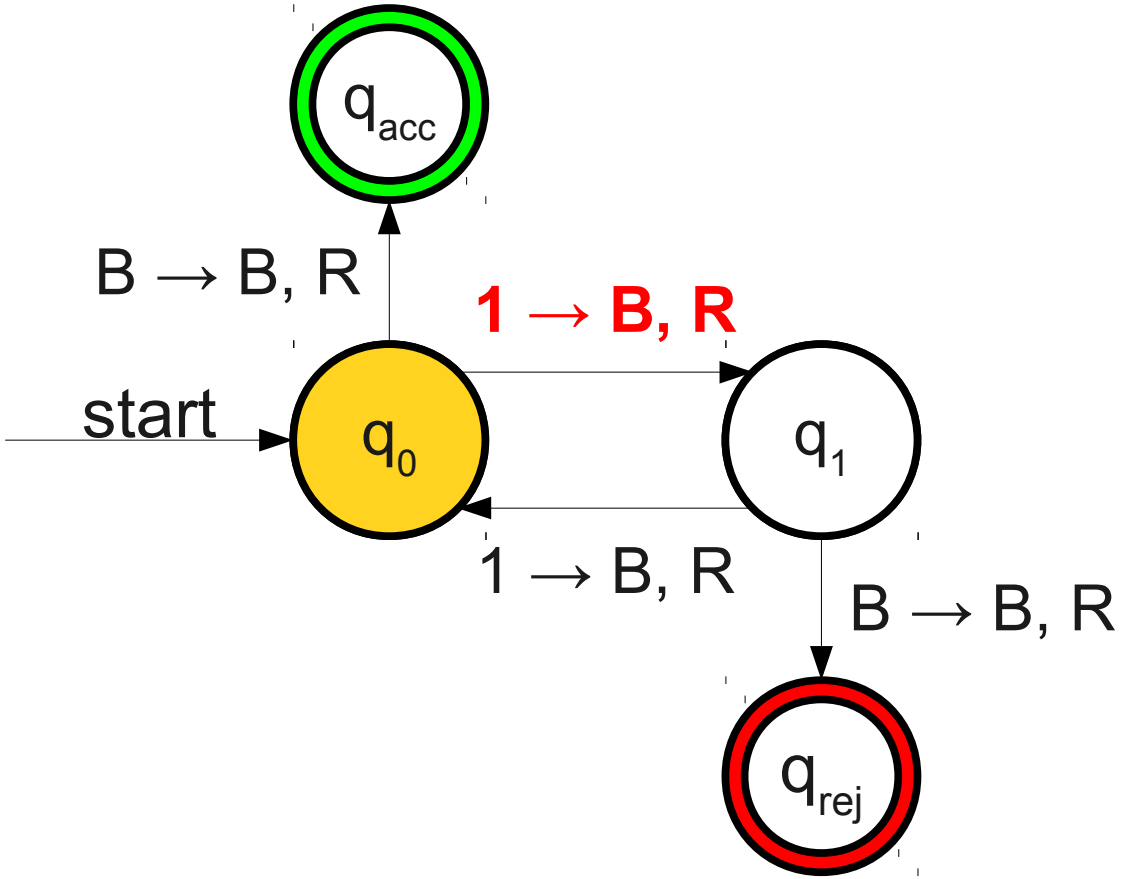


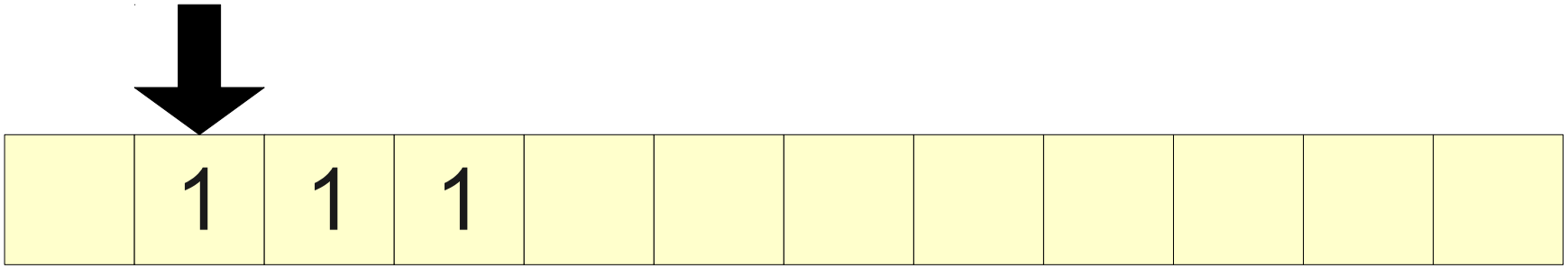
A Simple Turing Machine



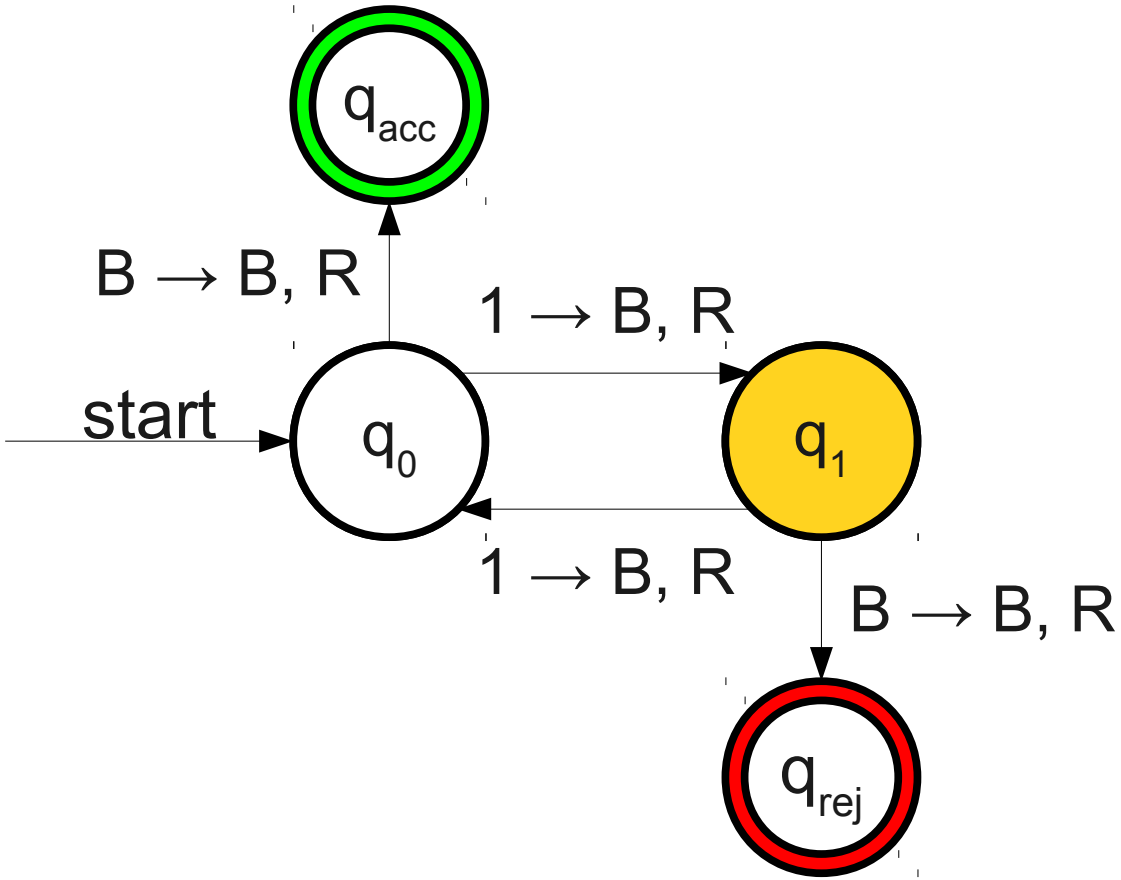


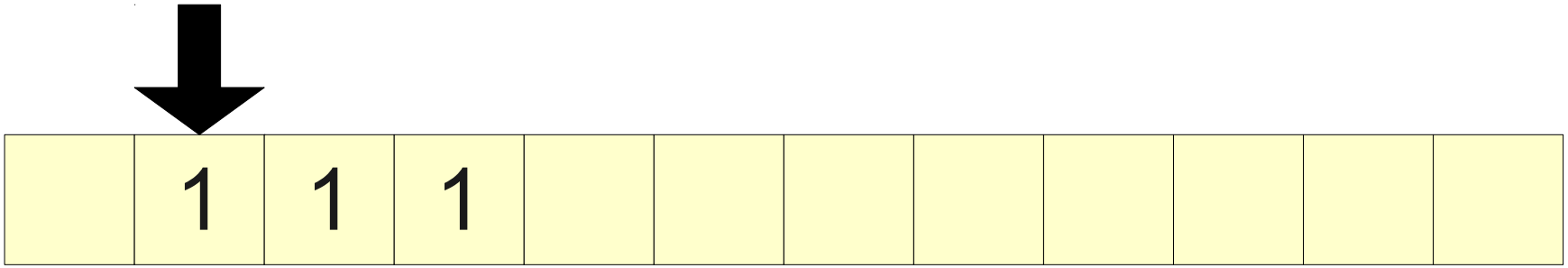
A Simple Turing Machine



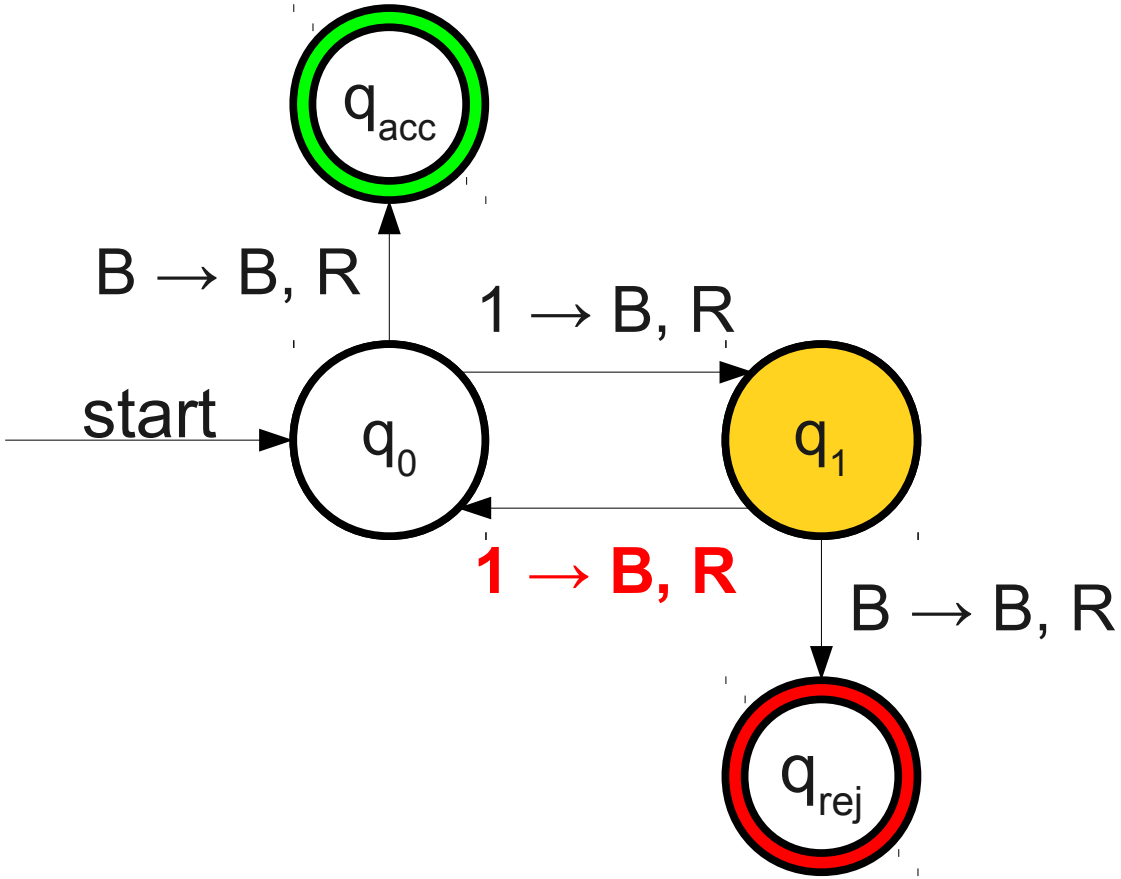


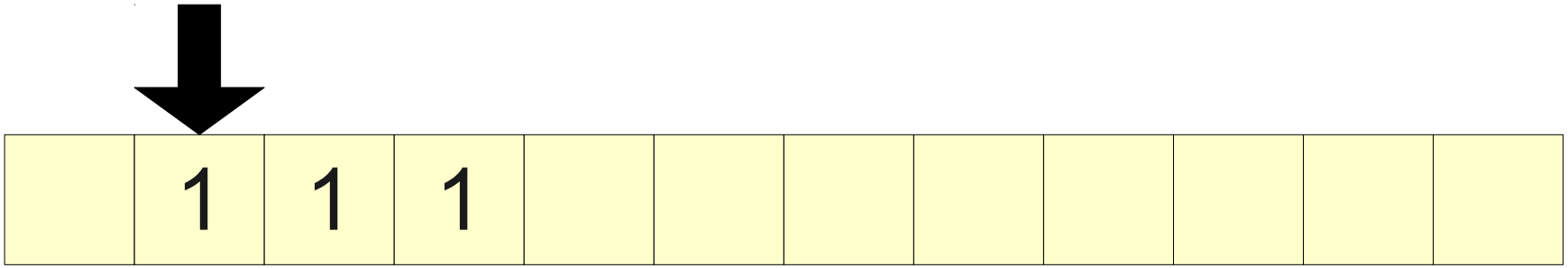
A Simple Turing Machine



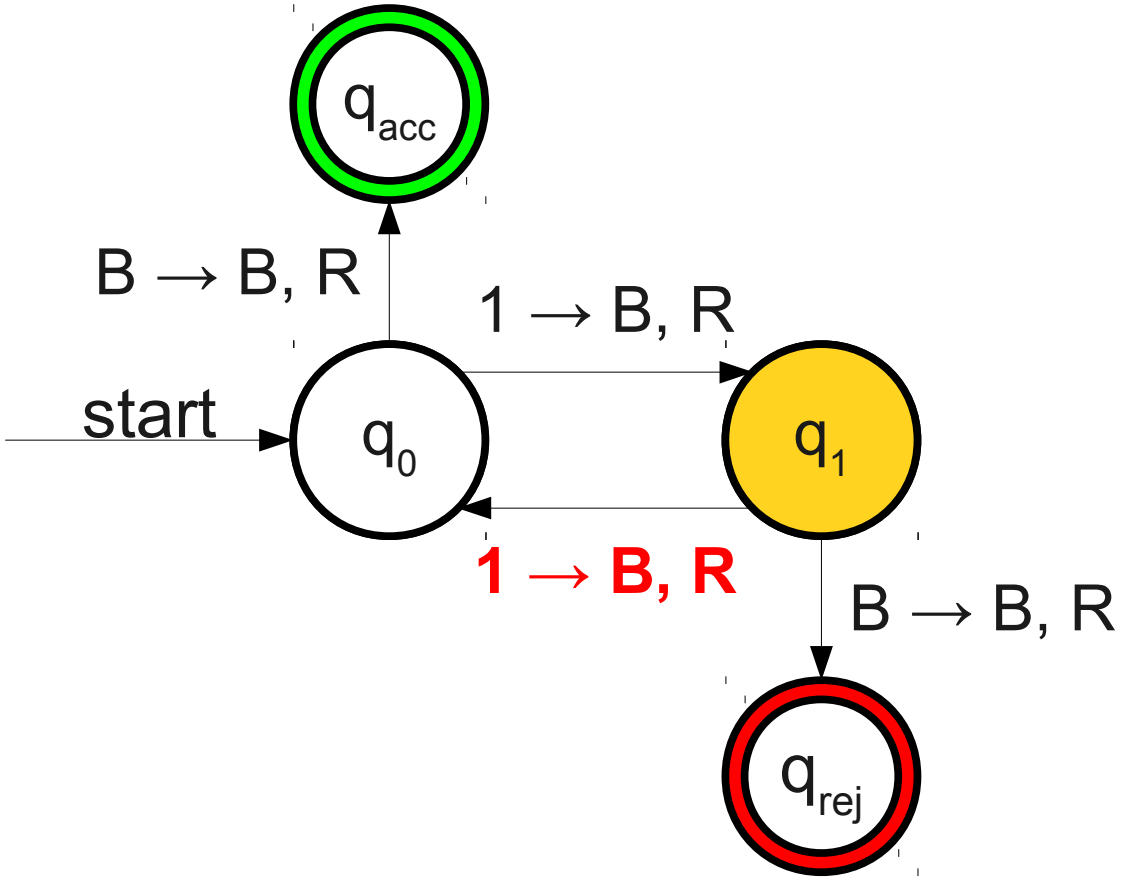


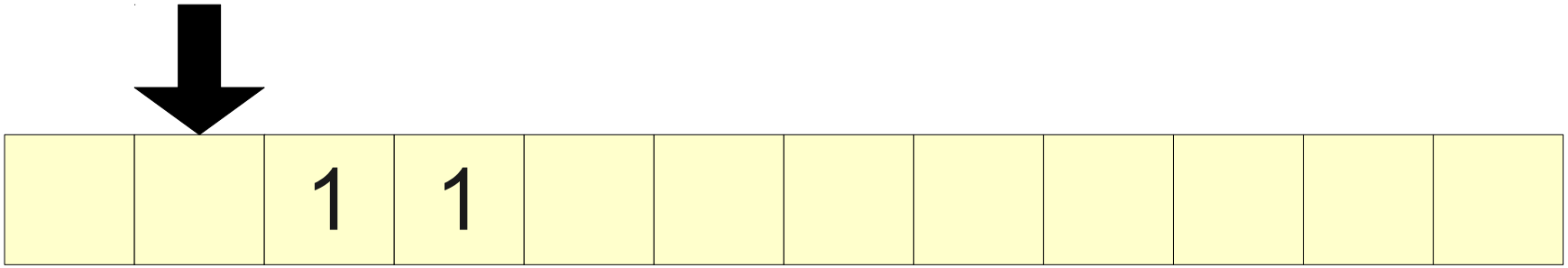
A Simple Turing Machine



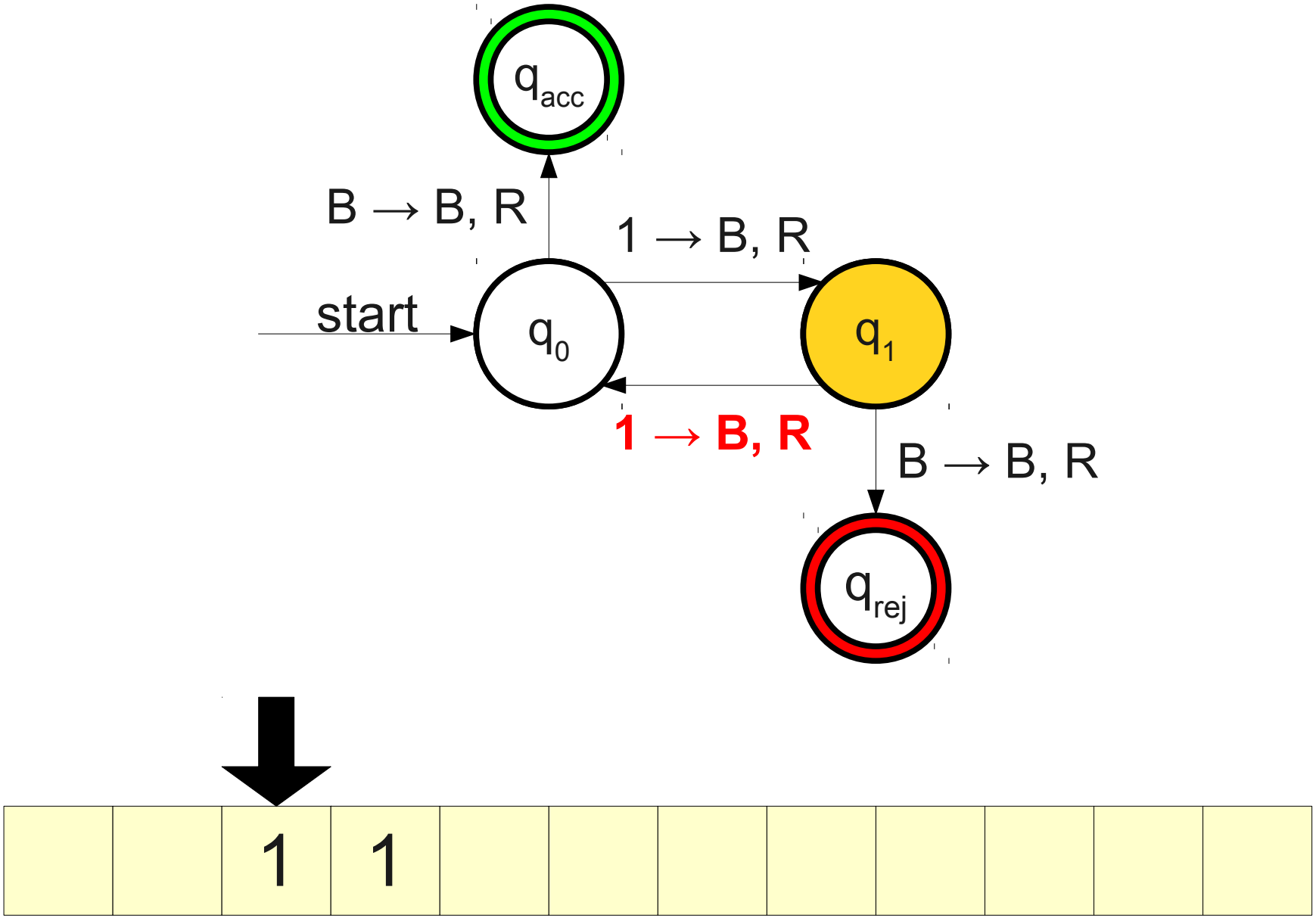


A Simple Turing Machine

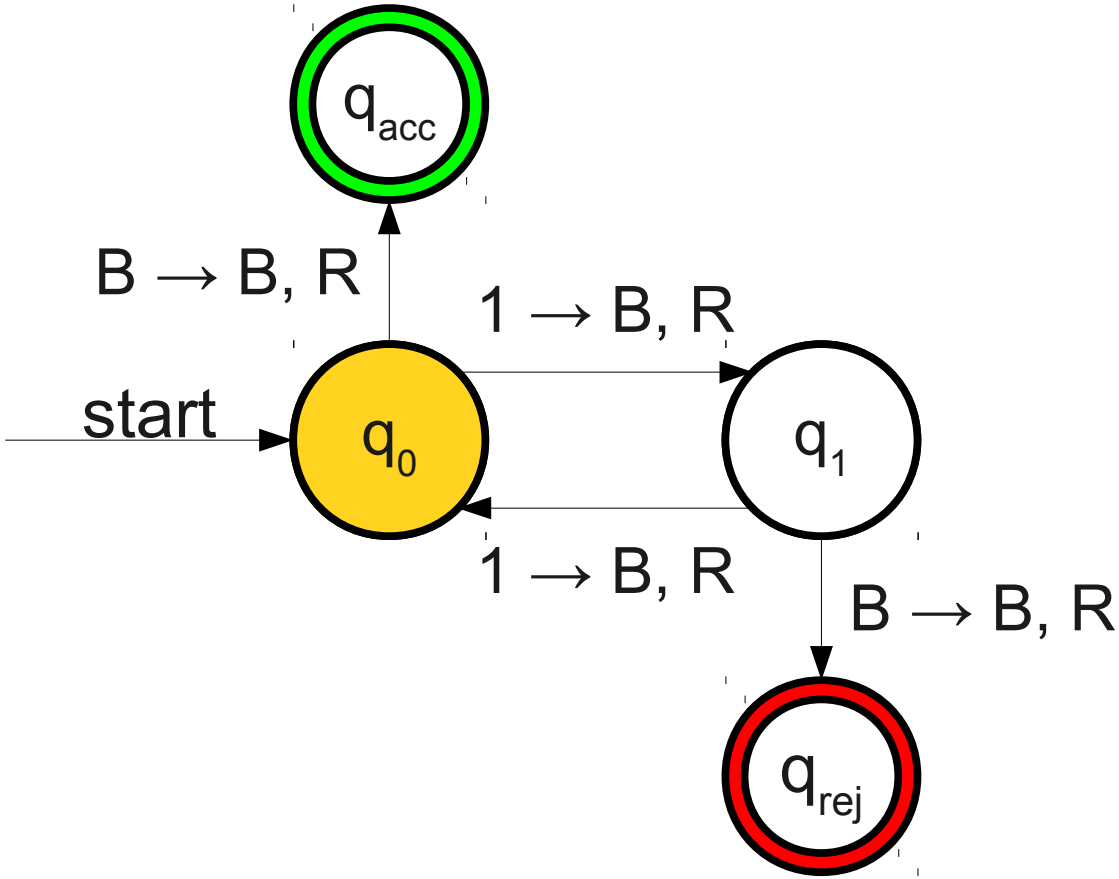


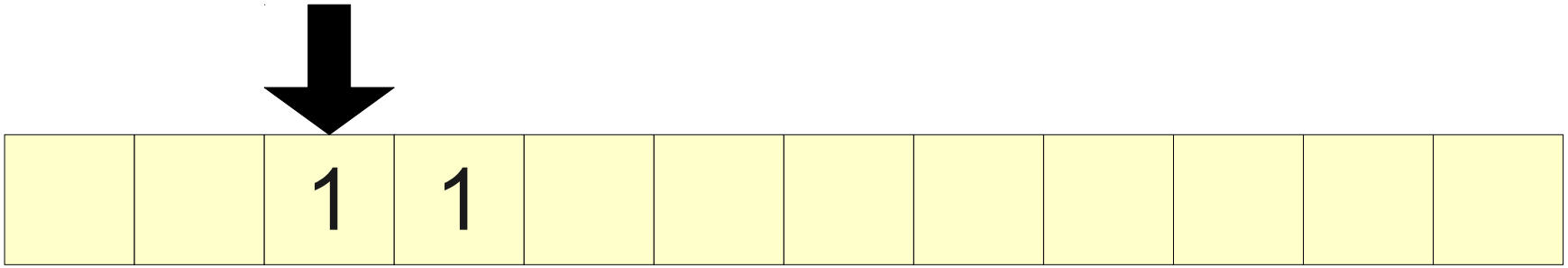


A Simple Turing Machine

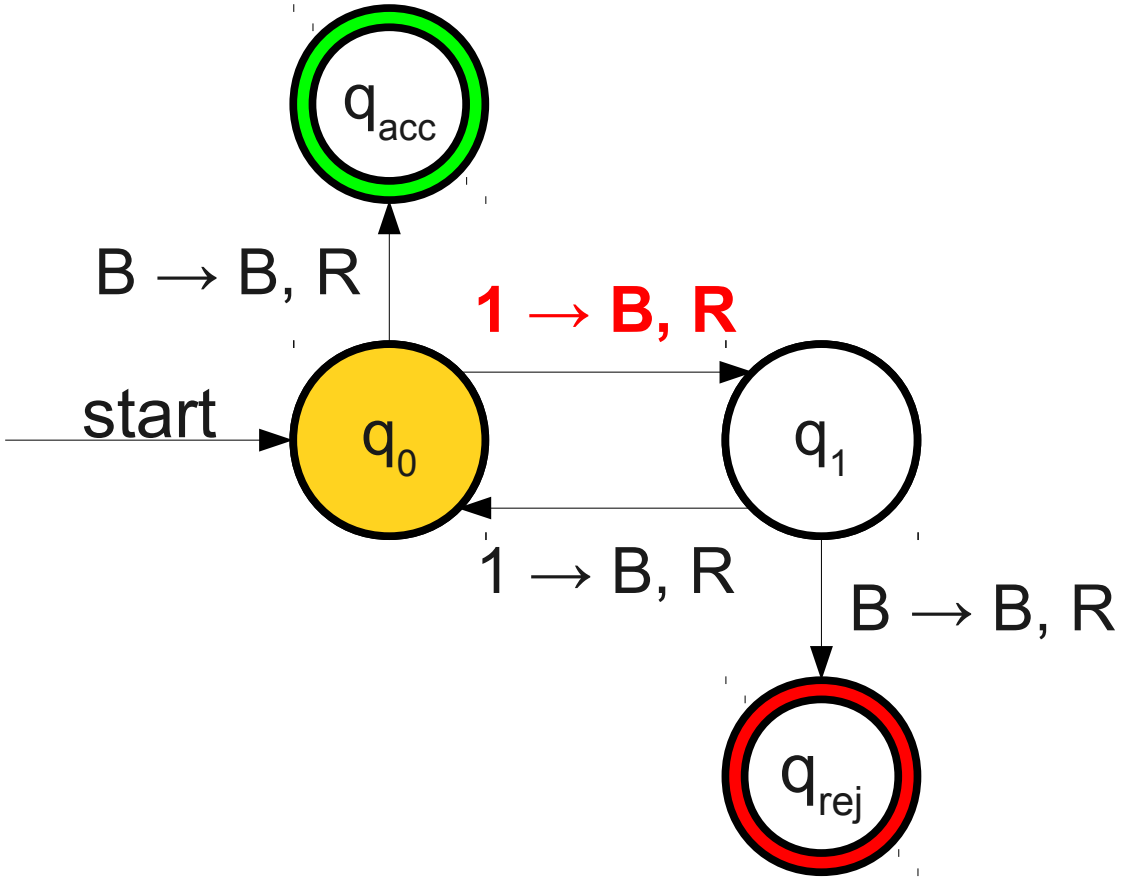


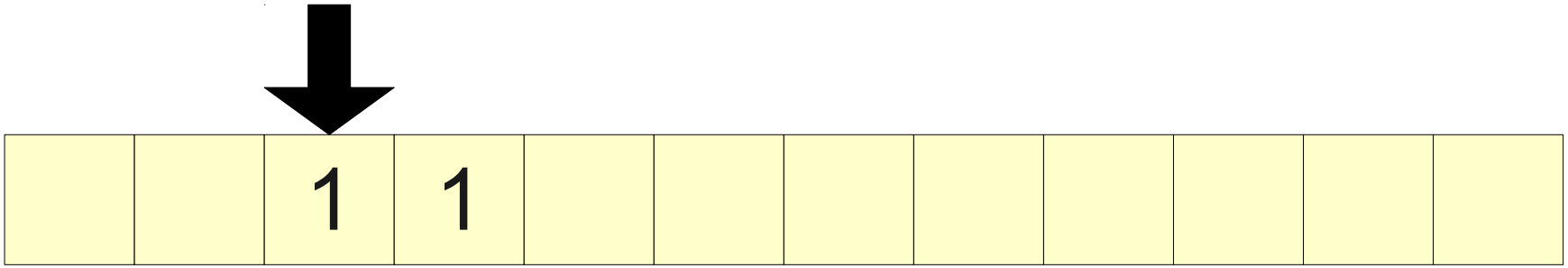
A Simple Turing Machine



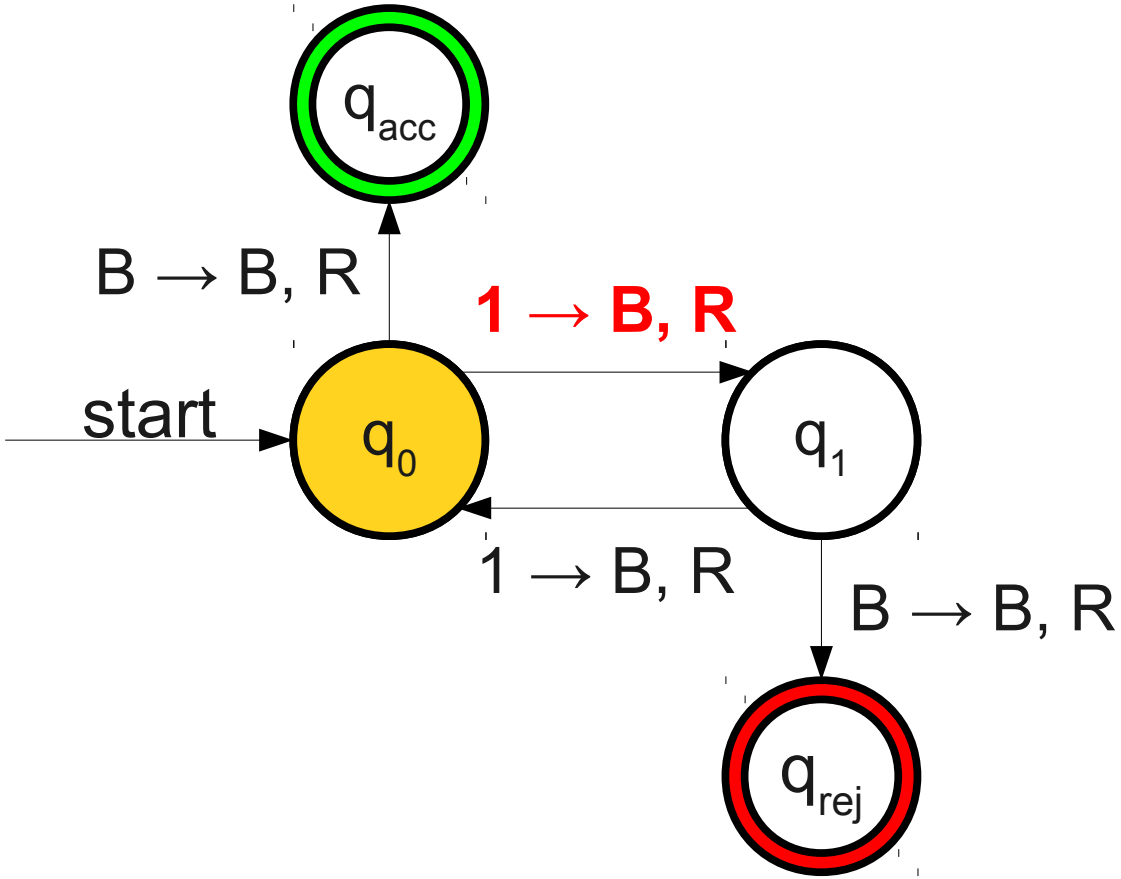


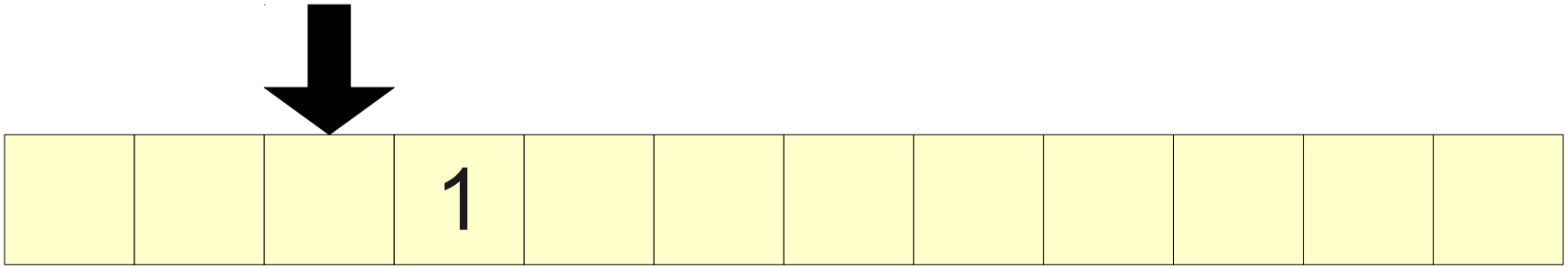
A Simple Turing Machine



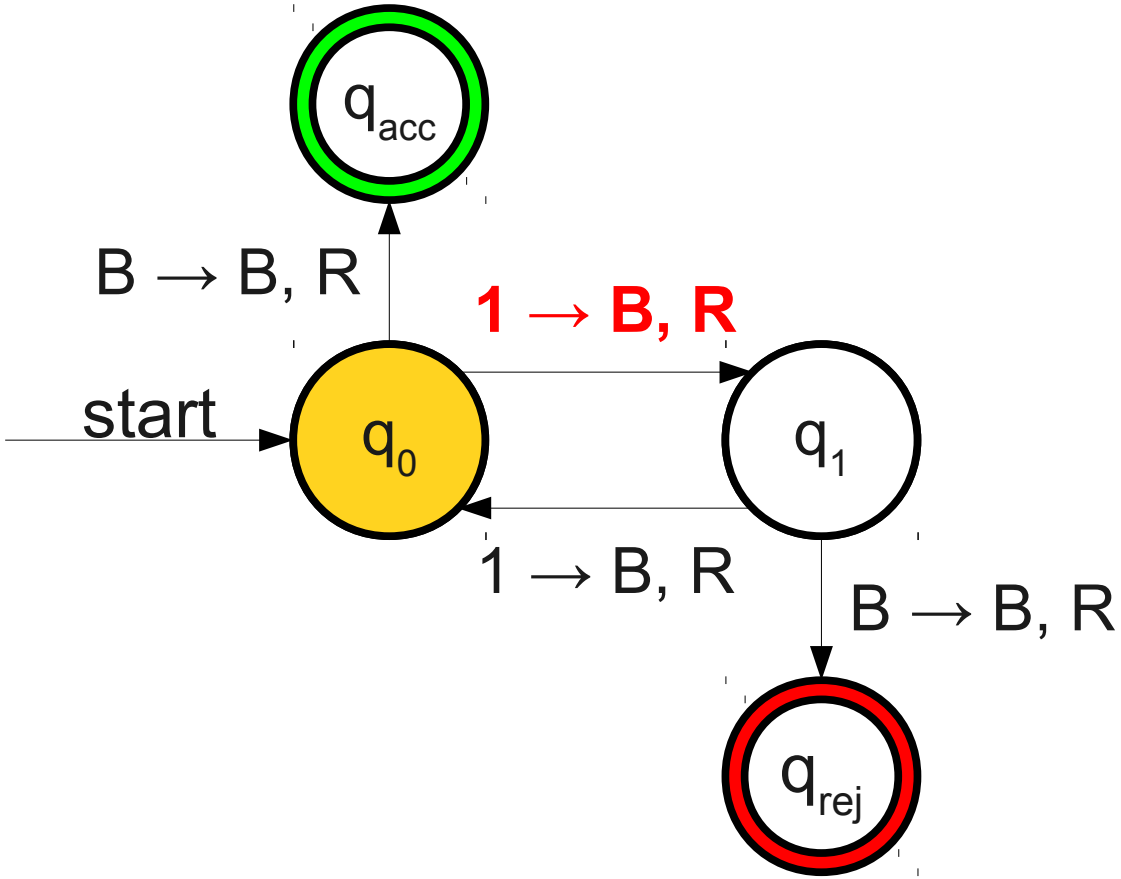


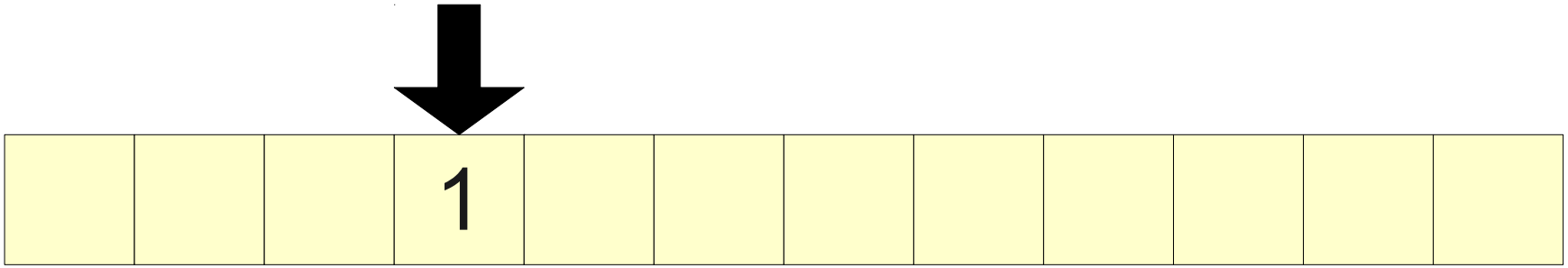
A Simple Turing Machine



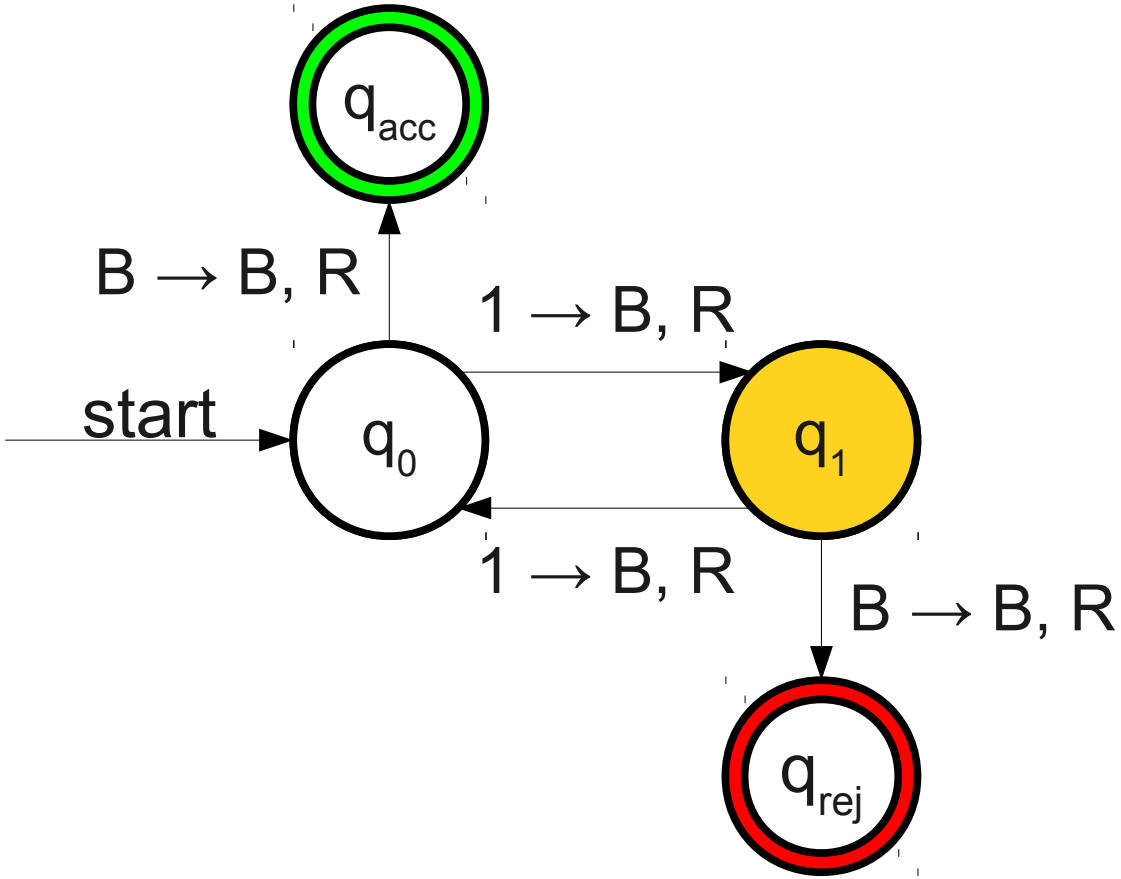


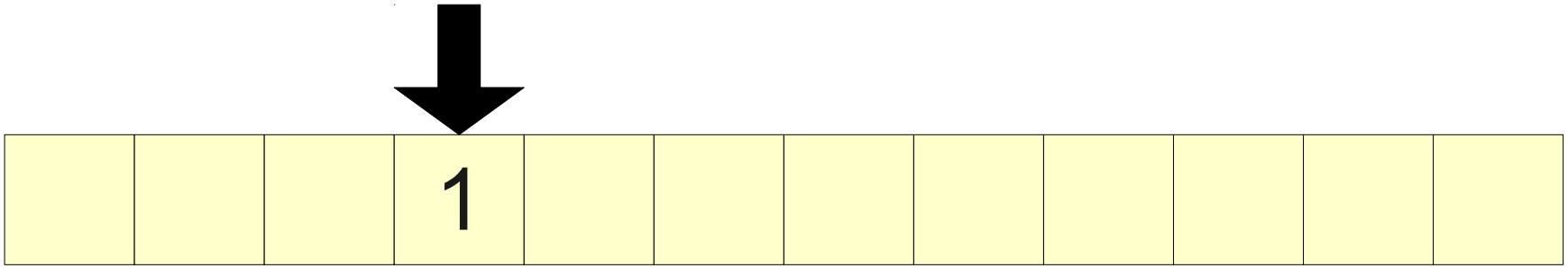
A Simple Turing Machine



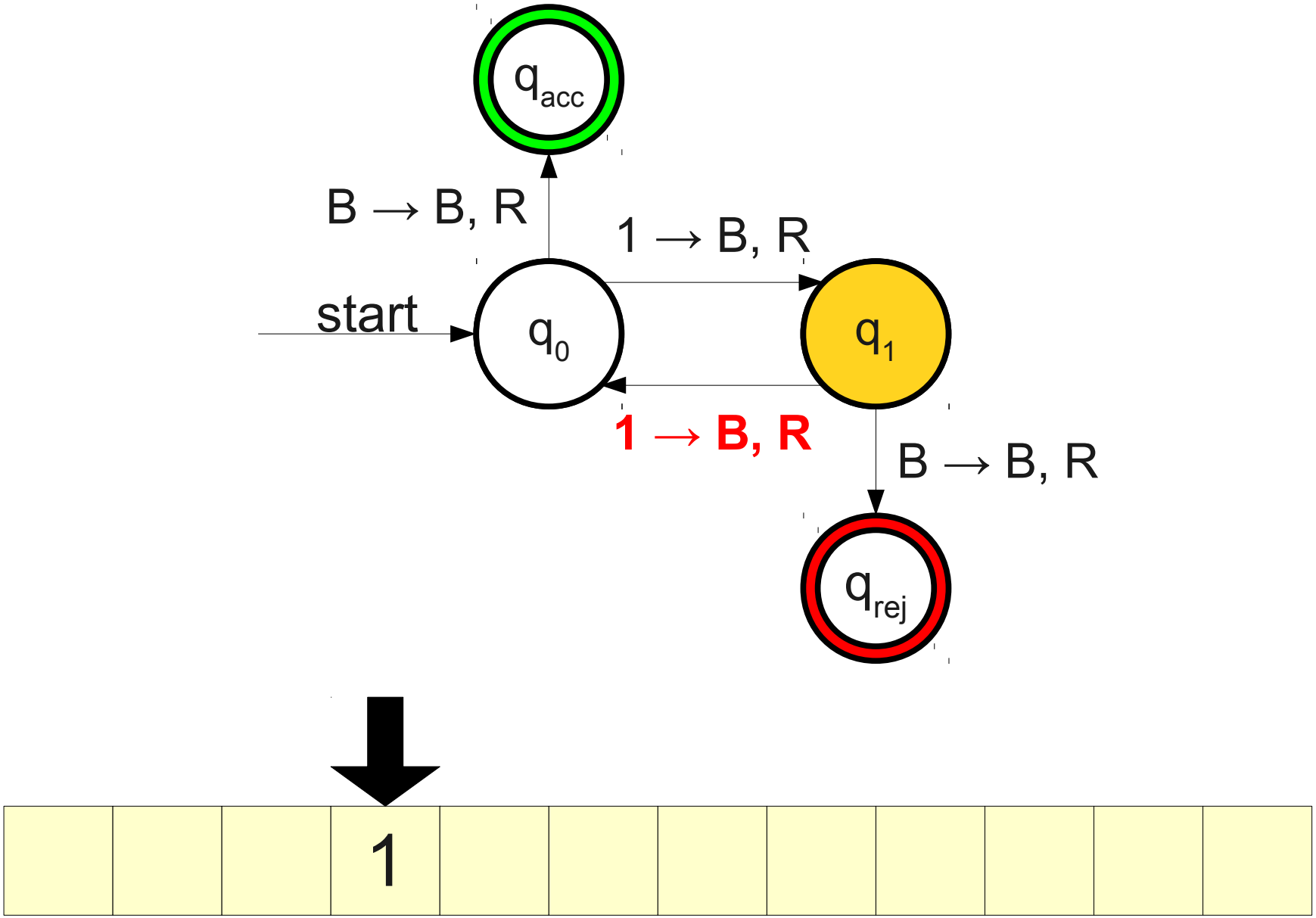


A Simple Turing Machine

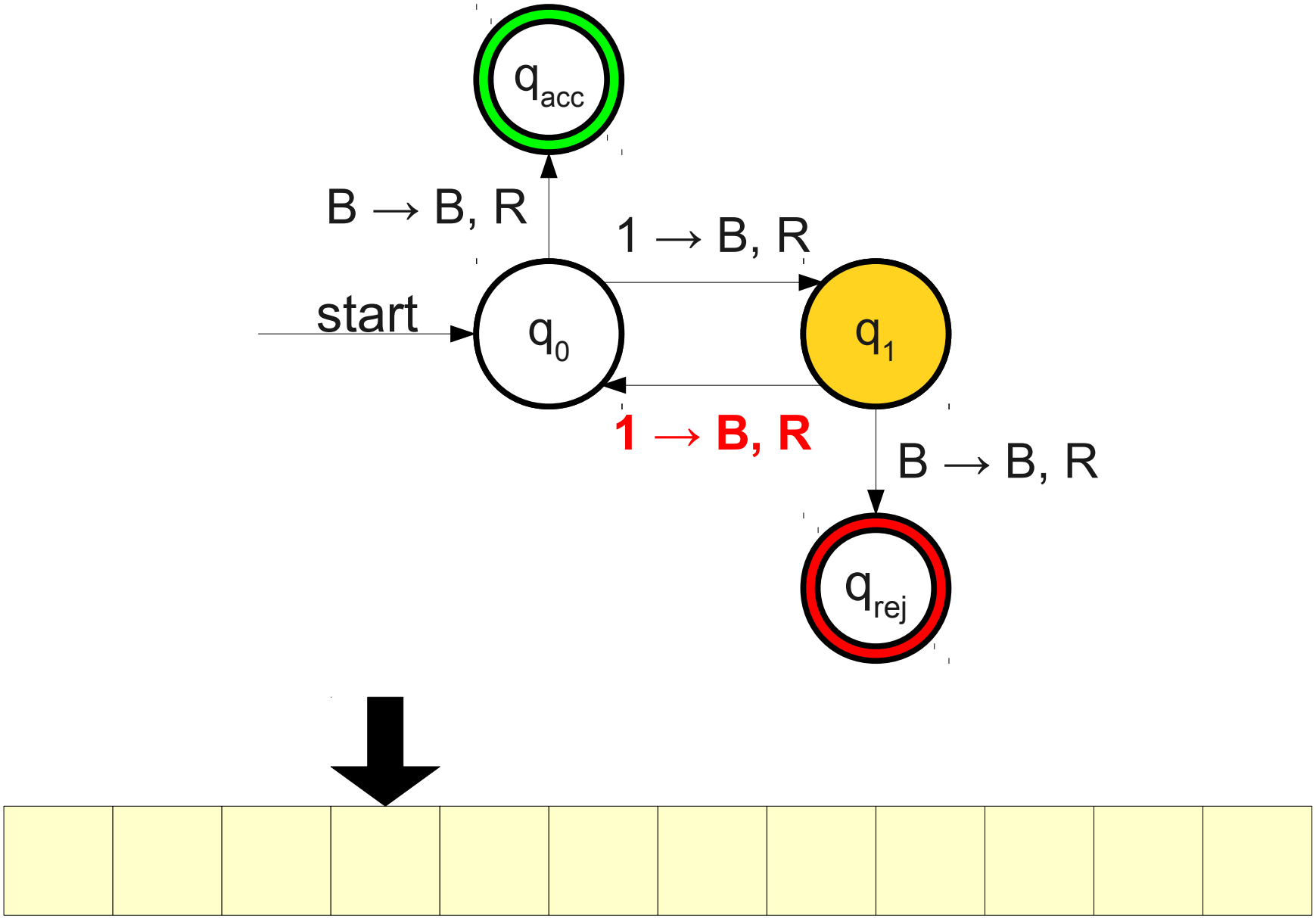




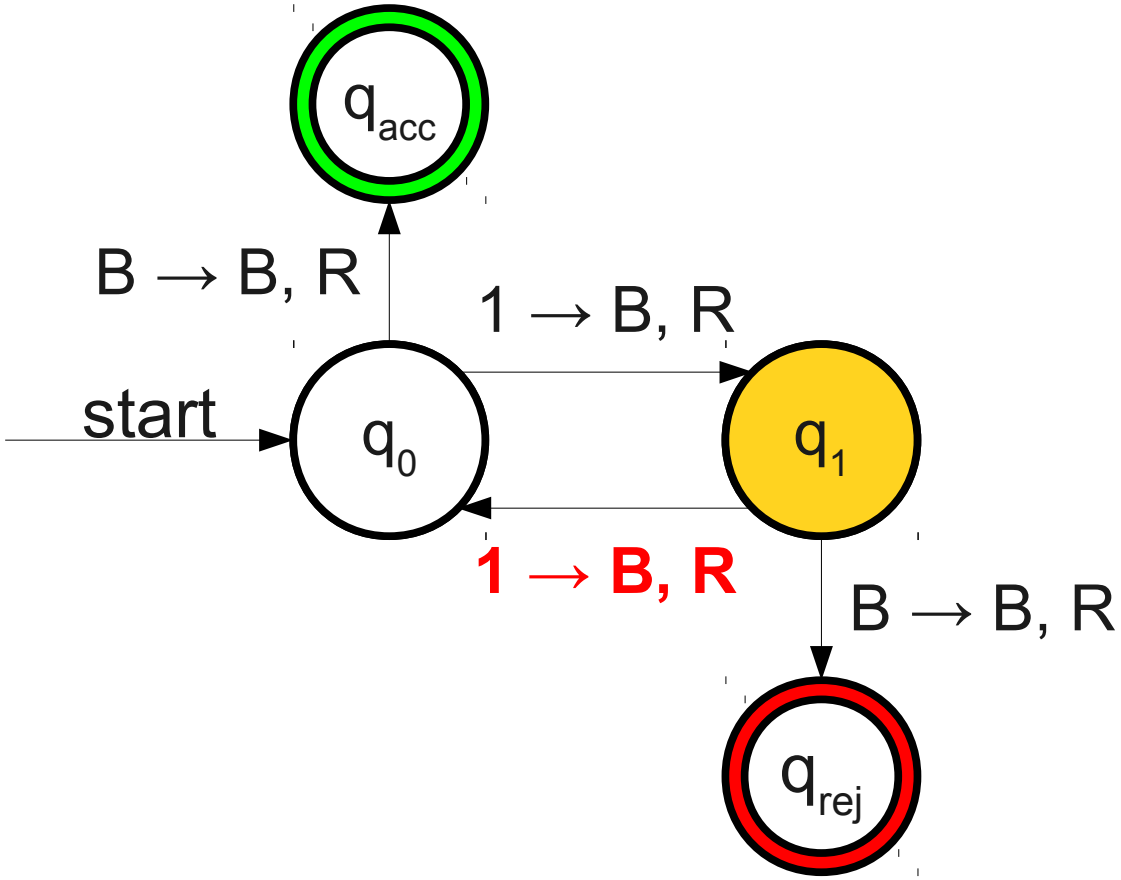
A Simple Turing Machine

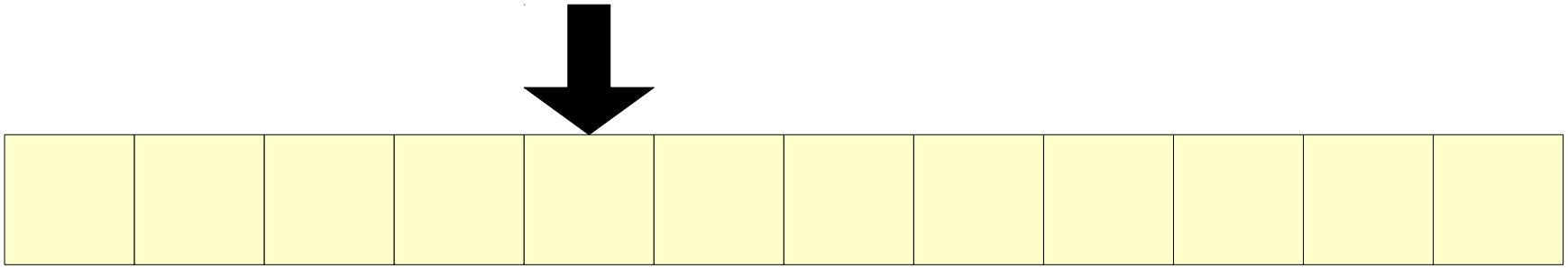


A Simple Turing Machine

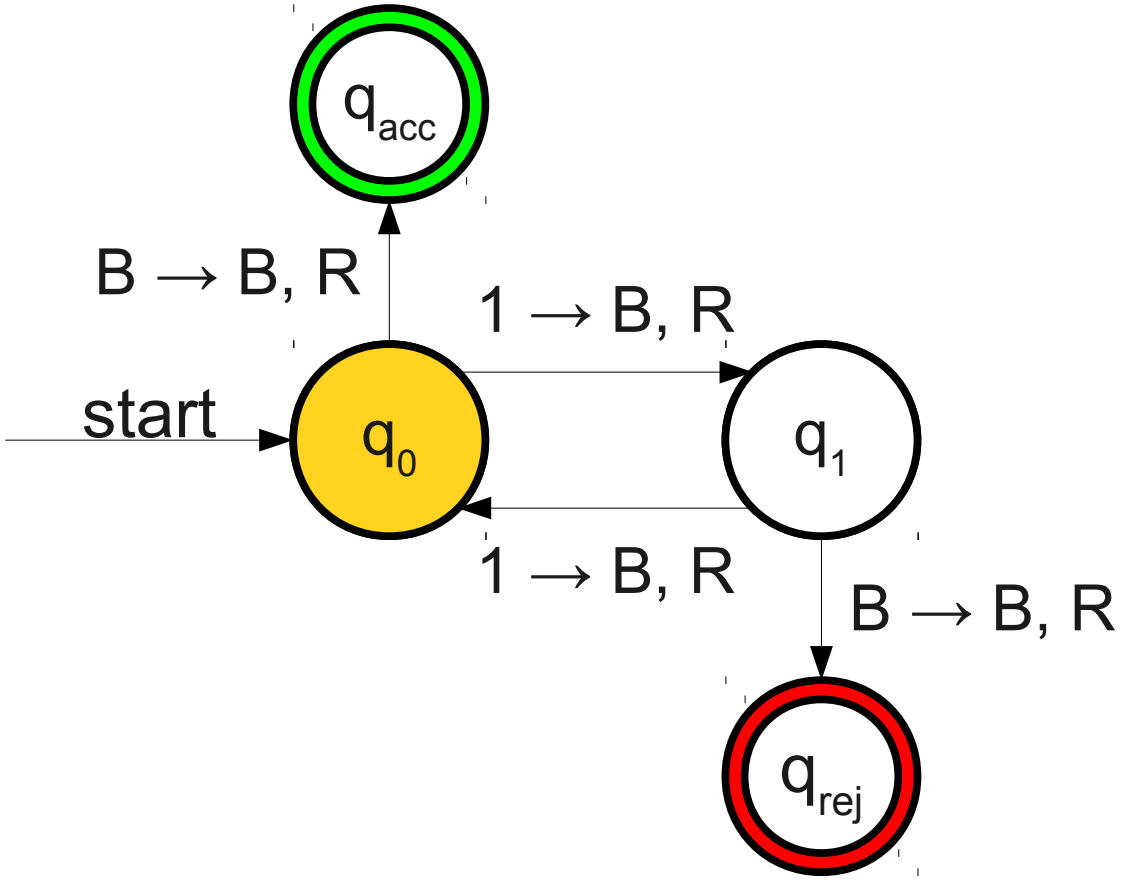


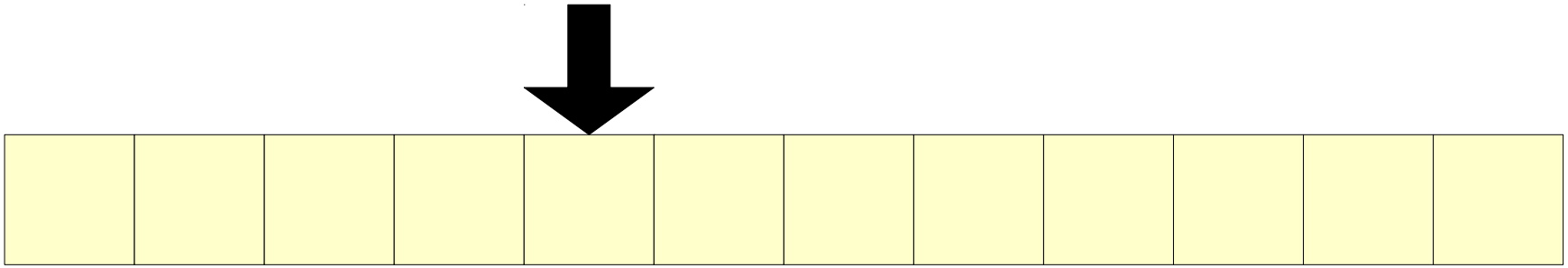
A Simple Turing Machine



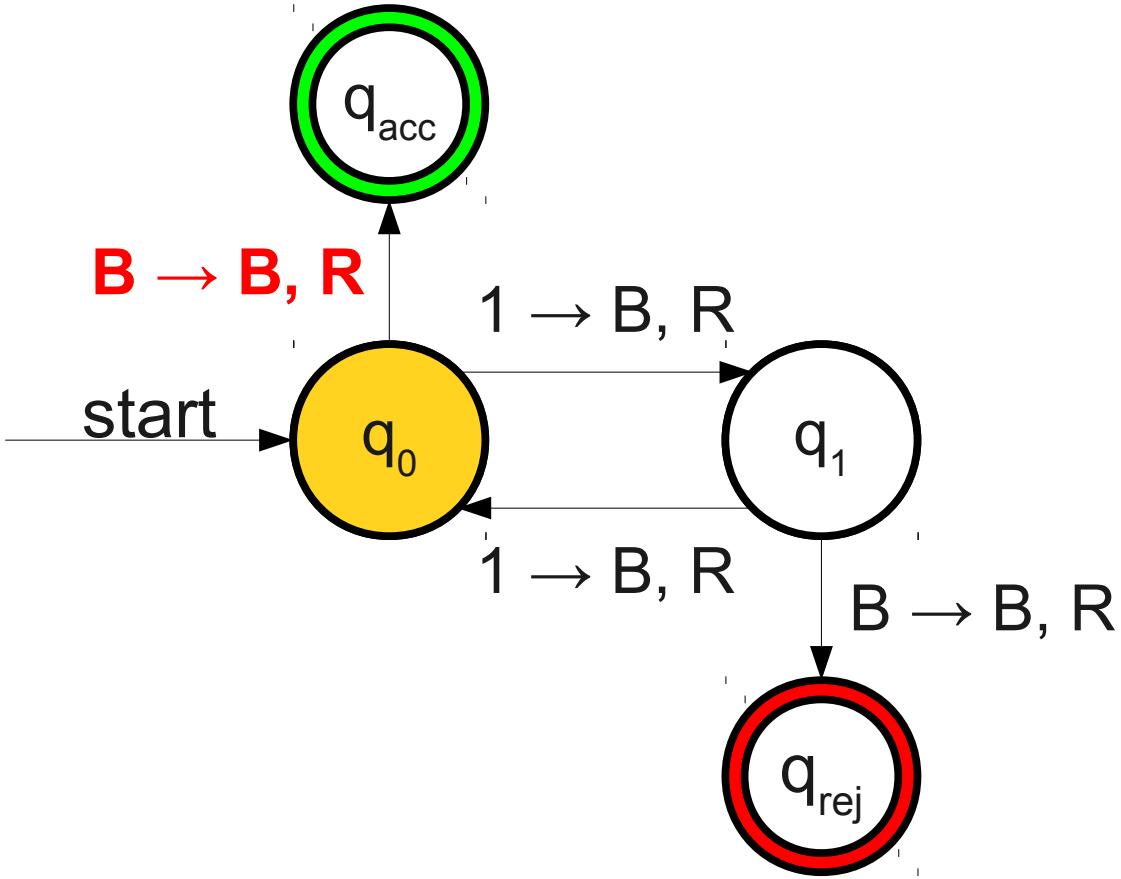


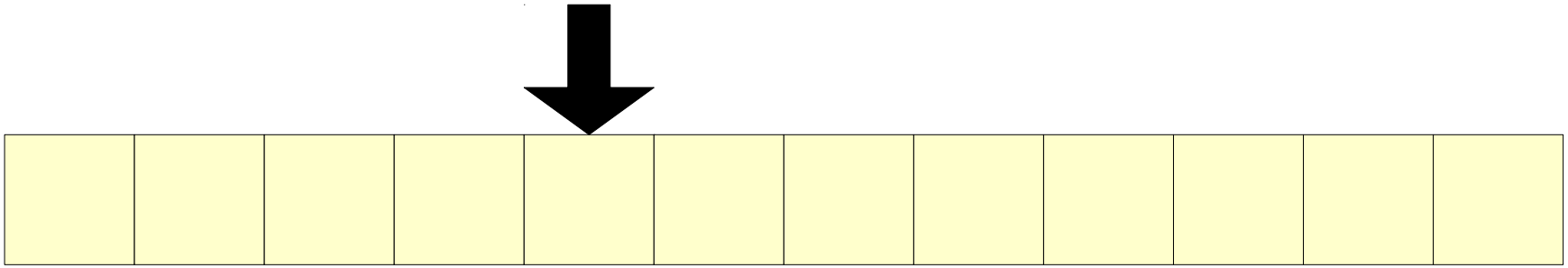
A Simple Turing Machine



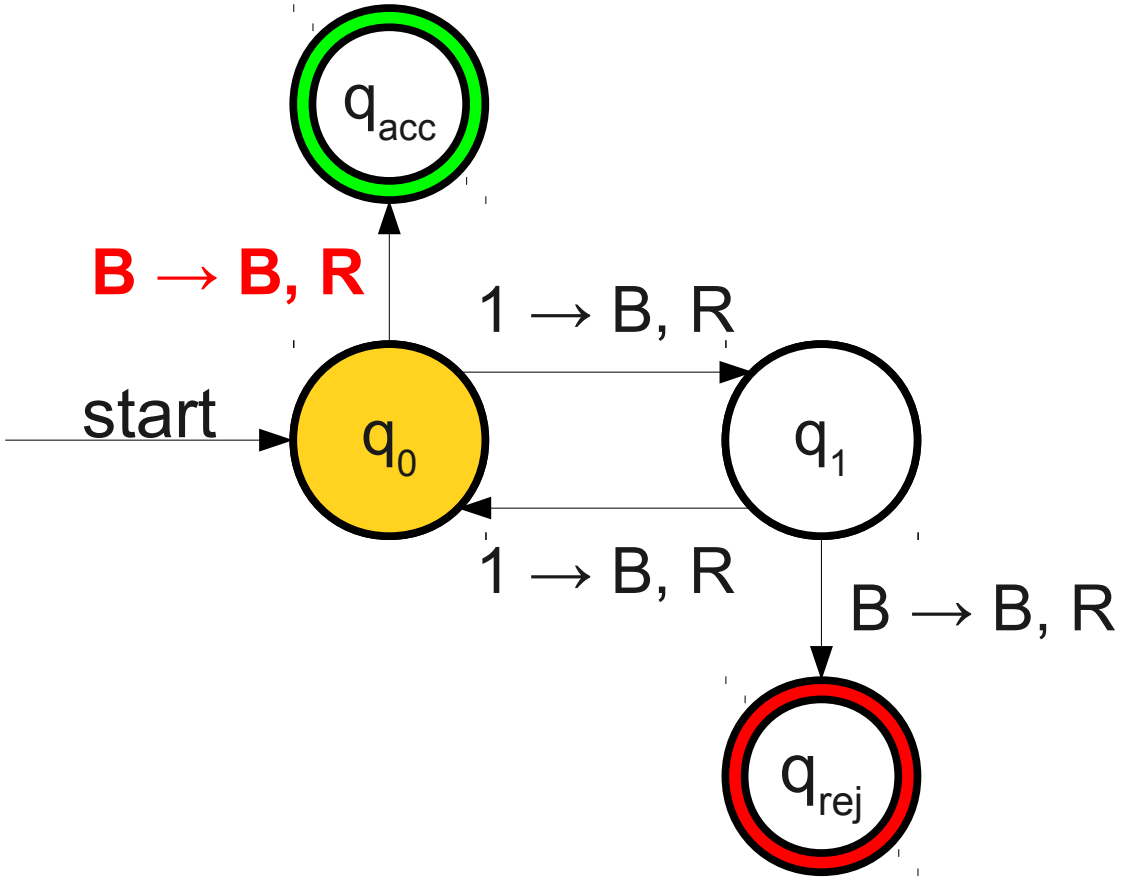


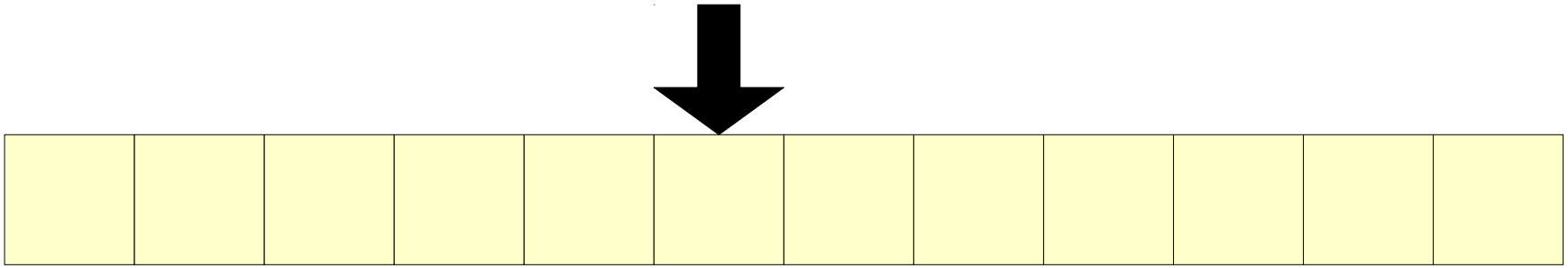
A Simple Turing Machine



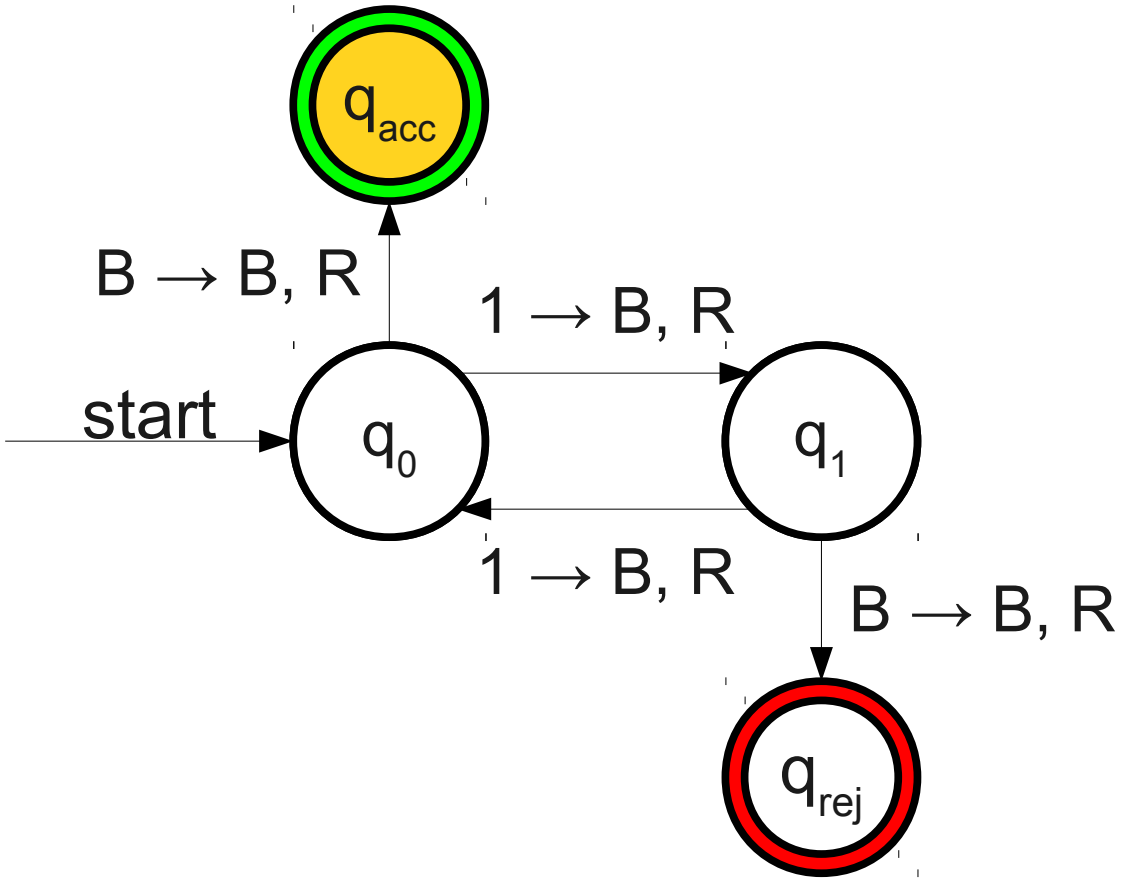


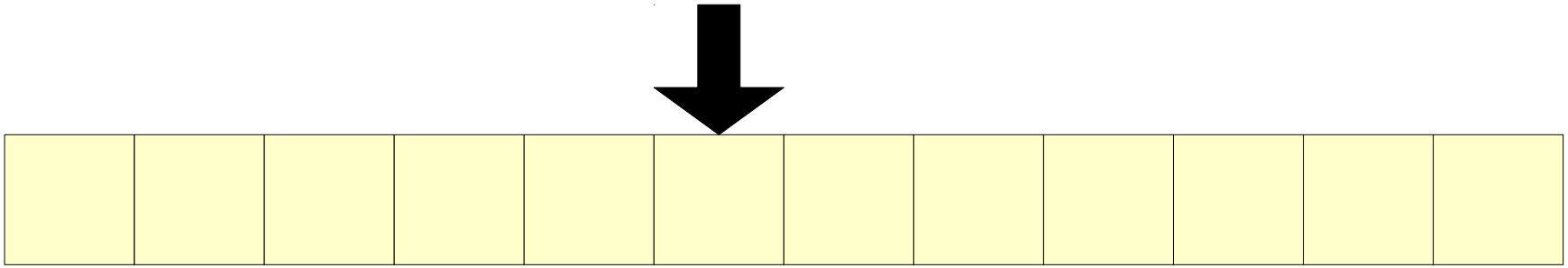
A Simple Turing Machine



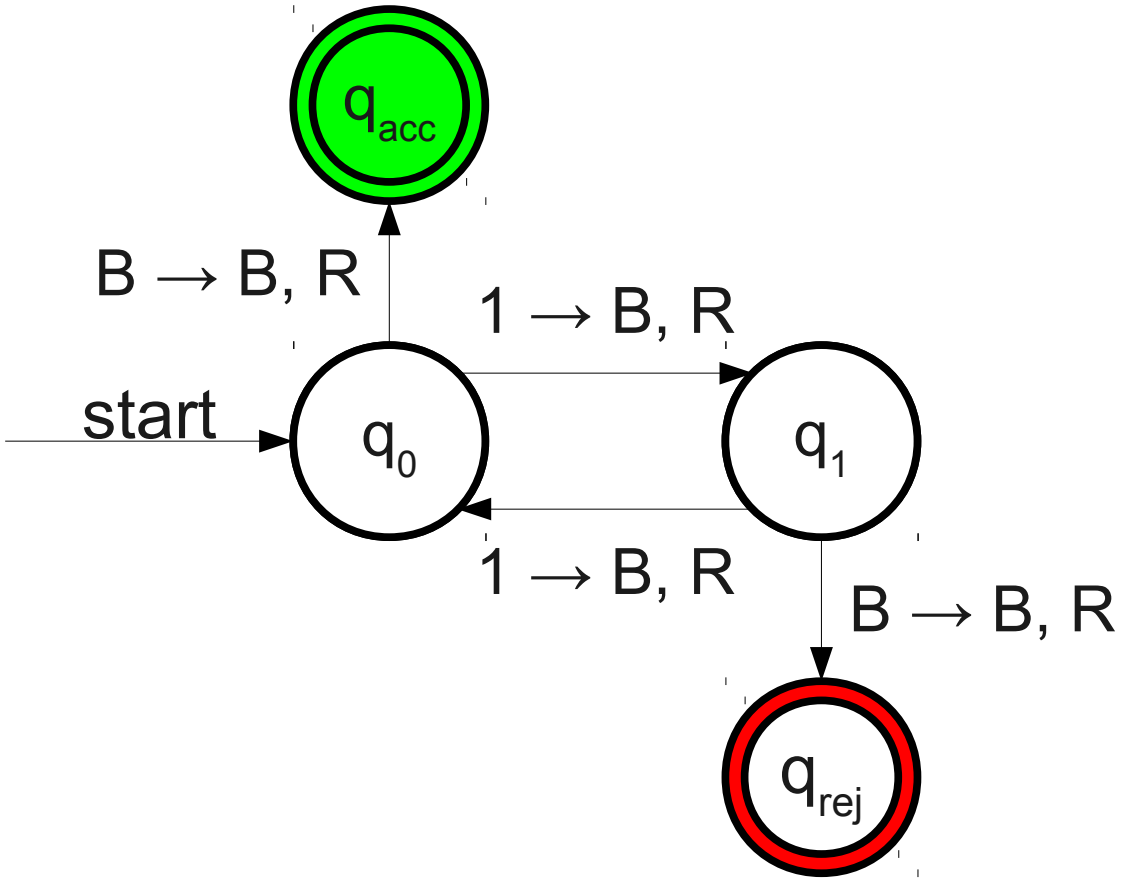


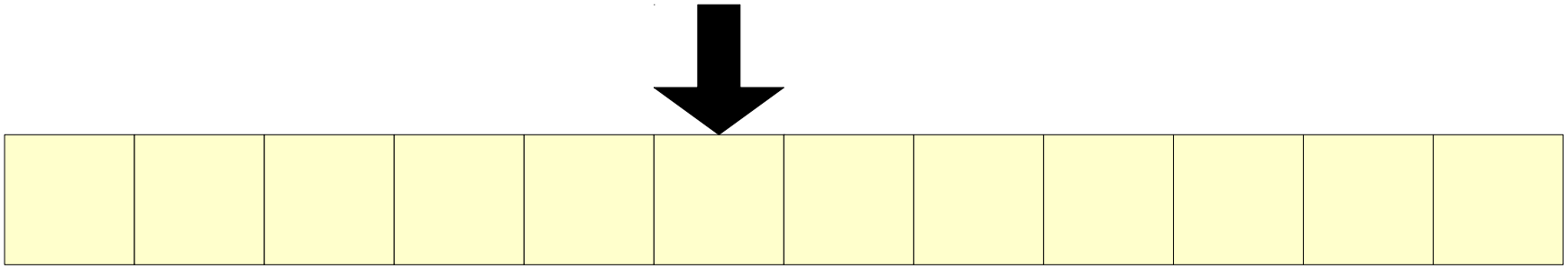
A Simple Turing Machine



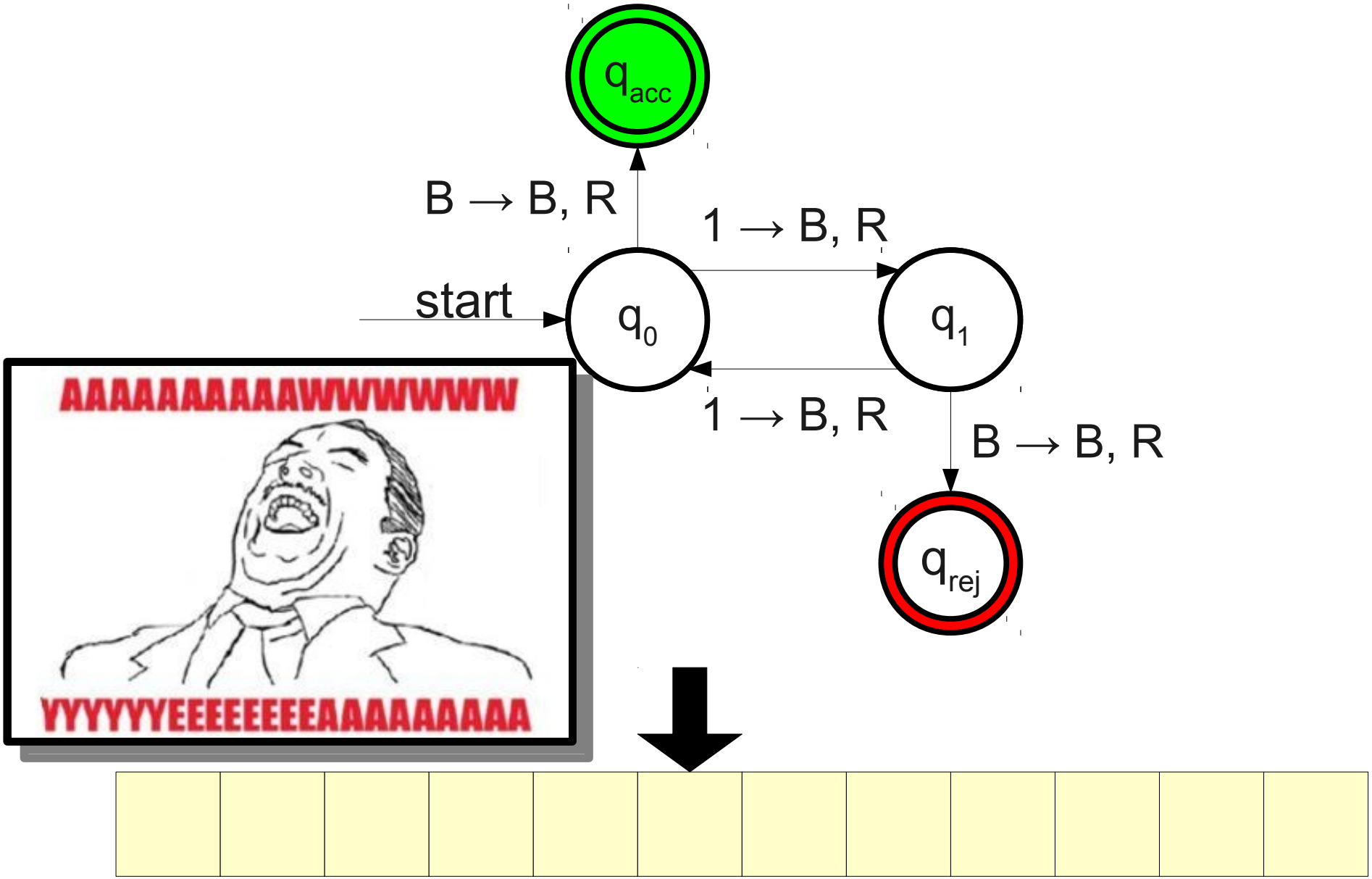


A Simple Turing Machine

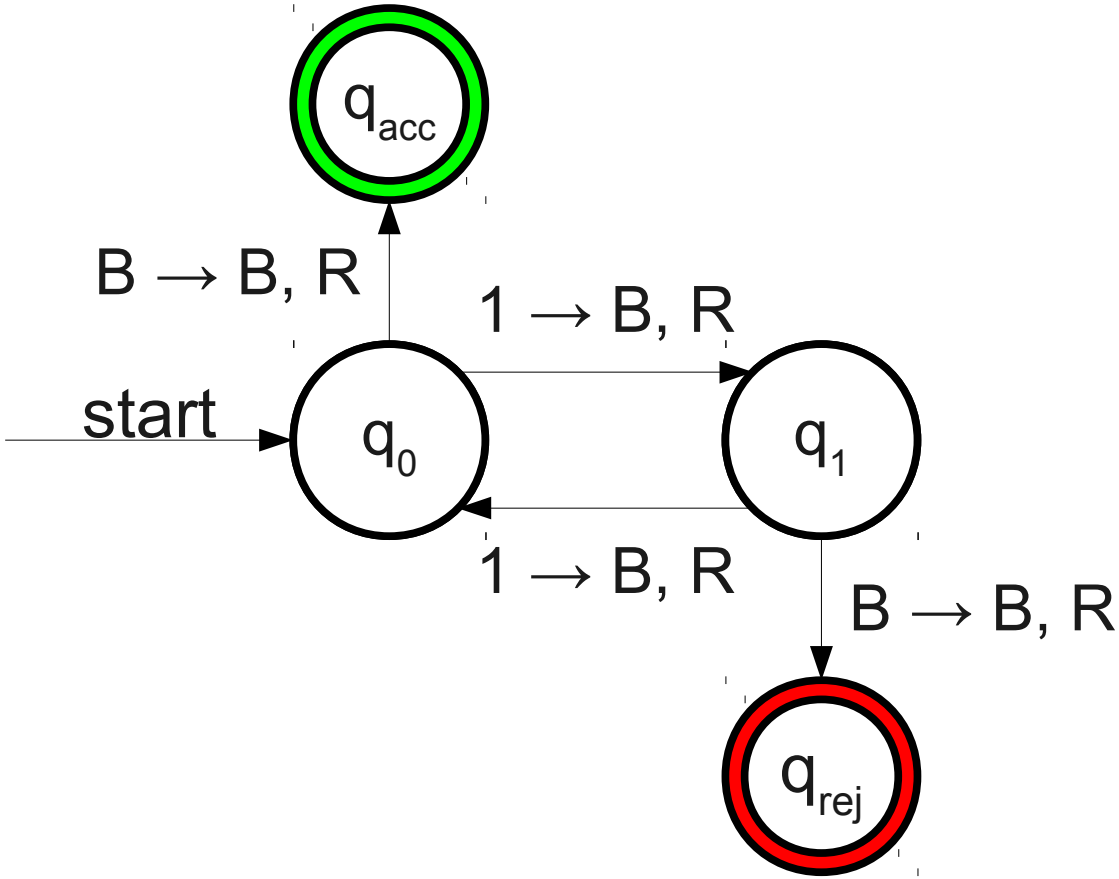




A Simple Turing Machine

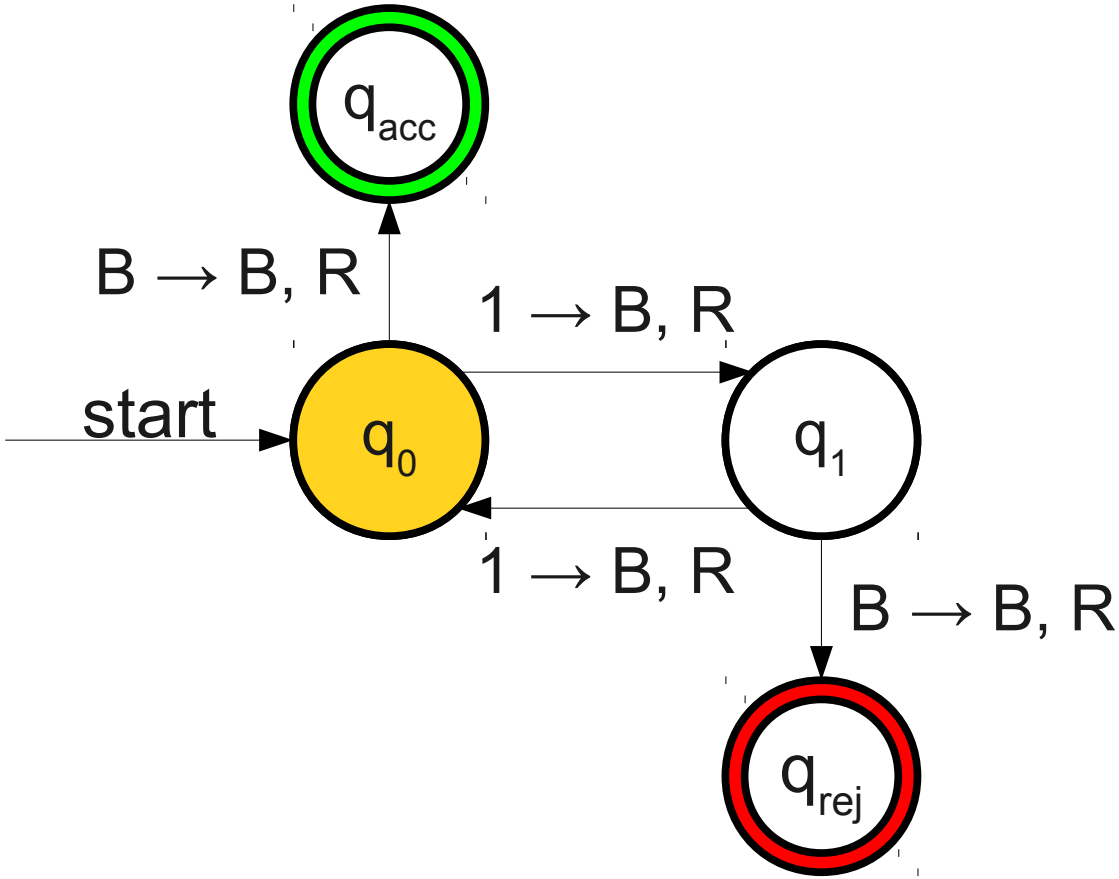


A Simple Turing Machine



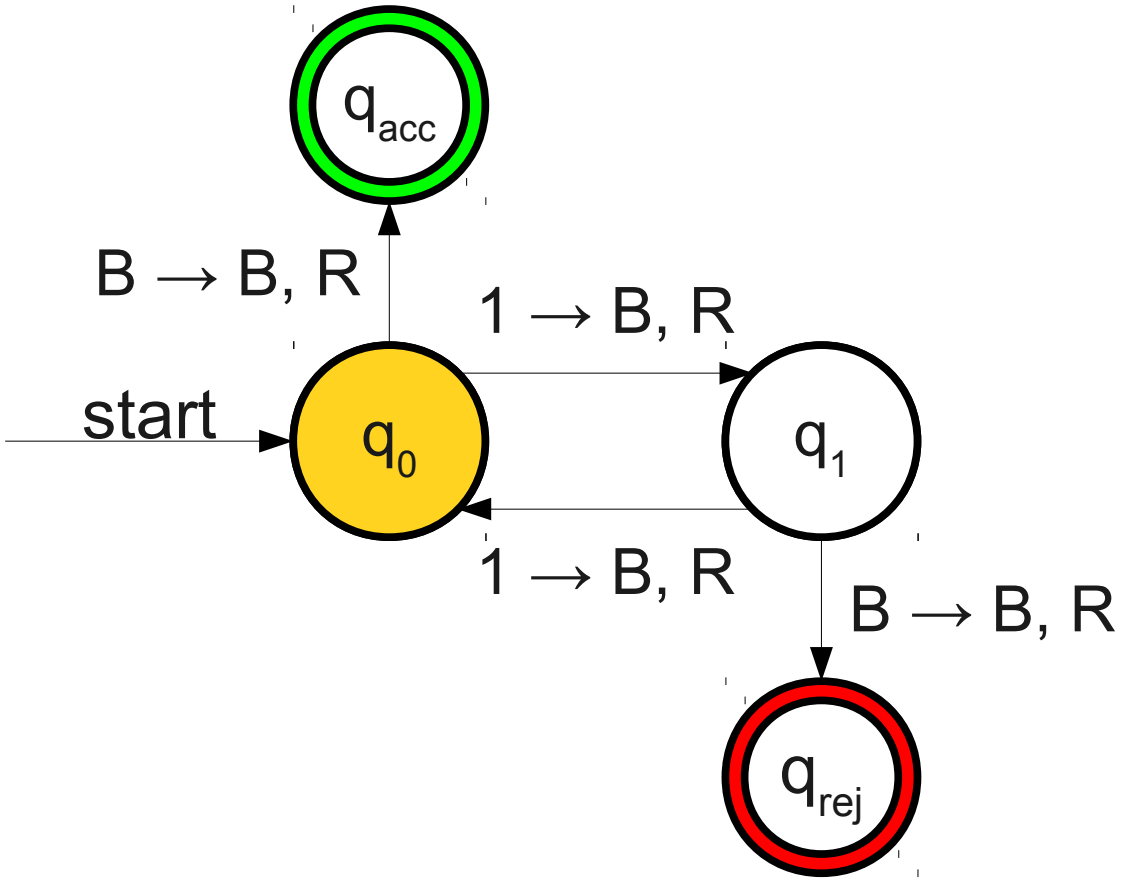
1											

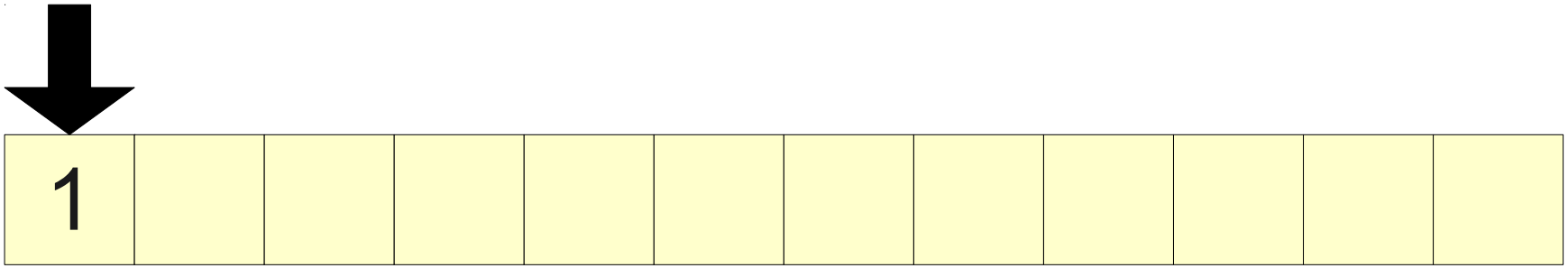
A Simple Turing Machine



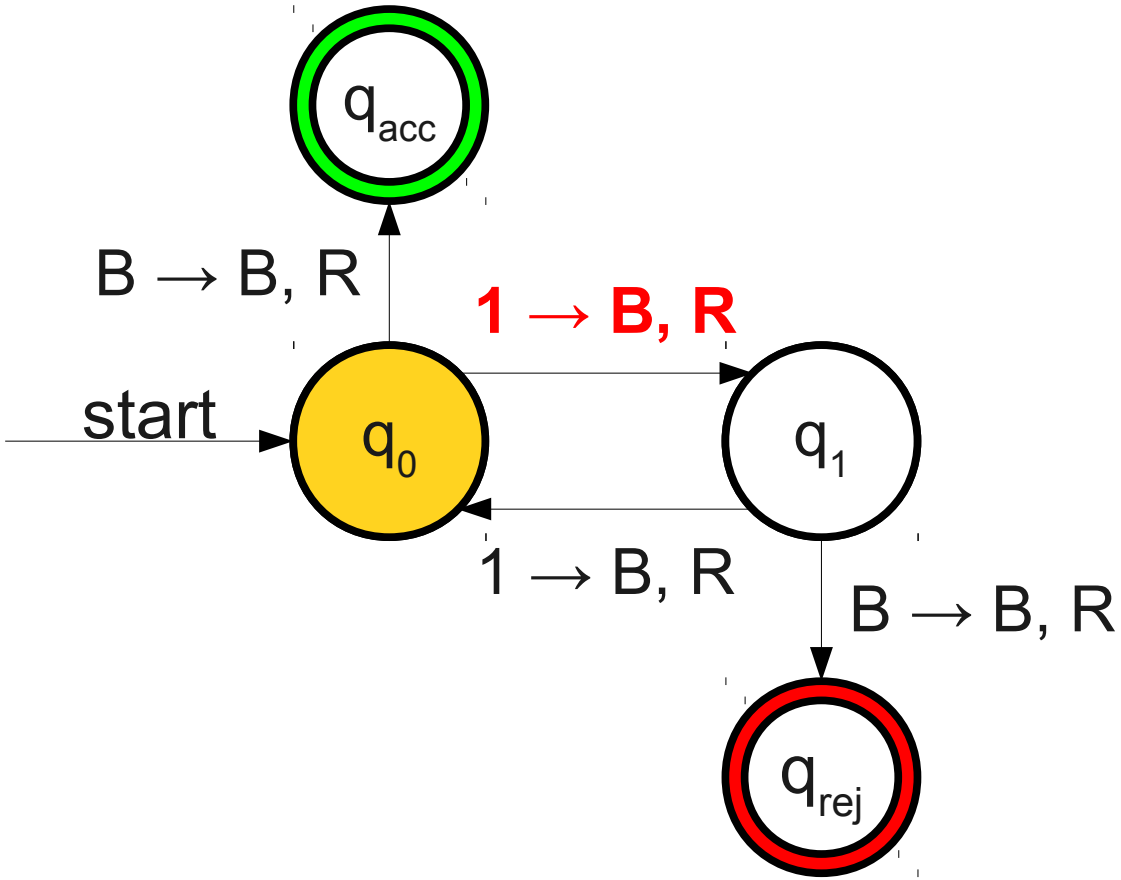
1											

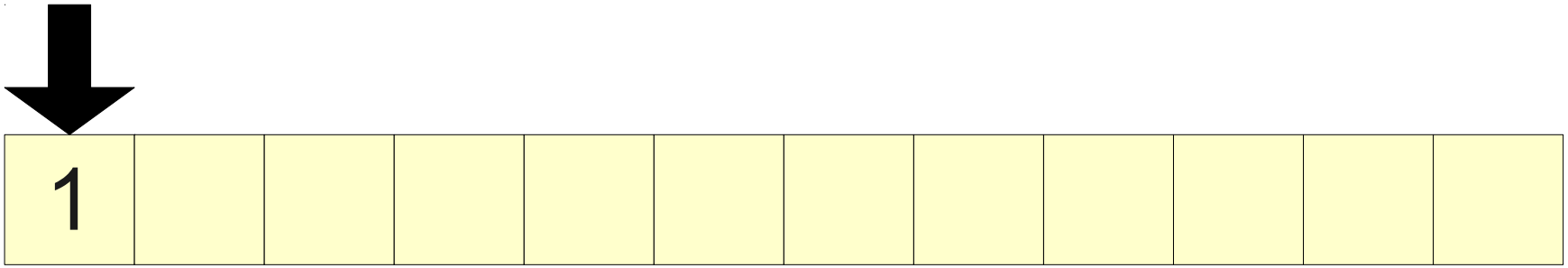
A Simple Turing Machine



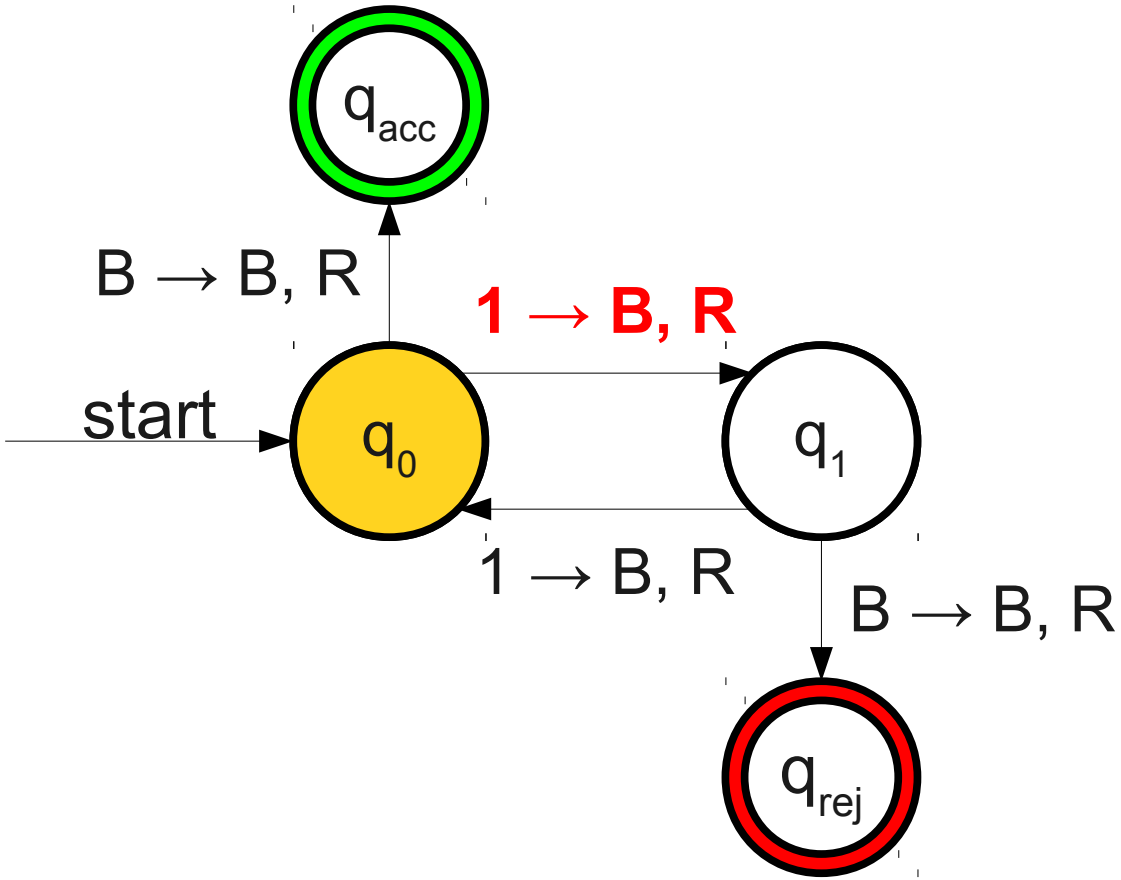


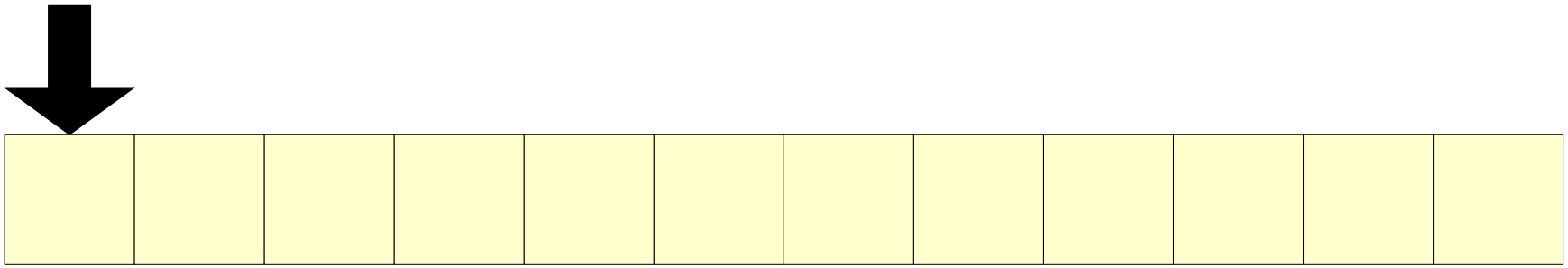
A Simple Turing Machine



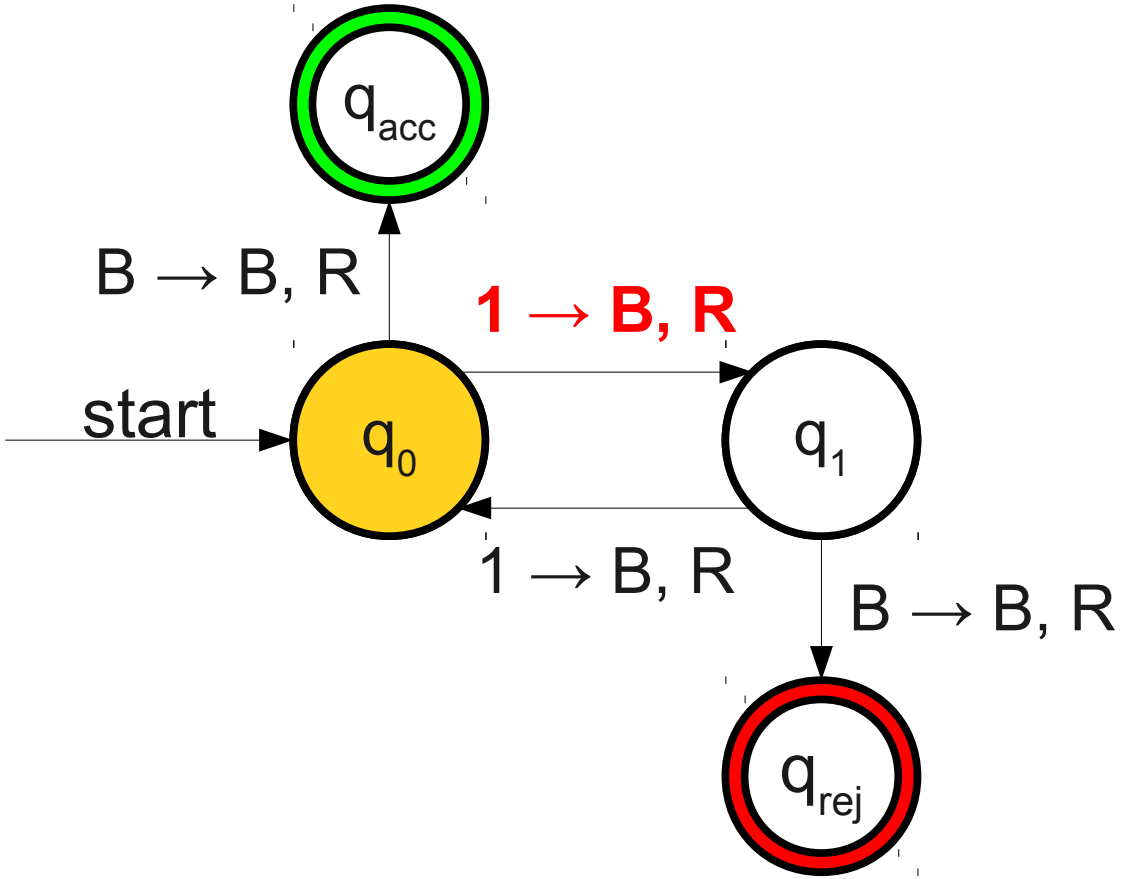


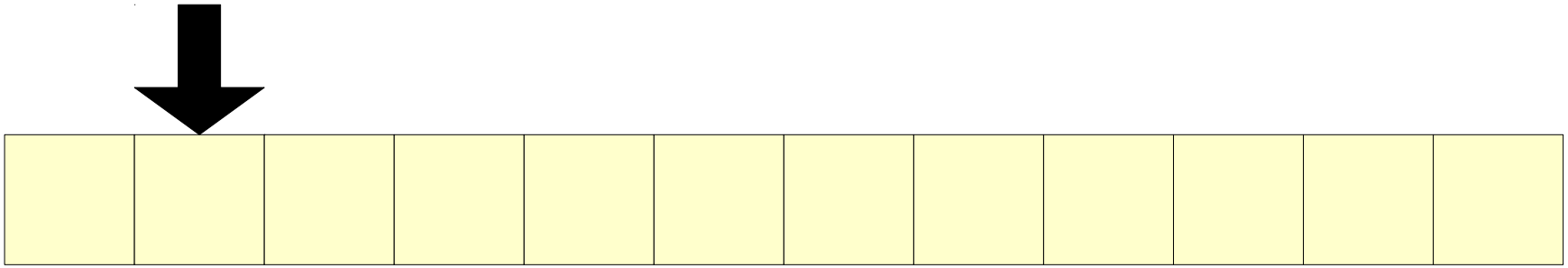
A Simple Turing Machine



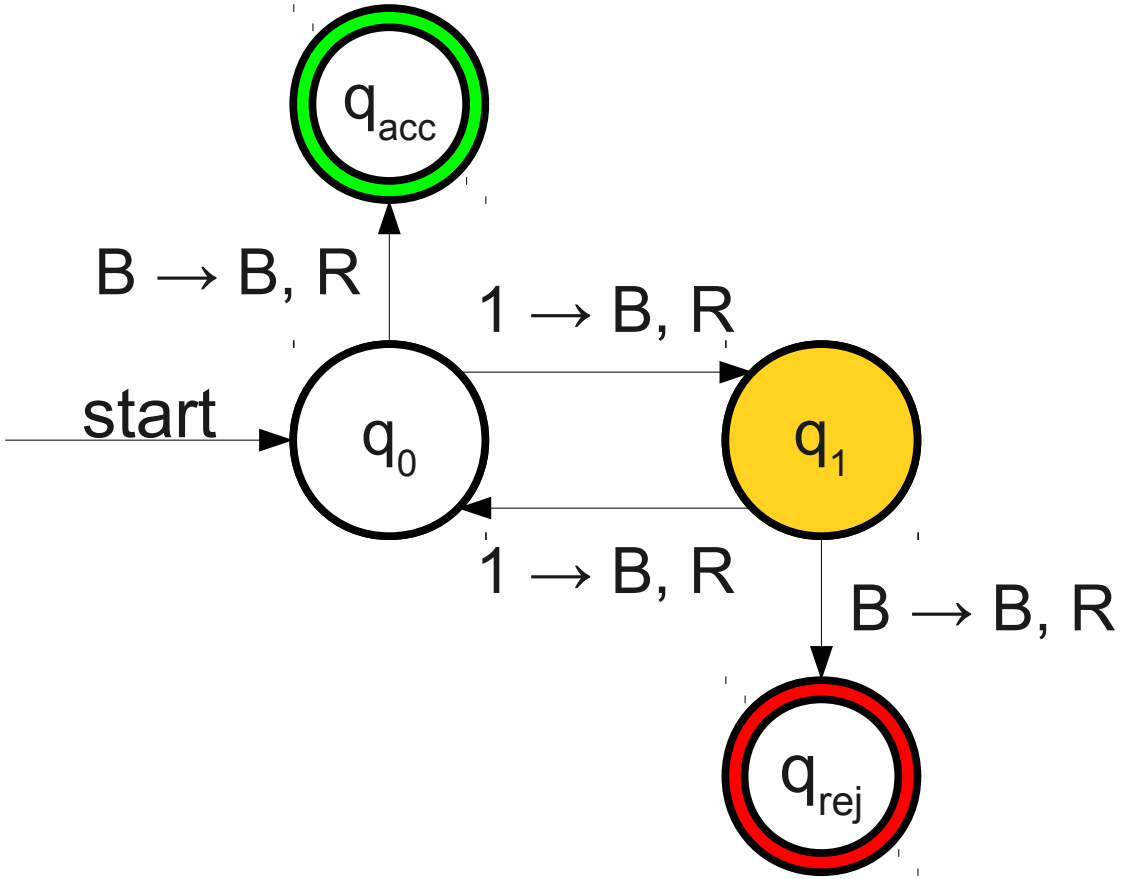


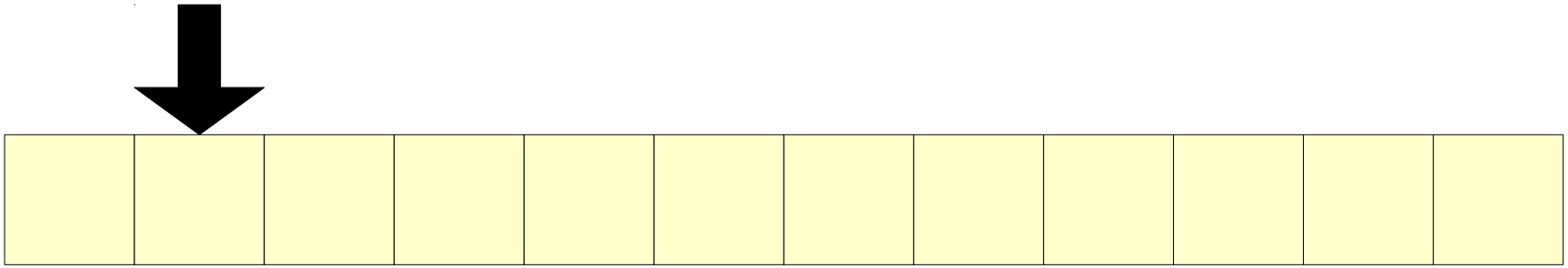
A Simple Turing Machine



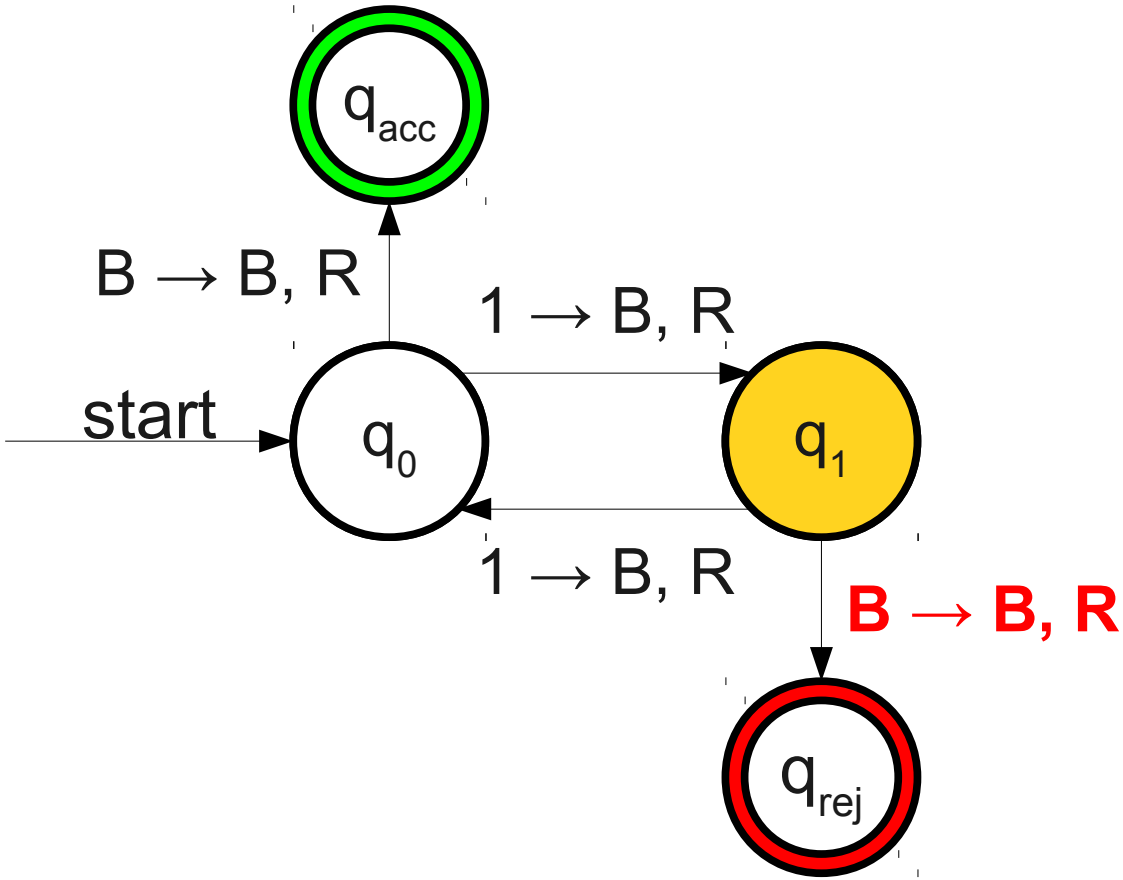


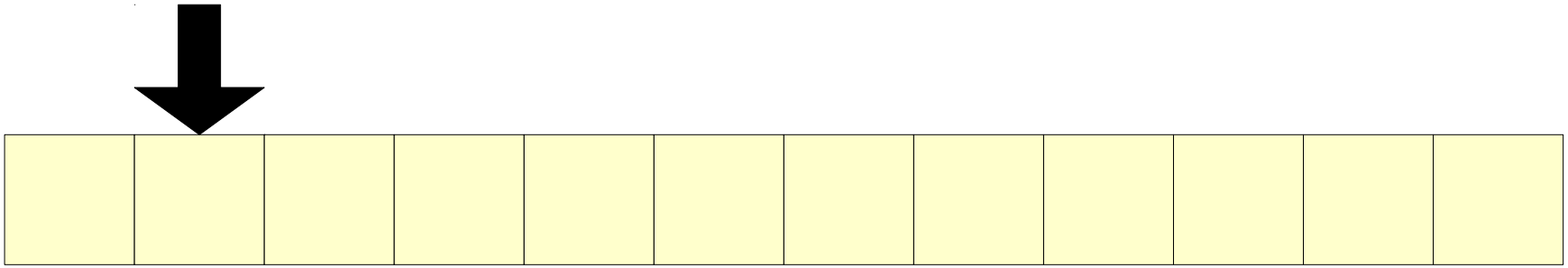
A Simple Turing Machine



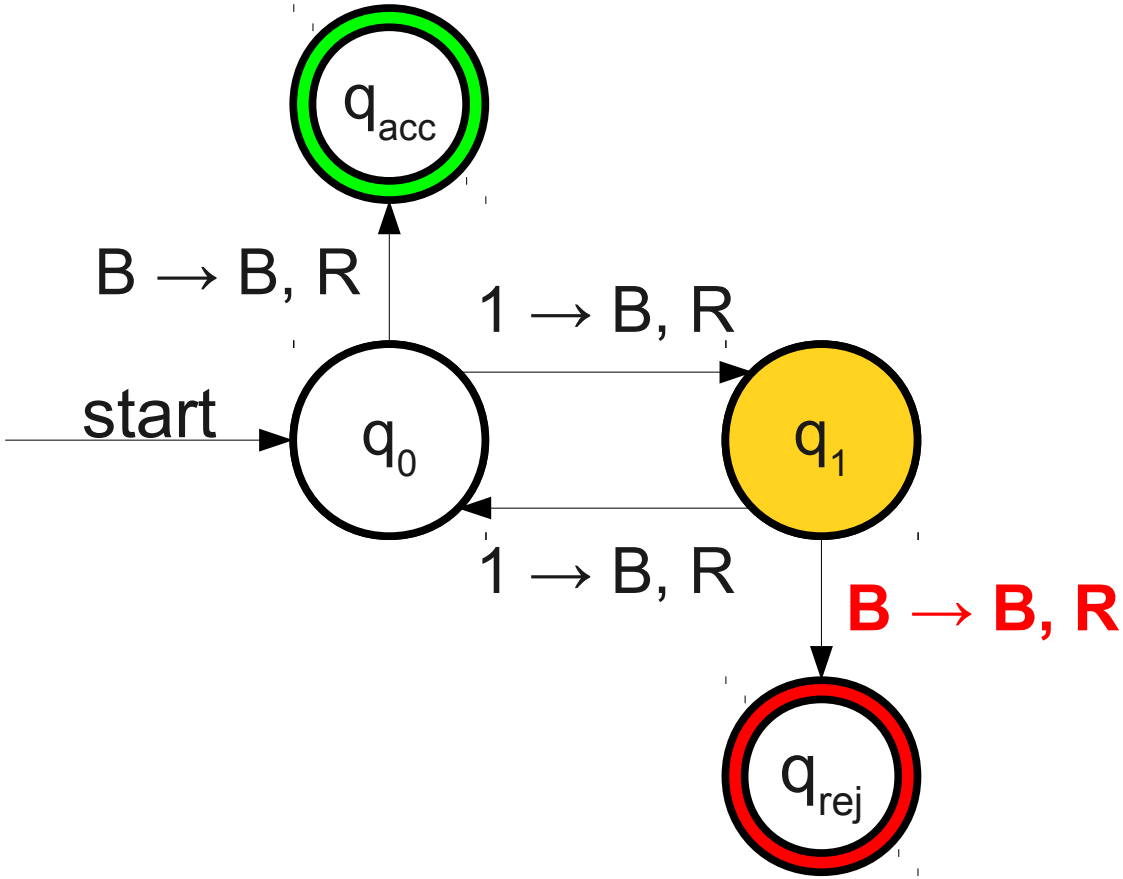


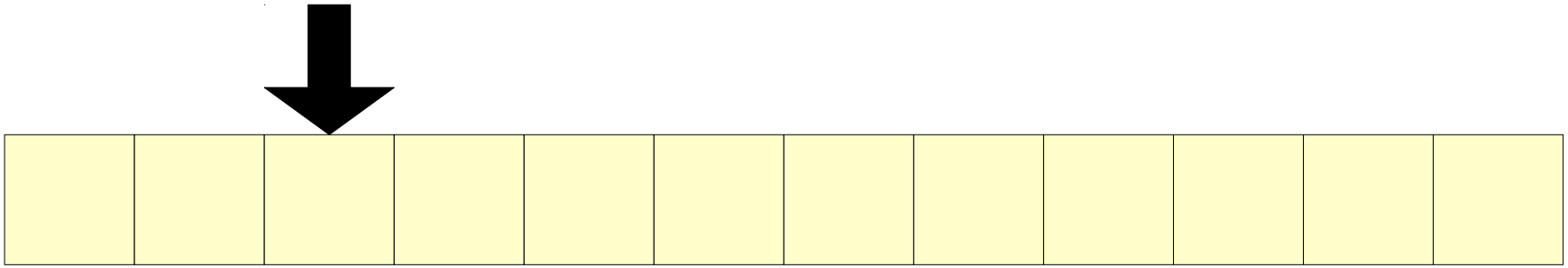
A Simple Turing Machine



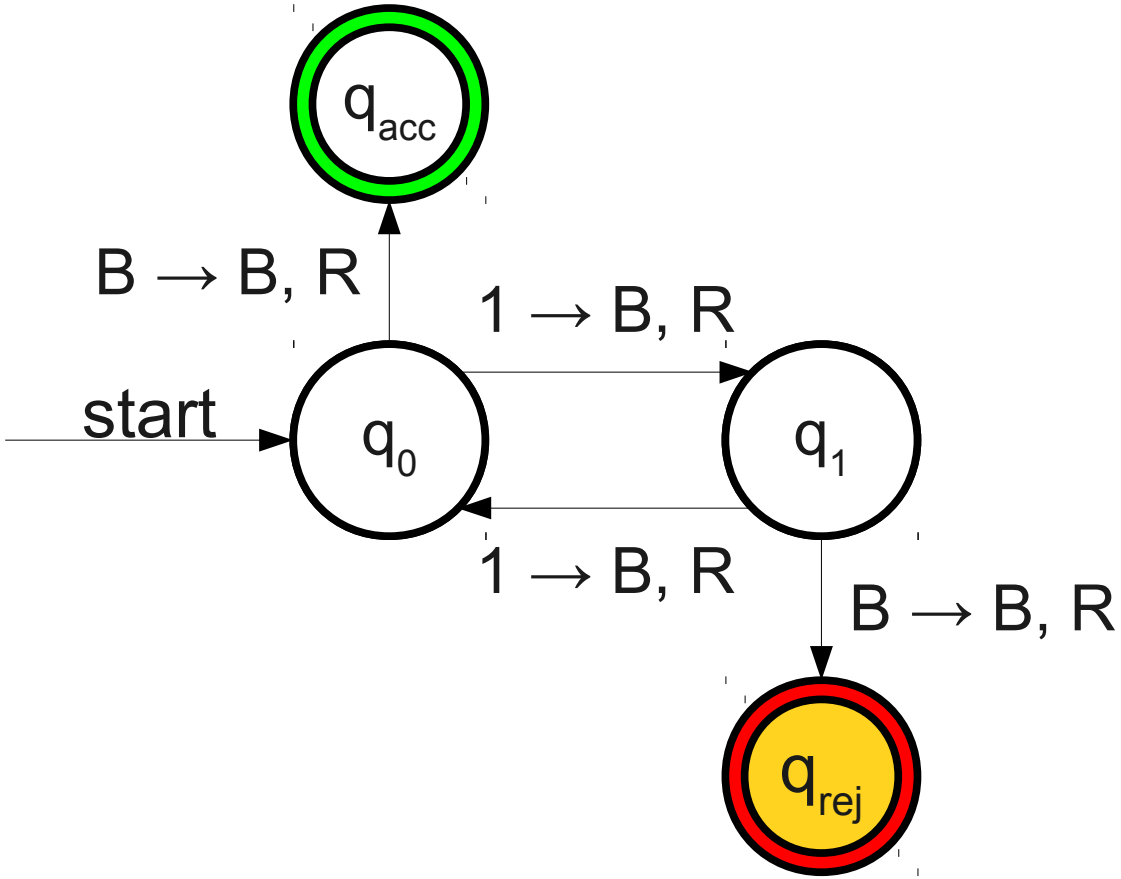


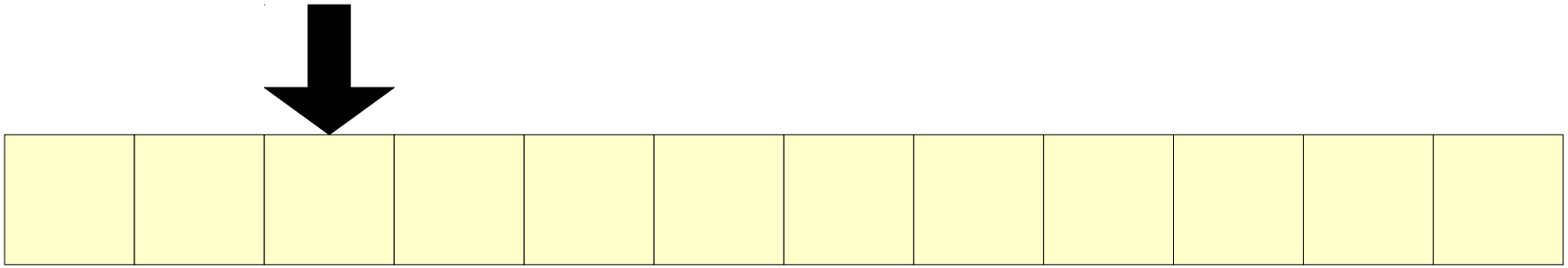
A Simple Turing Machine



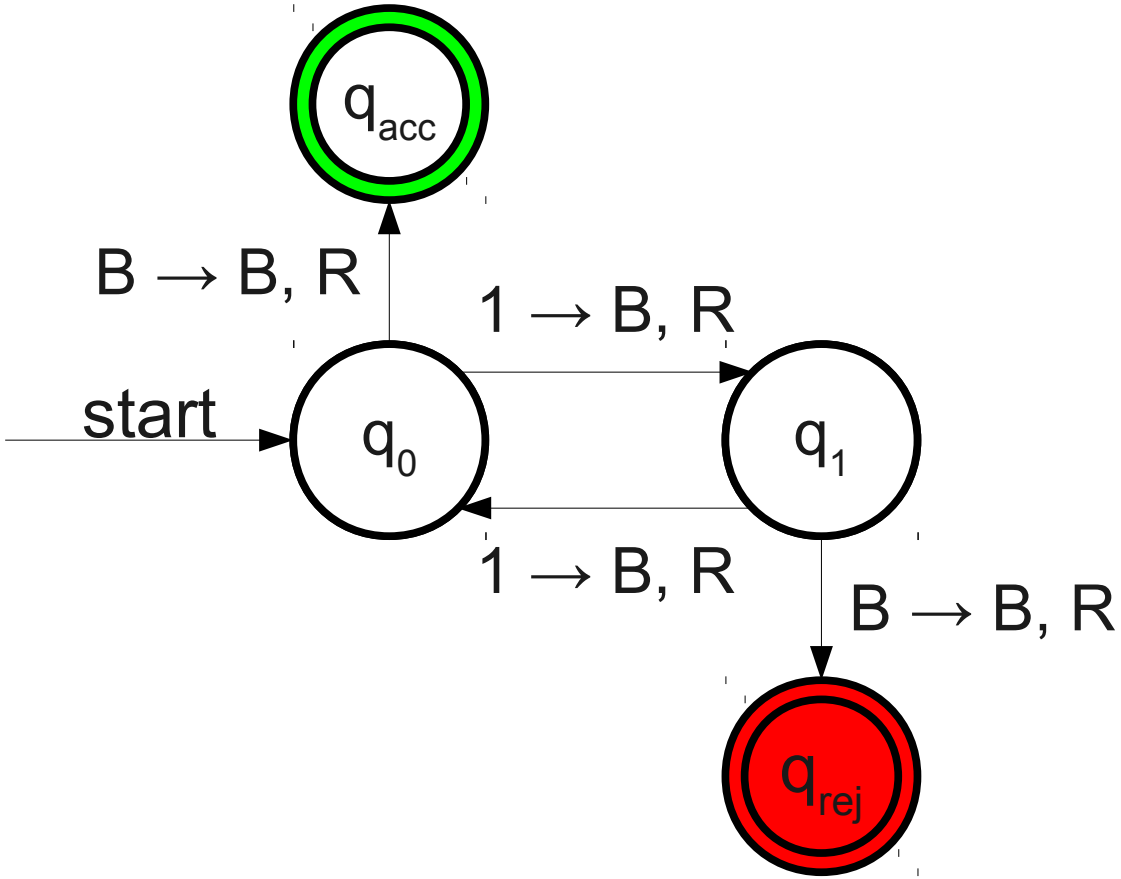


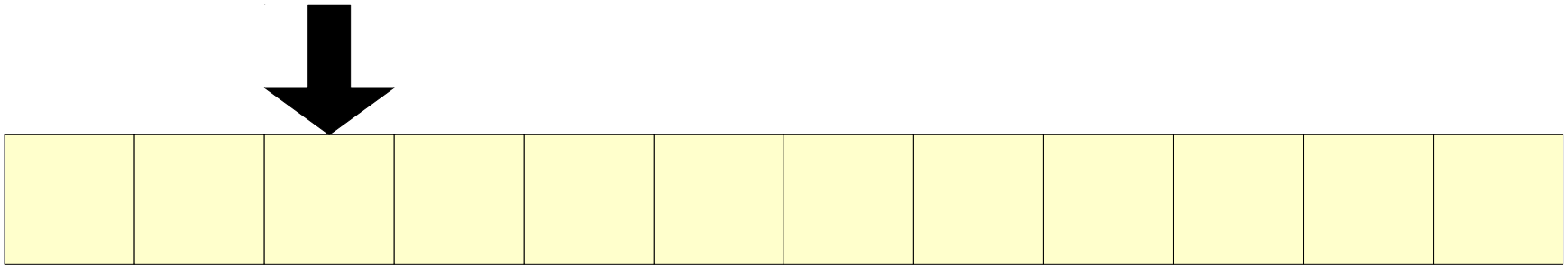
A Simple Turing Machine



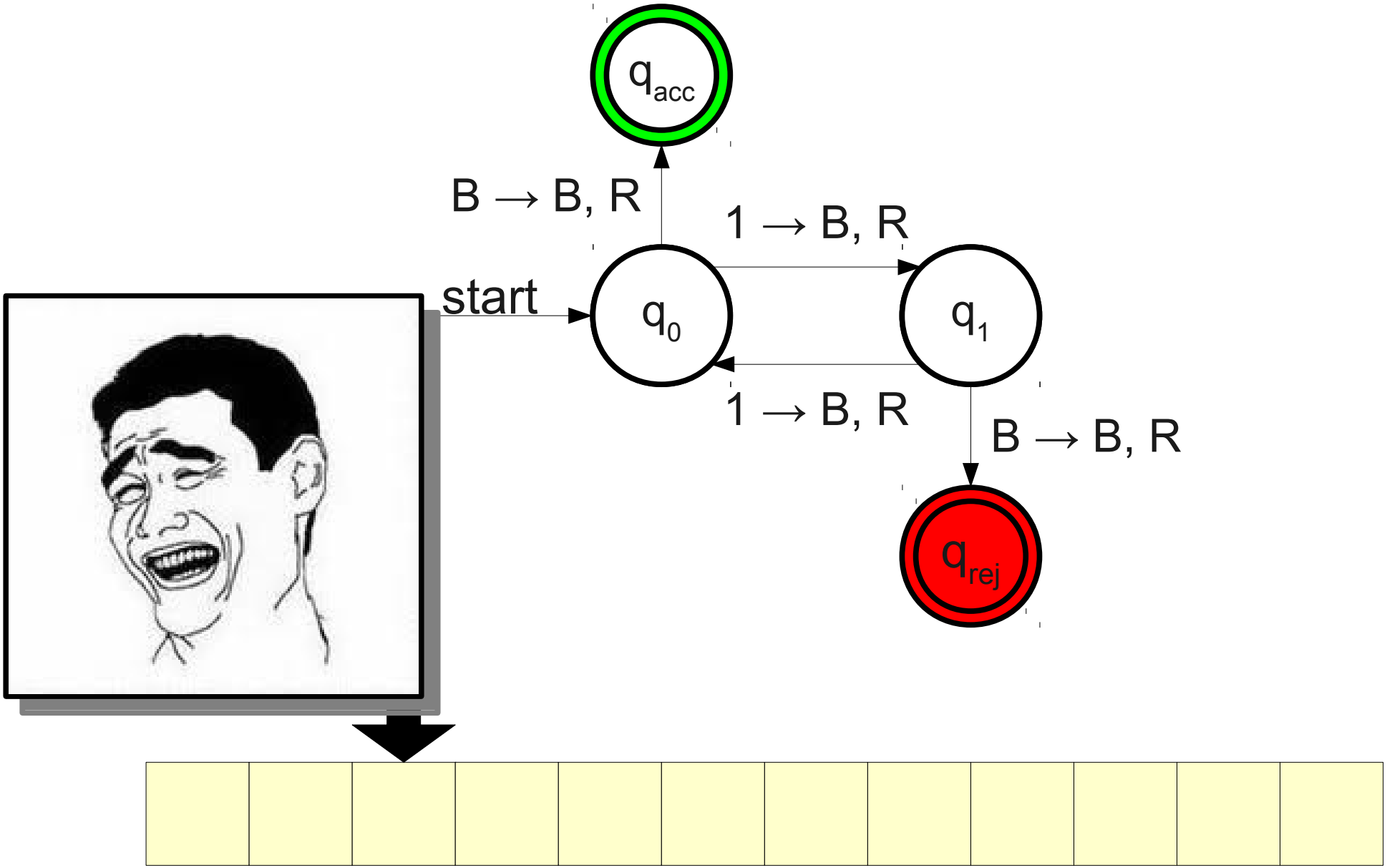


A Simple Turing Machine





A Simple Turing Machine



Acceptance

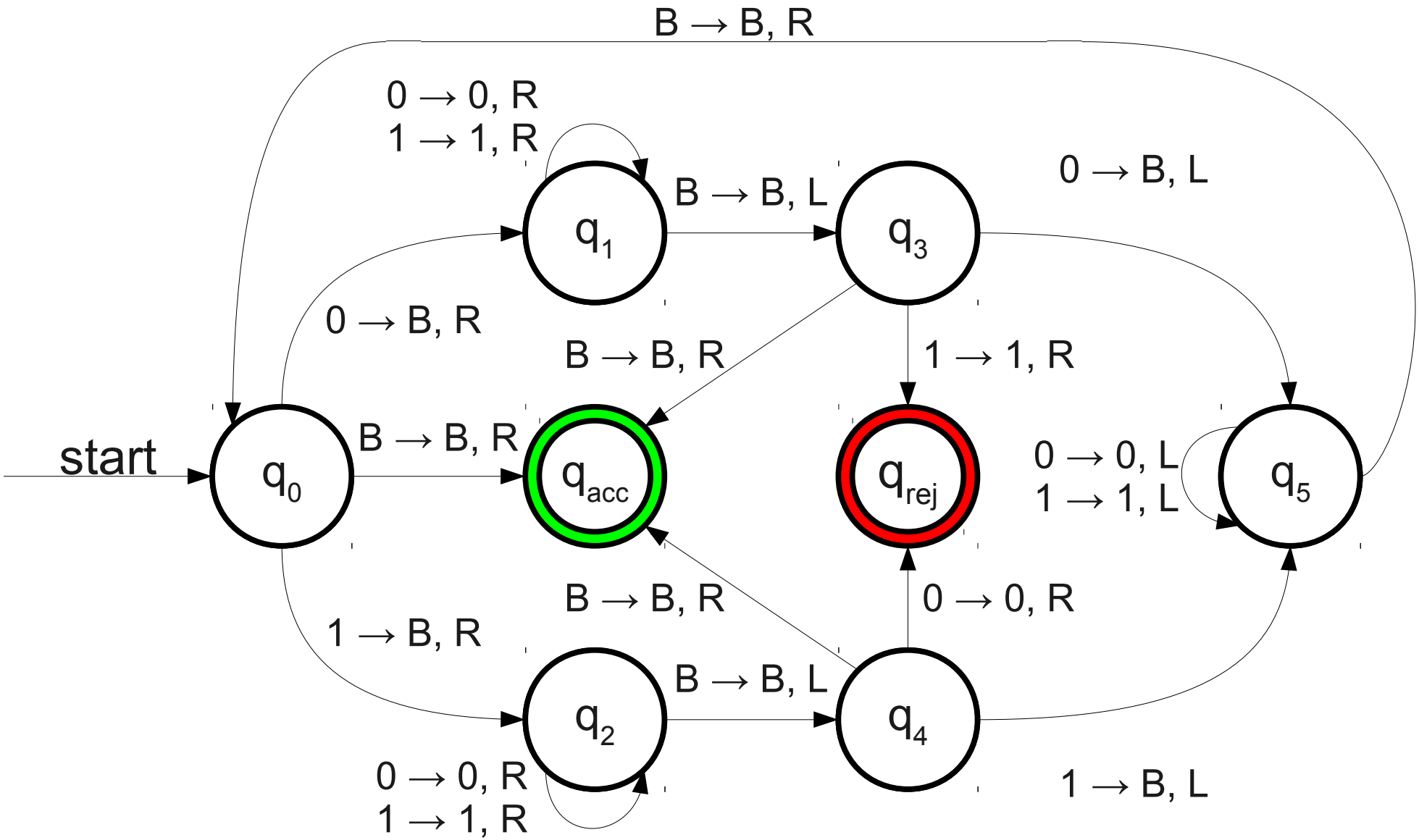
- Unlike the automata that we've seen before, the Turing machine can revisit characters from the input.
- The machine decides when it terminates, rather than stopping when no input is left.
- The Turing machine accepts if it enters a special **accept state**. It rejects if it enters a special **reject state**.
- **Turing machines can loop forever**.
  - More on that later...

A More Powerful Turing Machine

- Let $\Sigma = \{$`0`, `1`$\}$ and let

  $PALINDROME = \{ w \in \Sigma^* \mid w \text{ is a palindrome} \}$

- We can build a TM for *PALINDROME* as follows:
  - Look at the leftmost character of the string.
  - Scan across the tape until we find the end of the string.
  - If the last character doesn't match, reject the input.
  - Sweep back to the left of the tape and repeat.
  - If every character becomes matched, accept.


B → B, R
0 → 0, R
1 → 1, R
B → B, L
0 → B, L
0 → B, R
B → B, R
1 → 1, R
start
B → B, R
0 → 0, L
1 → 1, L
B → B, R
0 → 0, R
1 → B, R
B → B, L
0 → 0, R
1 → 1, R
1 → B, L
q_0
q_1
q_2
q_3
q_4
q_5
q_{acc}
q_{rej}

A More Sane Representation

	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation

0	0	1	0	0							

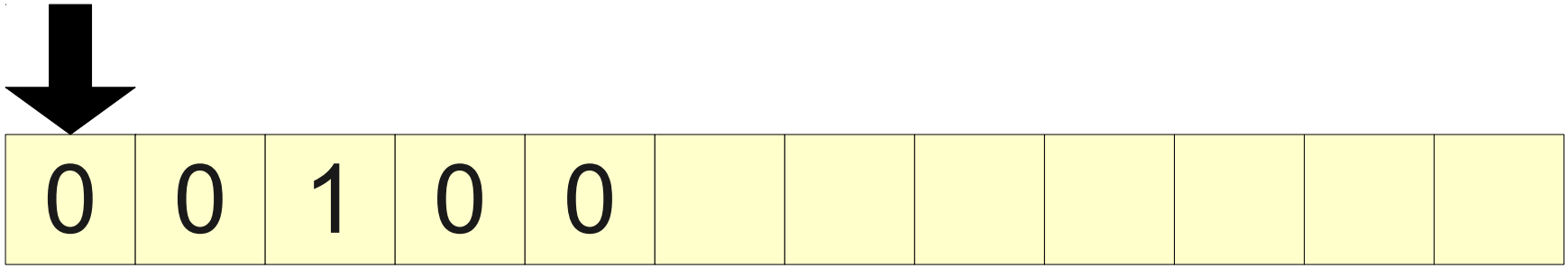
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation

0	0	1	0	0							

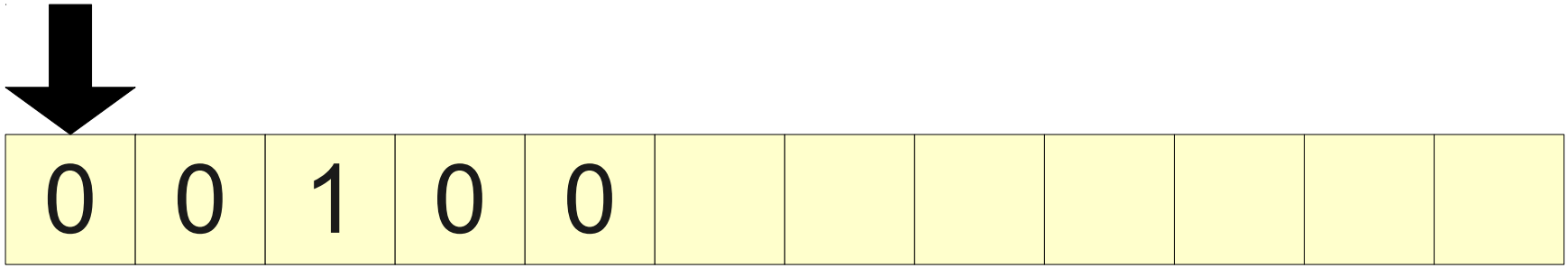
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



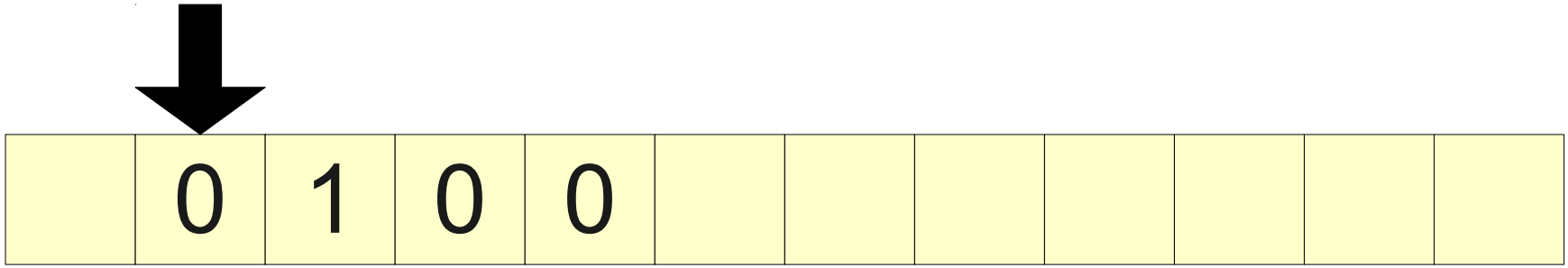
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



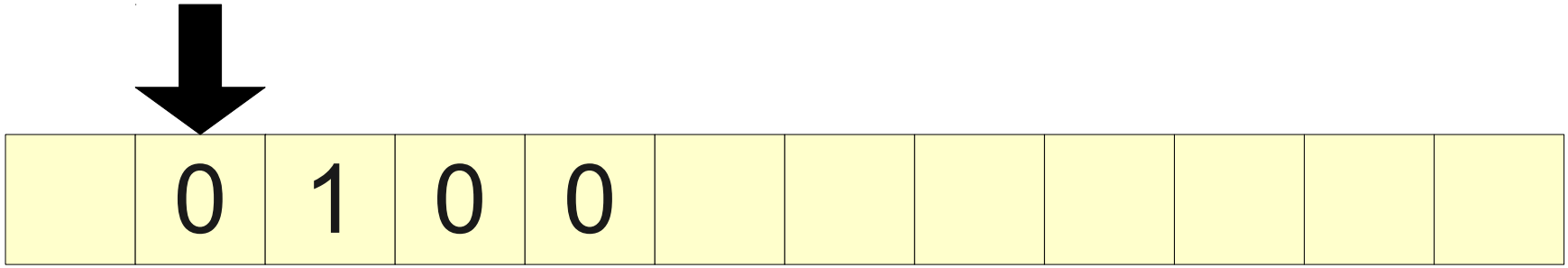
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



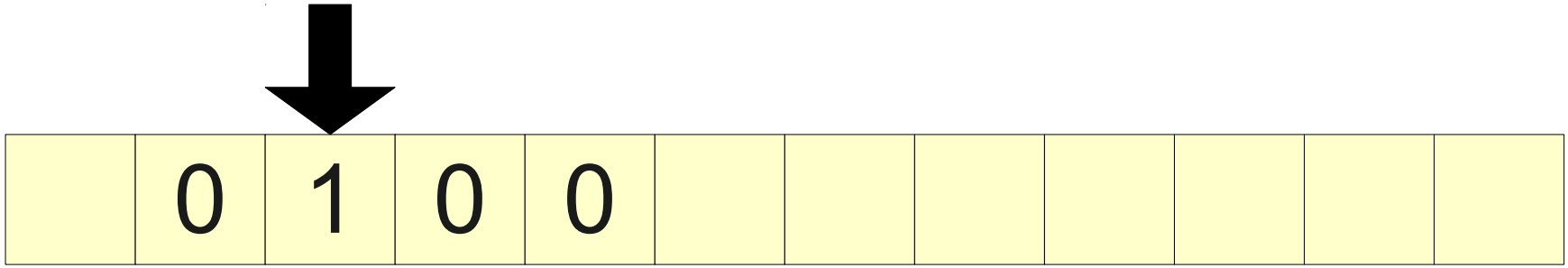
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



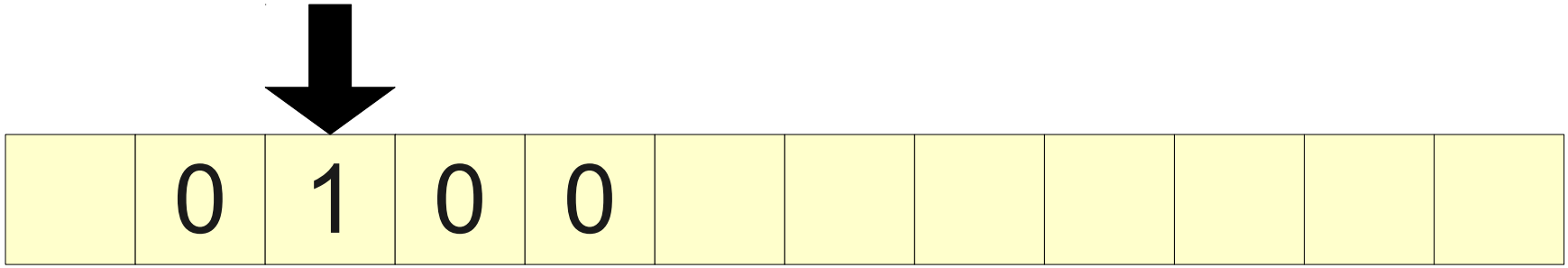
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



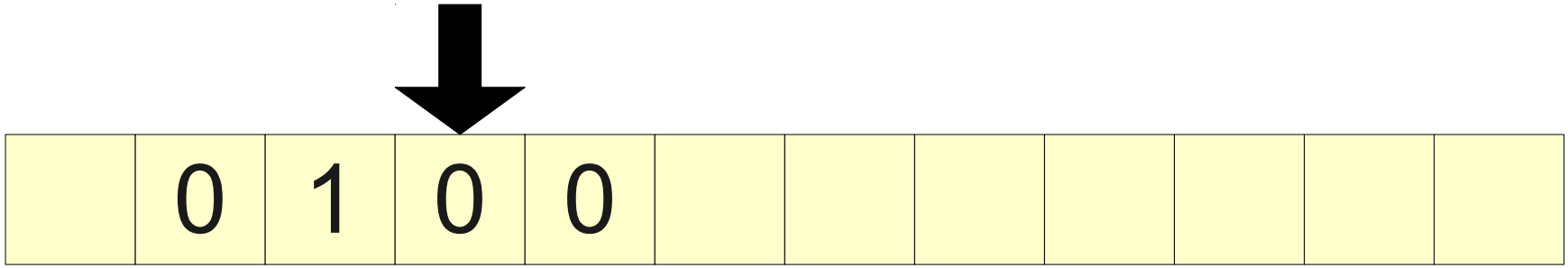
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



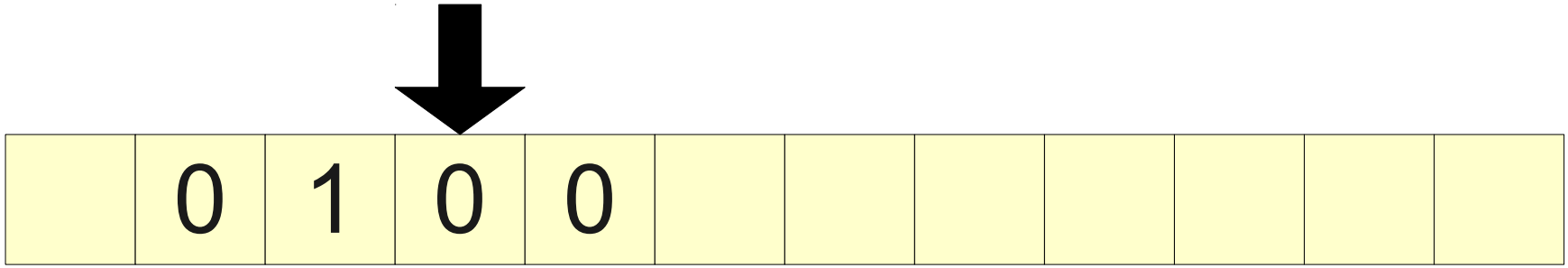
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



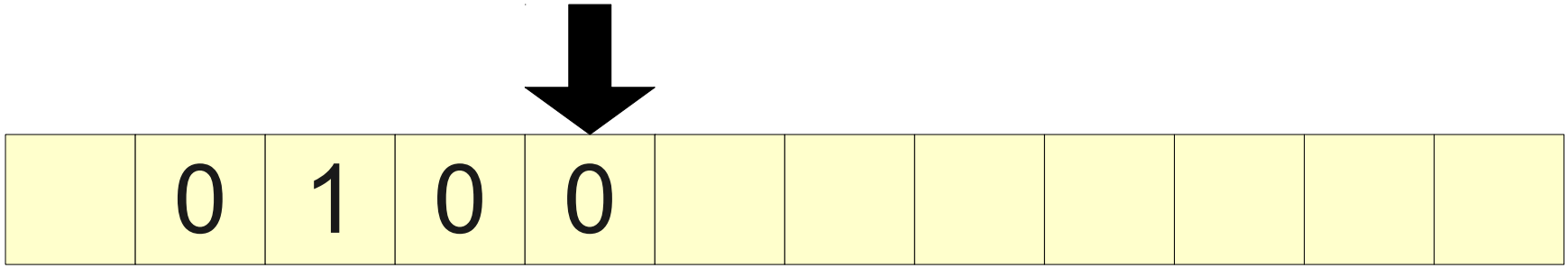
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



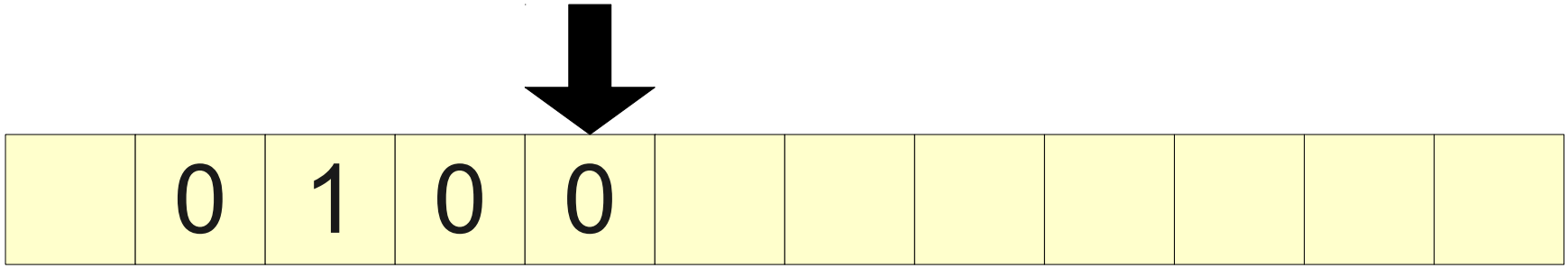
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



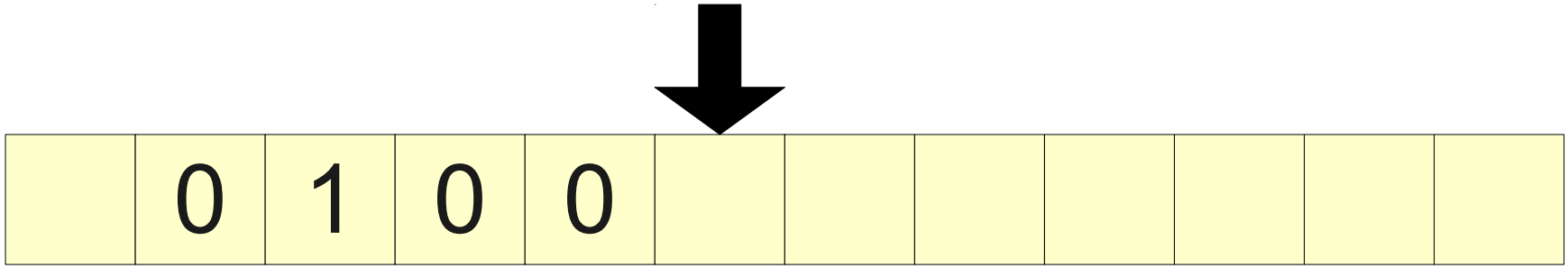
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



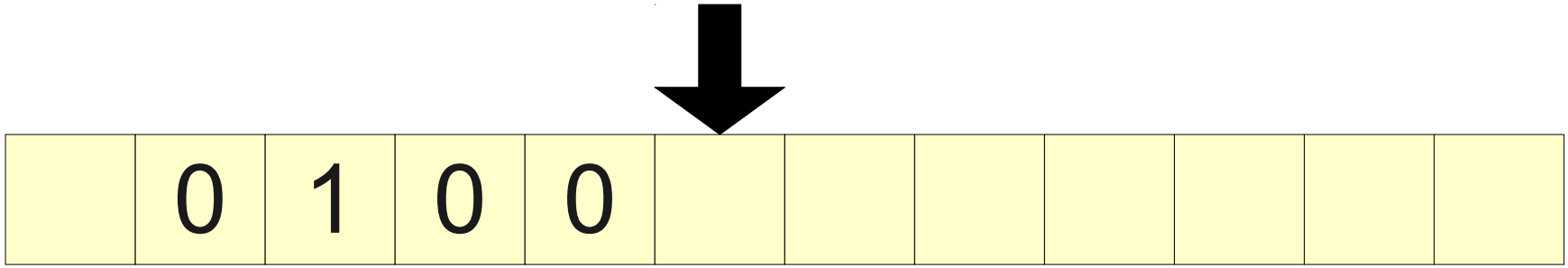
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



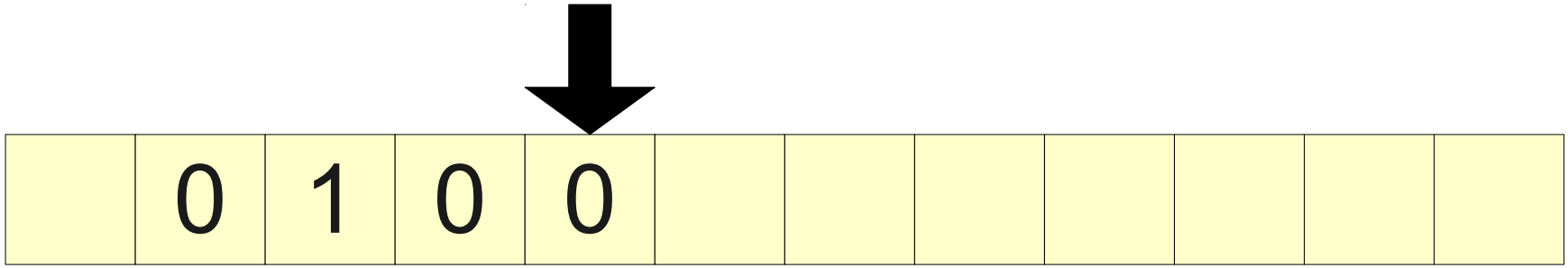
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



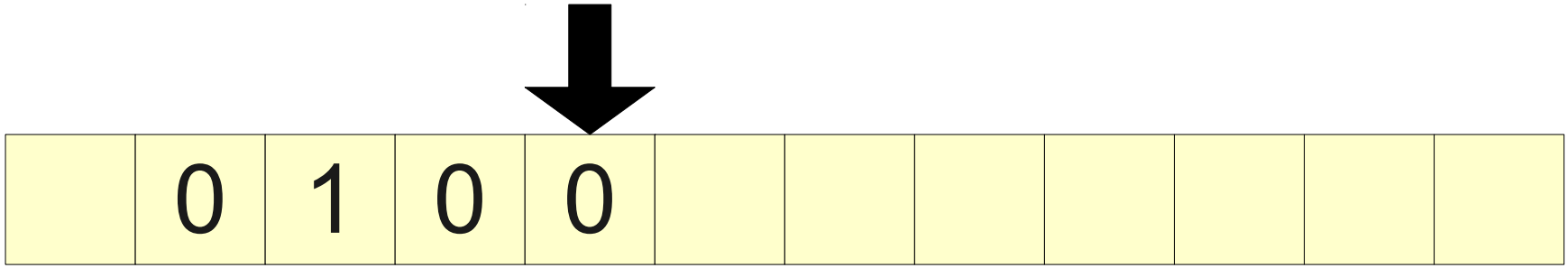
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



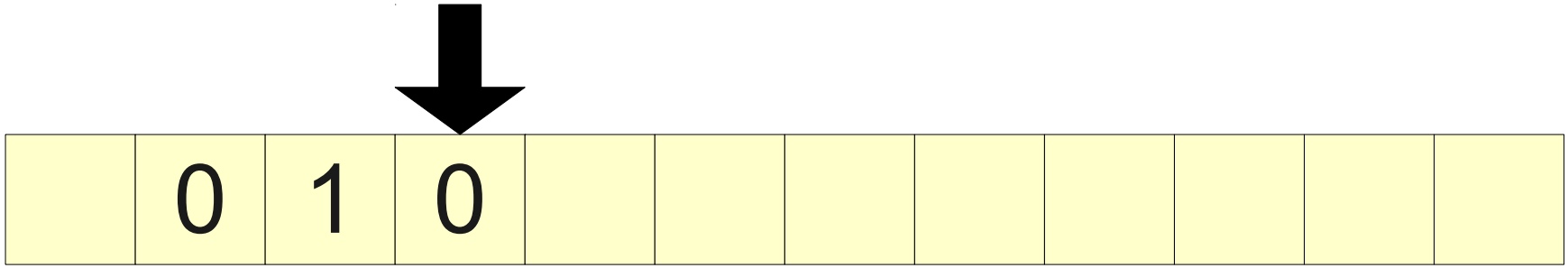
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



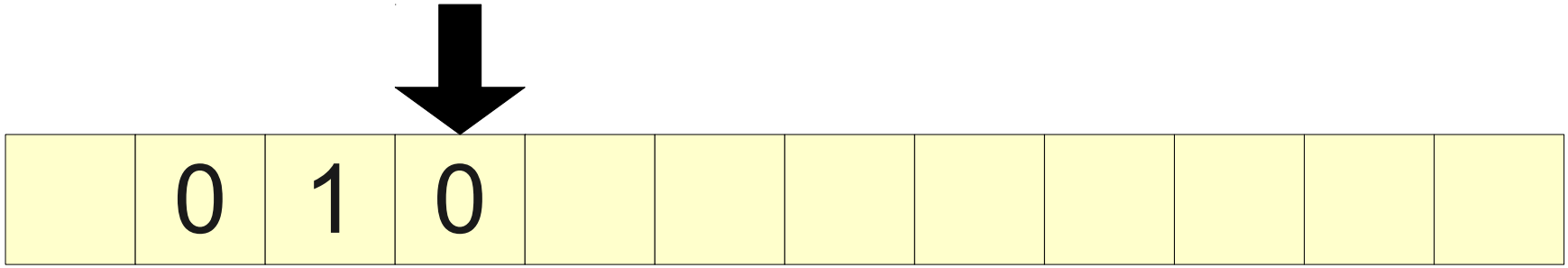
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



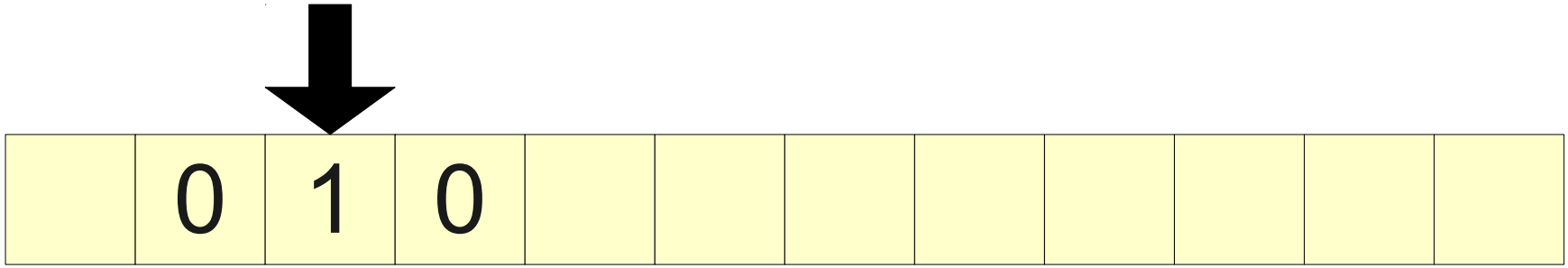
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



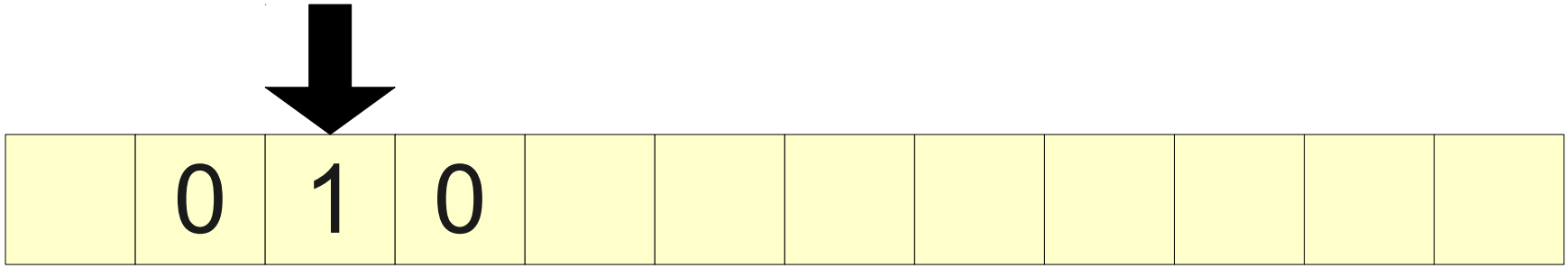
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



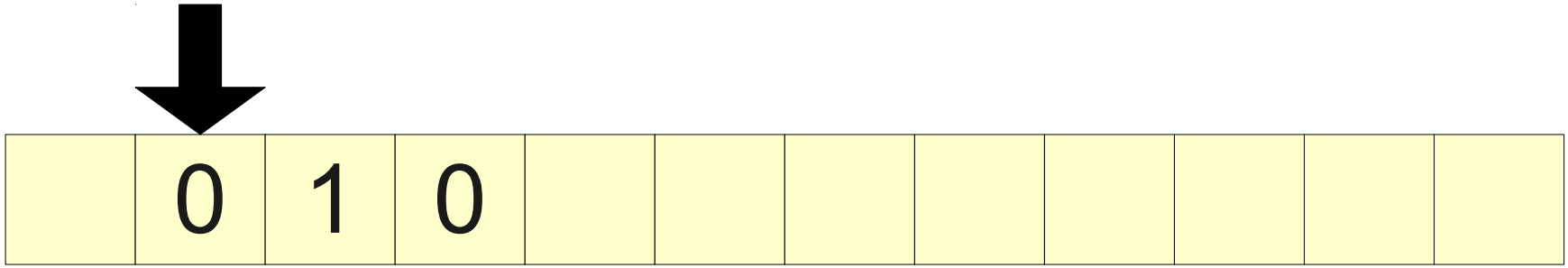
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



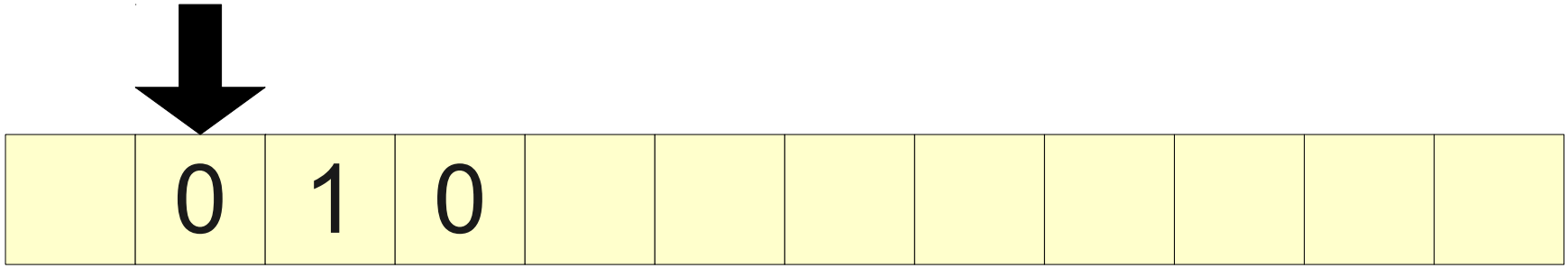
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



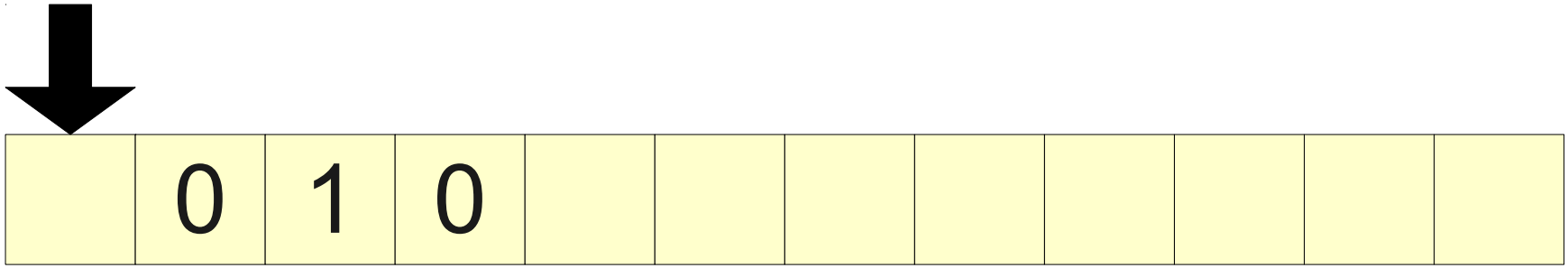
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



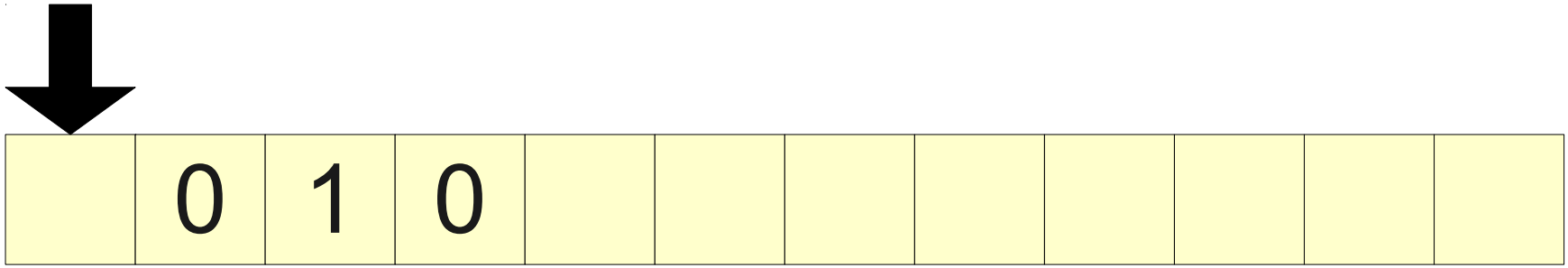
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



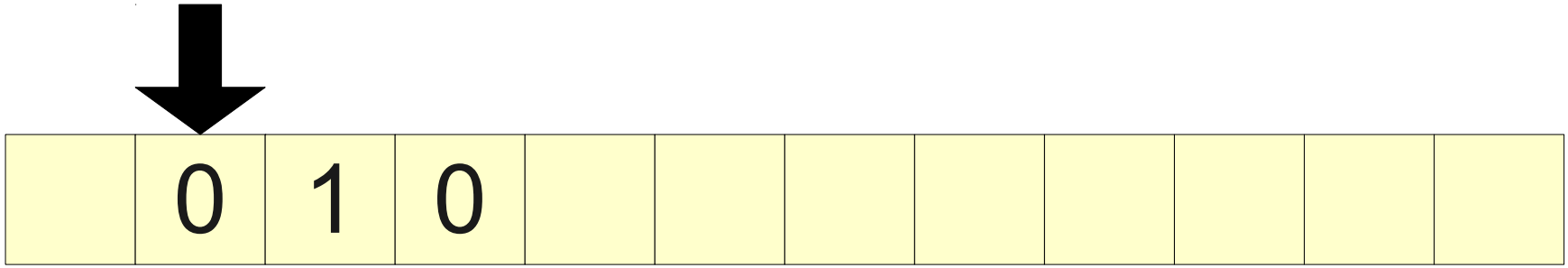
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



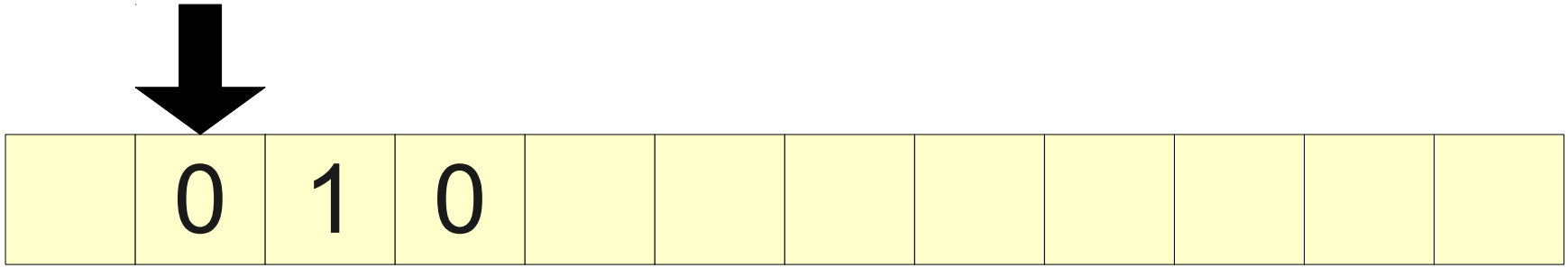
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



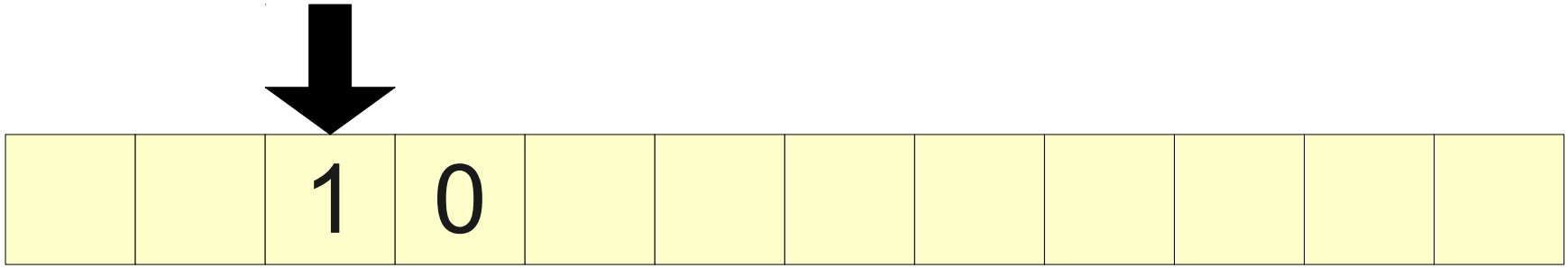
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



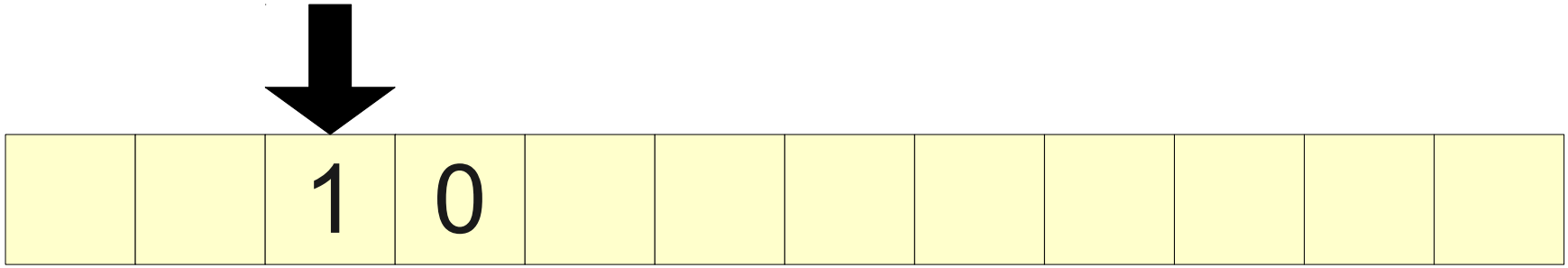
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



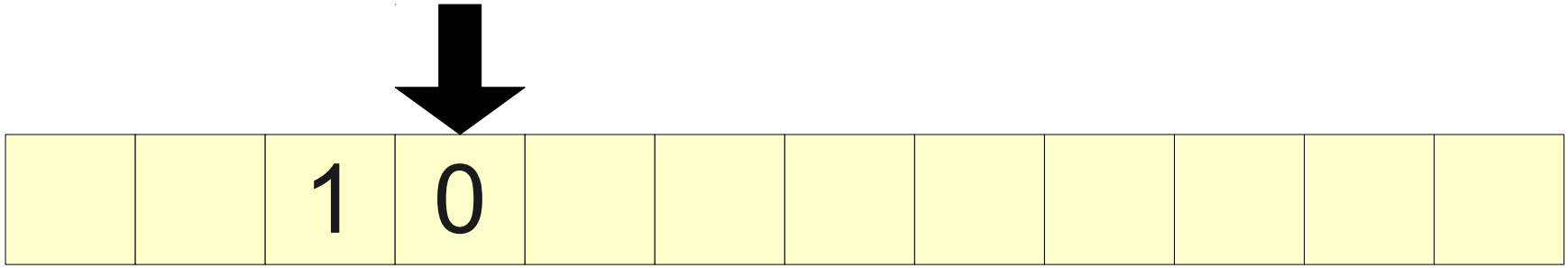
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



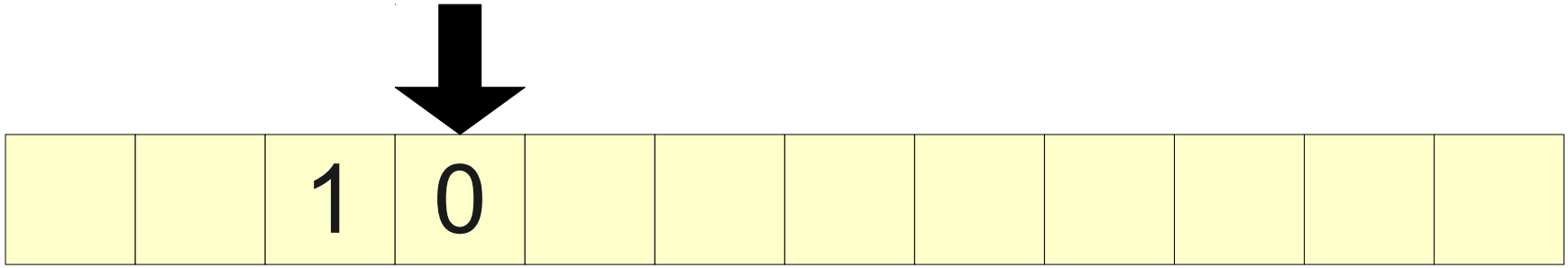
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



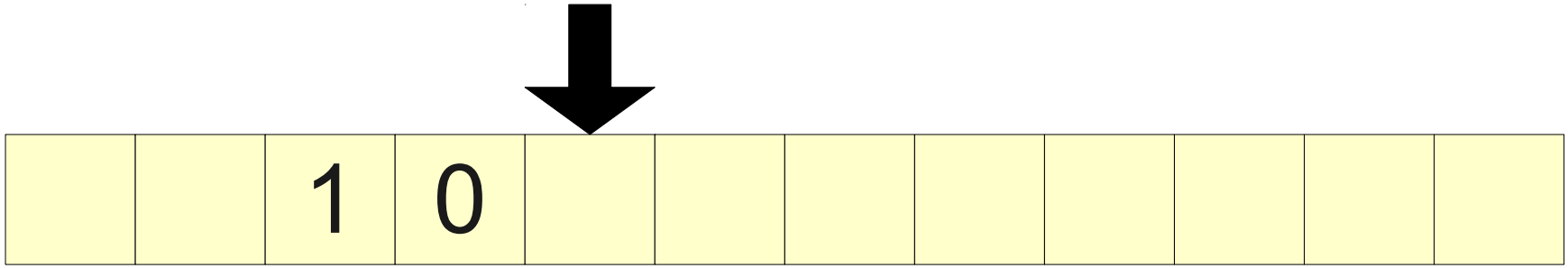
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



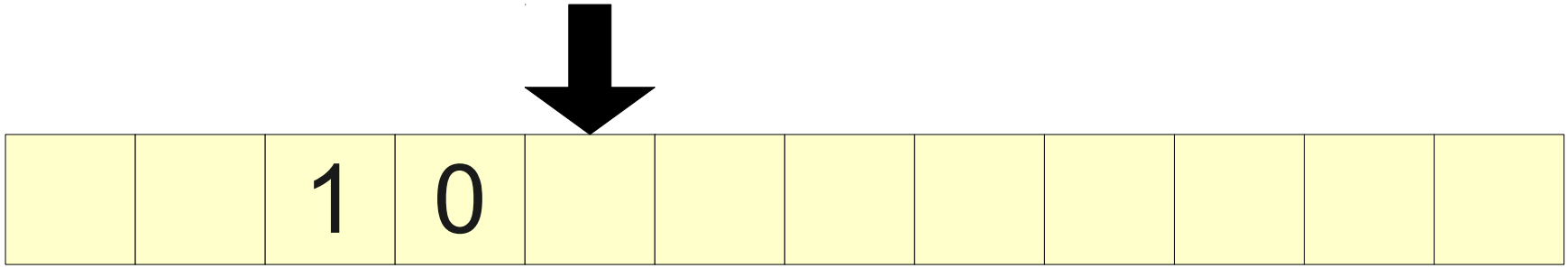
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



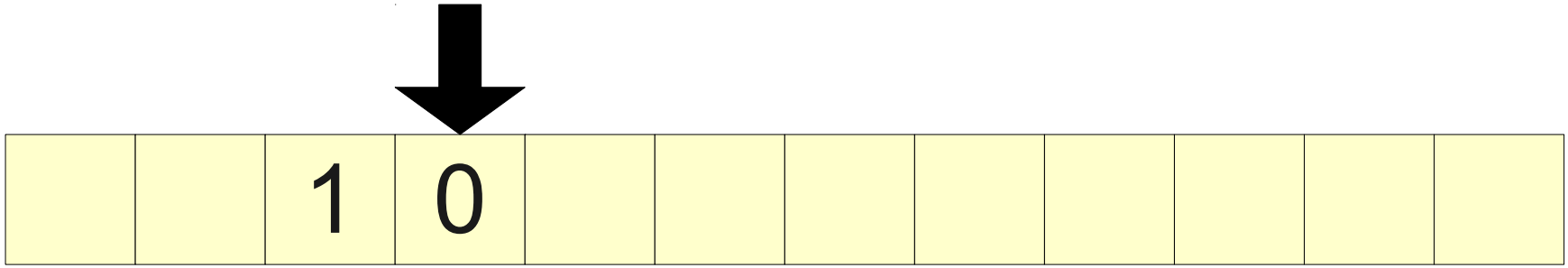
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



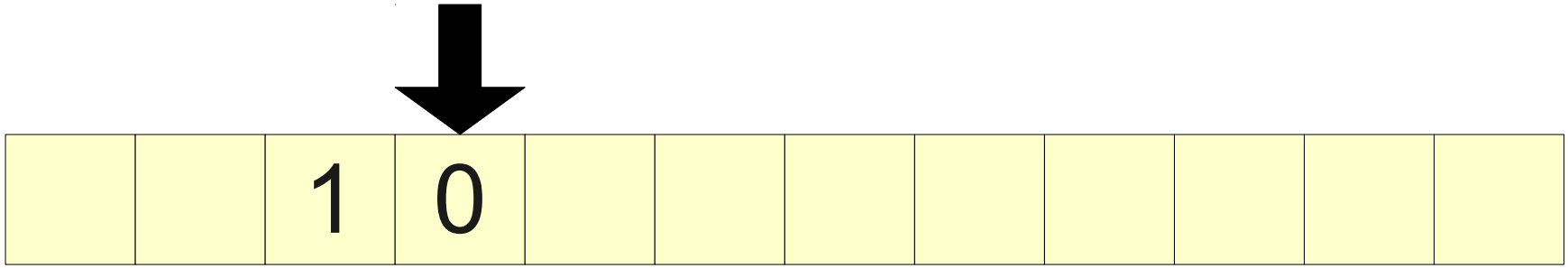
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



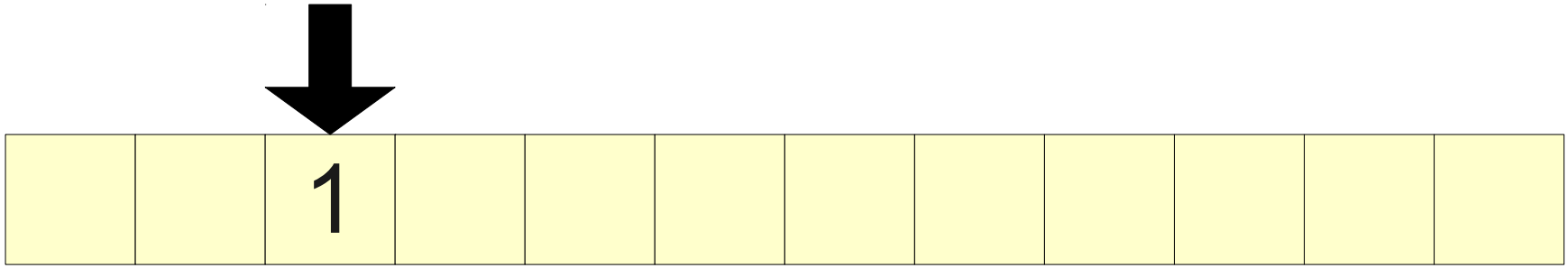
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



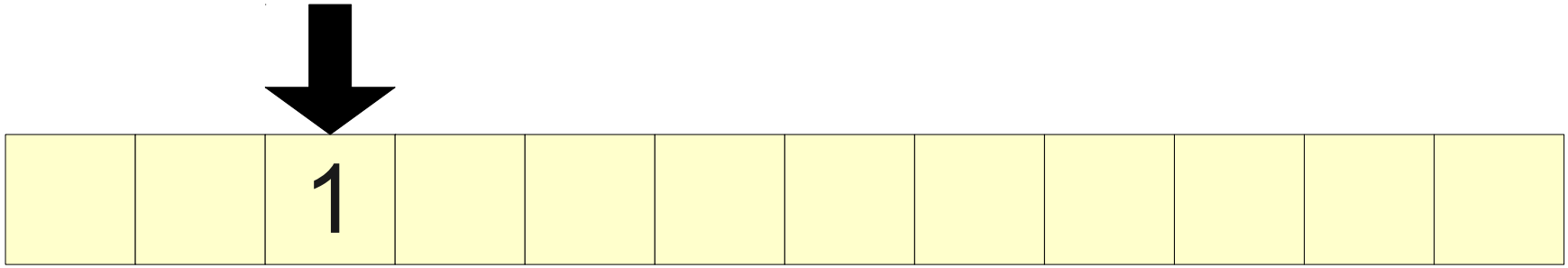
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



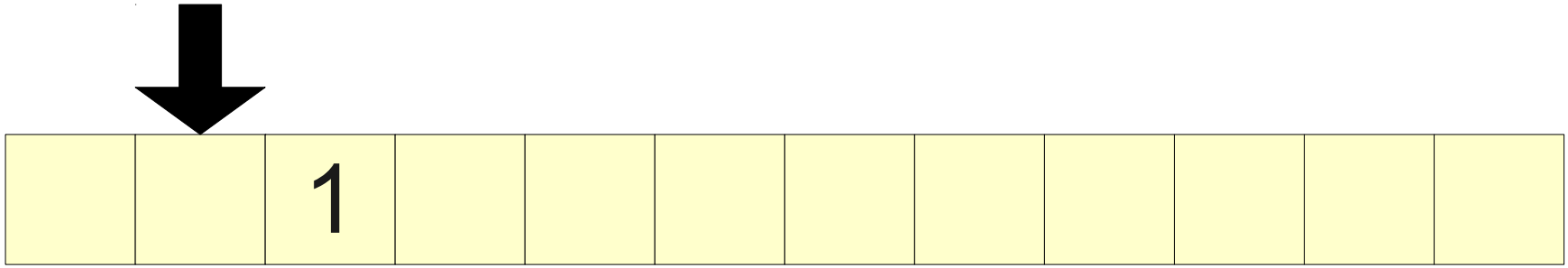
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



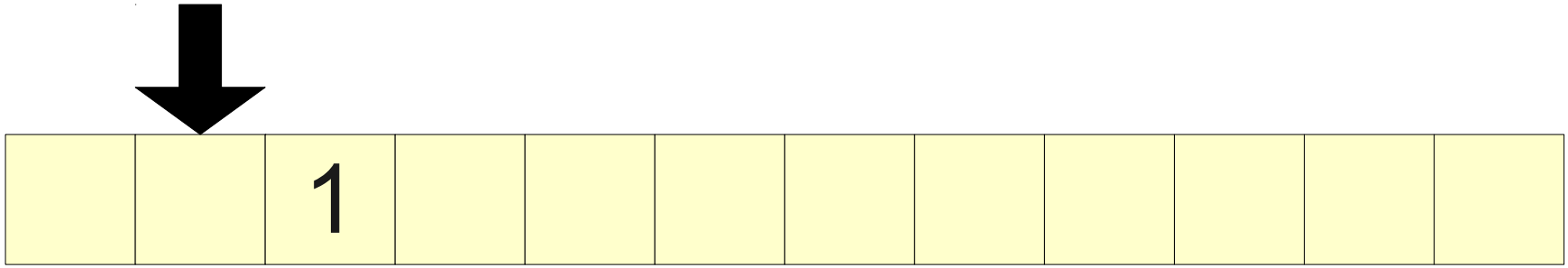
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



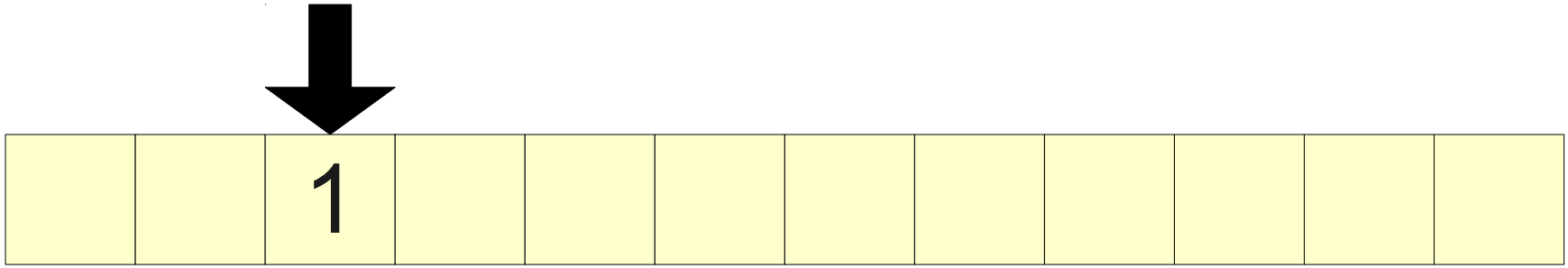
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



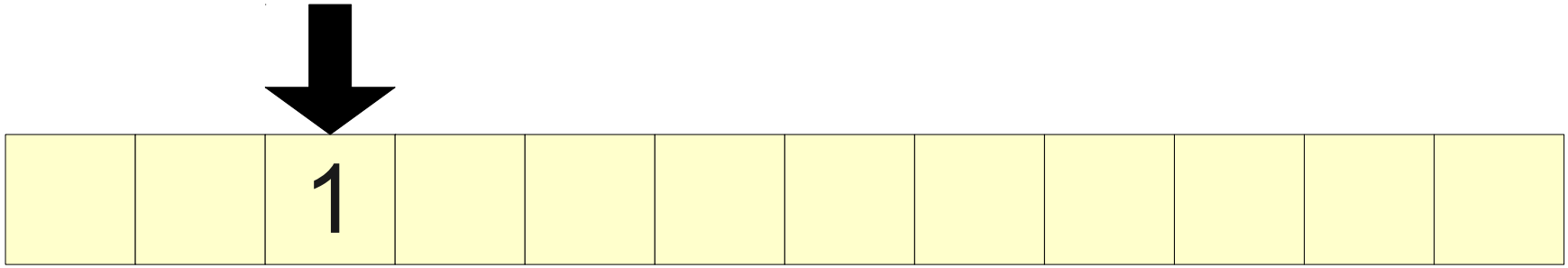
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



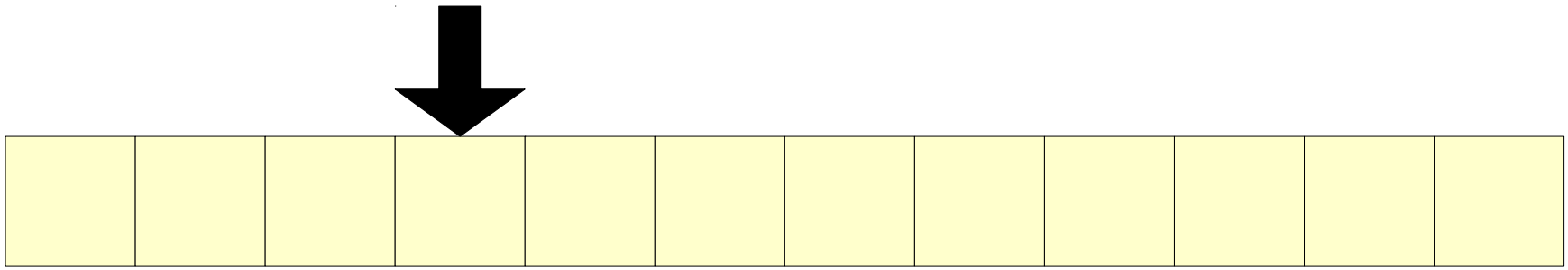
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



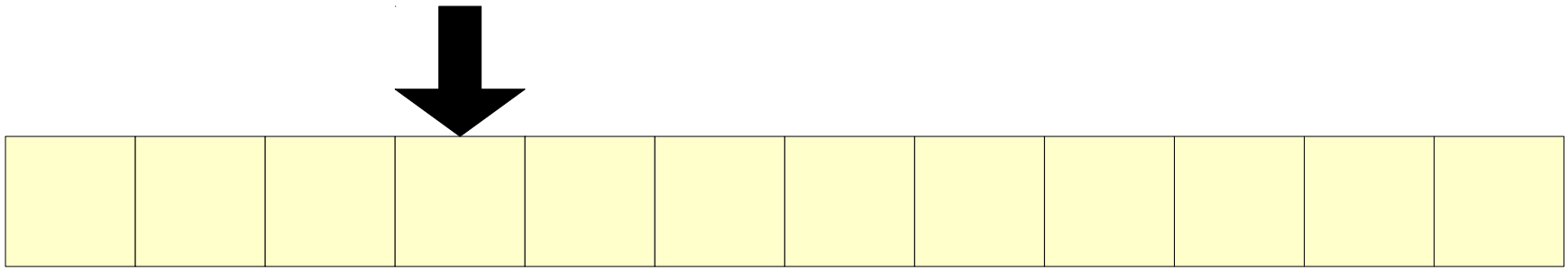
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



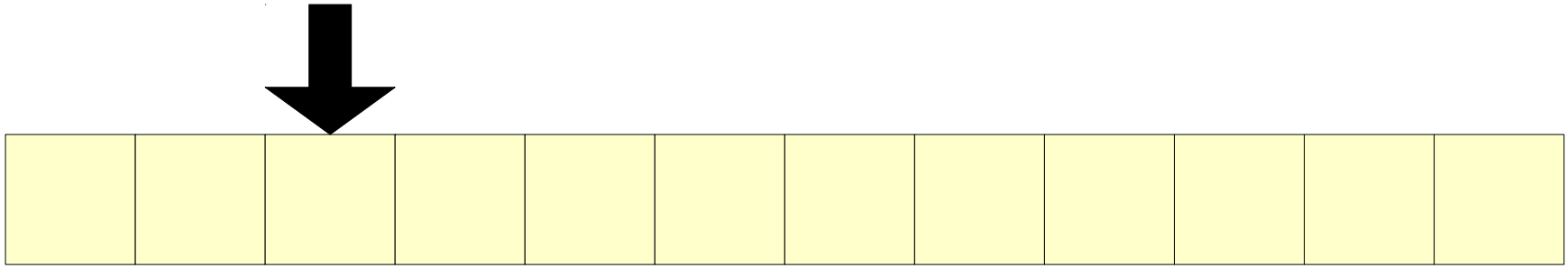
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



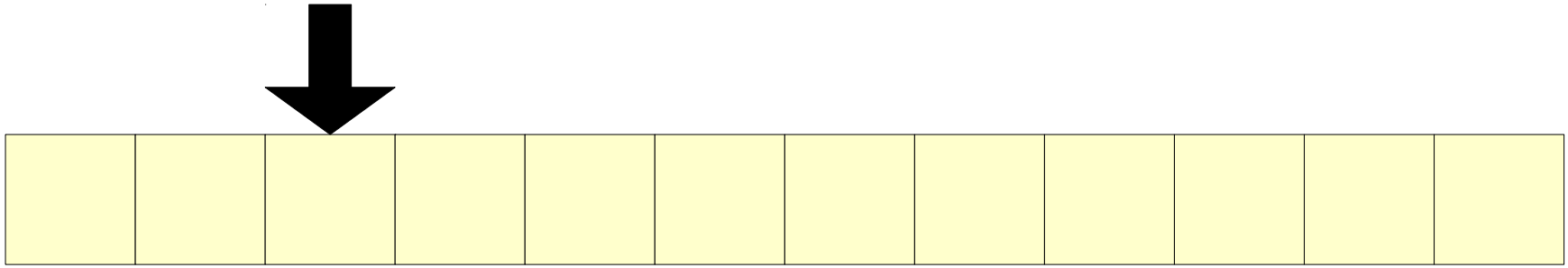
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



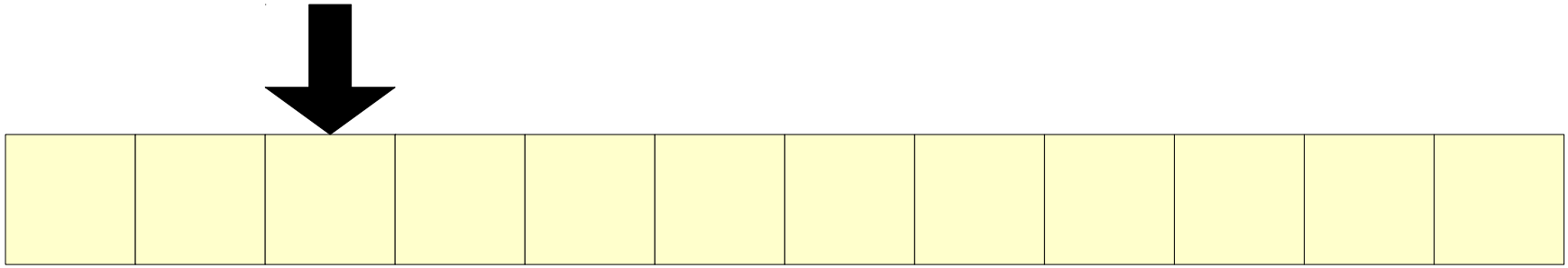
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



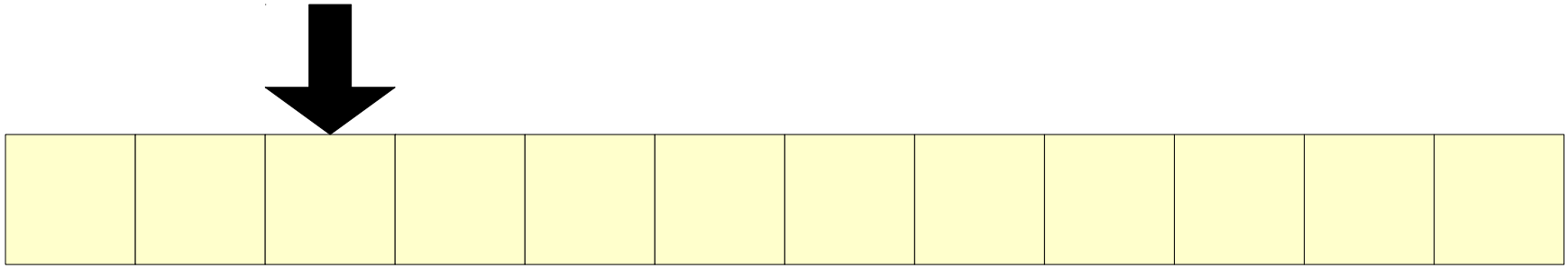
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



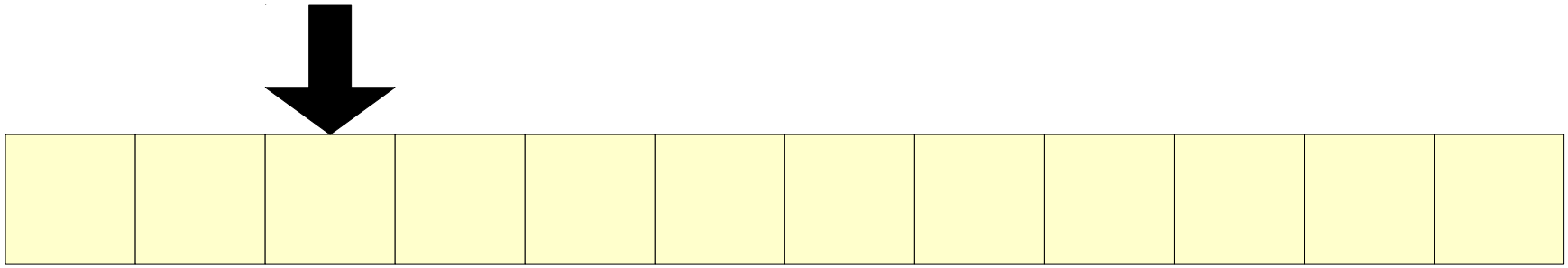
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



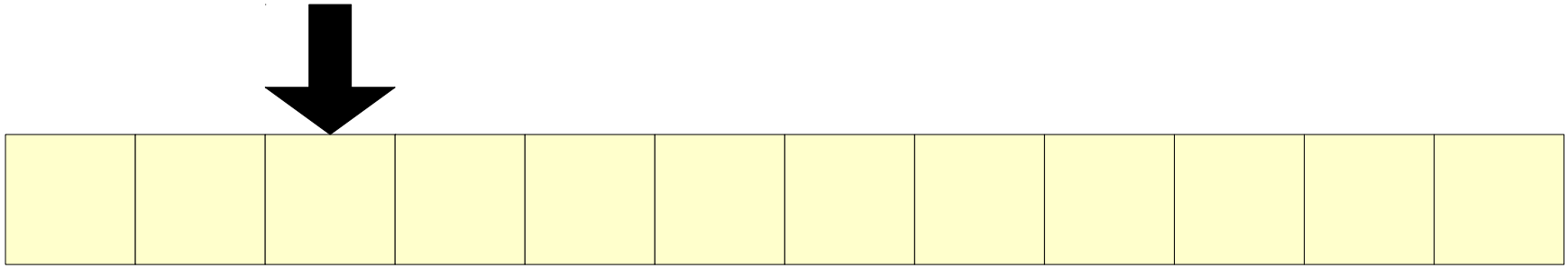
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



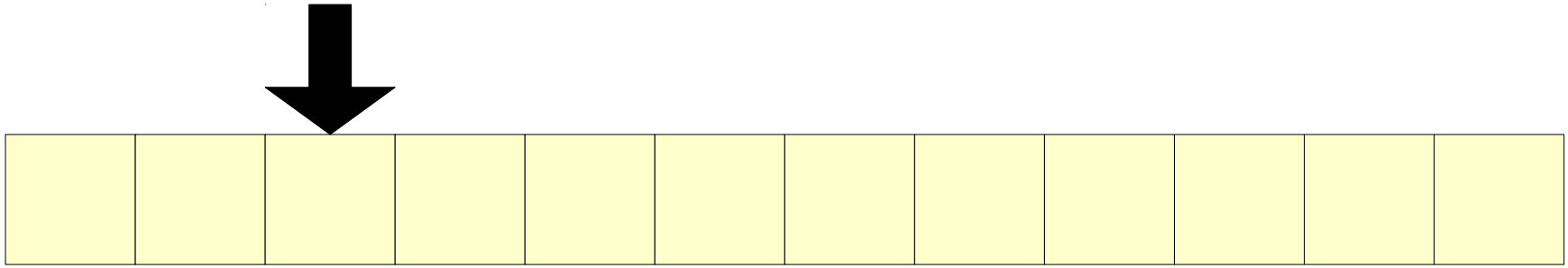
	0			1			B		
q_0	B	R	q_1	B	R	q_2	B	R	q_{acc}
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	1	R	q_{rej}	B	R	q_{acc}
q_4	0	R	q_{rej}	B	L	q_5	B	R	q_{acc}
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



	0			1			B		
q_0	B	R	q_1	B	R	q_2	Accept		
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	Reject			Accept		
q_4	Reject			B	L	q_5	Accept		
q_5	0	L	q_5	1	L	q_5	B	R	q_0

A More Sane Representation



	0			1			B		
q_0	B	R	q_1	B	R	q_2	Accept		
q_1	0	R	q_1	1	R	q_1	B	L	q_3
q_2	0	R	q_2	1	R	q_2	B	L	q_4
q_3	B	L	q_5	Reject			Accept		
q_4	Reject			B	L	q_5	Accept		
q_5	0	L	q_5	1	L	q_5	B	R	q_0

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,

Why must $\Sigma \subseteq \Gamma$?

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**

Each transition is based on the current tape symbol and state. Each transition maps to a new state, a new tape symbol, and a direction (either left or right).

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**,
  - $q_0 \in Q$ is the **start state**

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**,
  - $q_0 \in Q$ is the **start state**,
  - $q_{acc} \in Q$ is the **accept state**

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**,
  - $q_0 \in Q$ is the **start state**,
  - $q_{acc} \in Q$ is the **accept state**,
  - $q_{rej} \in Q$, $q_{rej} \neq q_{acc}$, is the **reject state**

Turing Machines, Formally

- A **Turing machine** is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where
  - Q is a finite set of states,
  - Σ is a finite **input alphabet**,
  - Γ is a finite **tape alphabet**, with $\Sigma \subseteq \Gamma$,
  - $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$ is the **transition function**,
  - $q_0 \in Q$ is the **start state**,
  - $q_{acc} \in Q$ is the **accept state**,
  - $q_{rej} \in Q$, $q_{rej} \neq q_{acc}$, is the **reject state**, and
  - $B \in \Gamma - \Sigma$ is the **blank symbol**.

The Language of a Turing Machine

- The **language of a TM** M is the set

  $$\mathscr{L}(\mathrm{M}) = \{\ w \in \Sigma^* \mid M \text{ enters } q_{acc} \text{ when run on } w\ \}$$

- If there is a TM M such that $\mathscr{L}(M) = L$, we say that L is **Turing-recognizable**.
  - "**Recognizable**" for short.
  - These languages are sometimes called **recursively enumerable**.
- Any regular language is recognizable (why?)
- Harder fact: Any context-free language is recognizable.

Programming Turing Machines

Programming Turing Machines

- Let's begin with a simple language over $\Sigma = \{\texttt{0}, \texttt{1}\}$:
- $BALANCE = \{ w \in \Sigma^* \mid w$ contains the same number of `0`s and `1`s $\}$
- How might we build a TM for *BALANCE*?

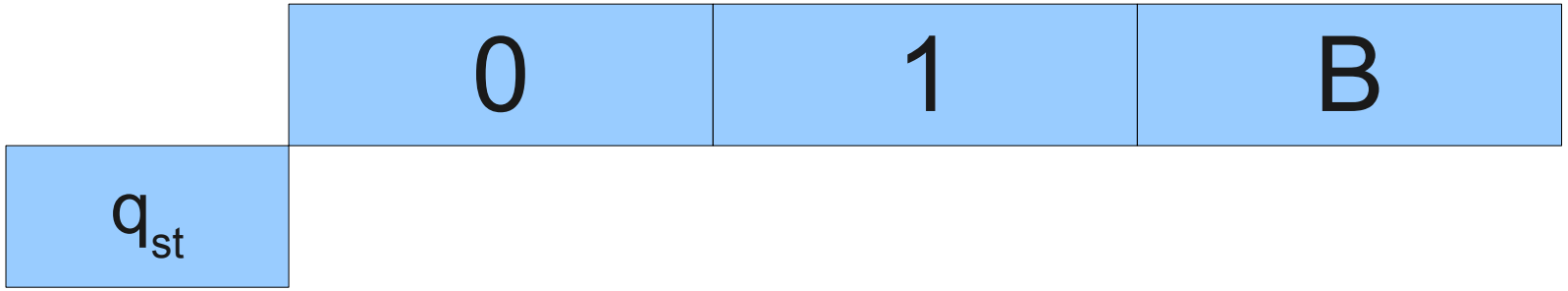
The Intuition

- Match the first symbol on the tape with the next available symbol that matches it.
- Match the first symbol on the tape with the next available symbol that matches it.
- Repeat until no symbols are left.
- If everything matches, we're done.
- If there is a mismatch, report failure.

TM for *BALANCE*

0	1	B

TM for *BALANCE*



TM for *BALANCE*

	0	1	B
q_{st}			Accept

TM for *BALANCE*

	0			1	B
q_{st}	B	R	q_{m0}		Accept

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}							

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}				

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}				Reject

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject
q_{ret}							

TM for *BALANCE*

	0			1			B
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	

TM for *BALANCE*

	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

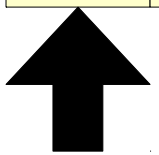
TM for *BALANCE*

0	0	1	1										

	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

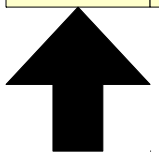
0	0	1	1										



	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

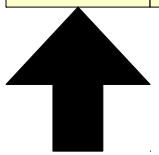
0	0	1	1										



	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

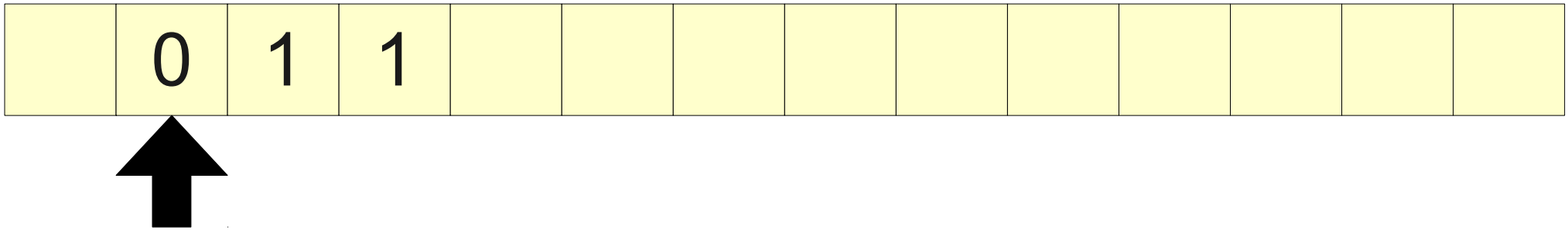
TM for *BALANCE*

0	0	1	1										



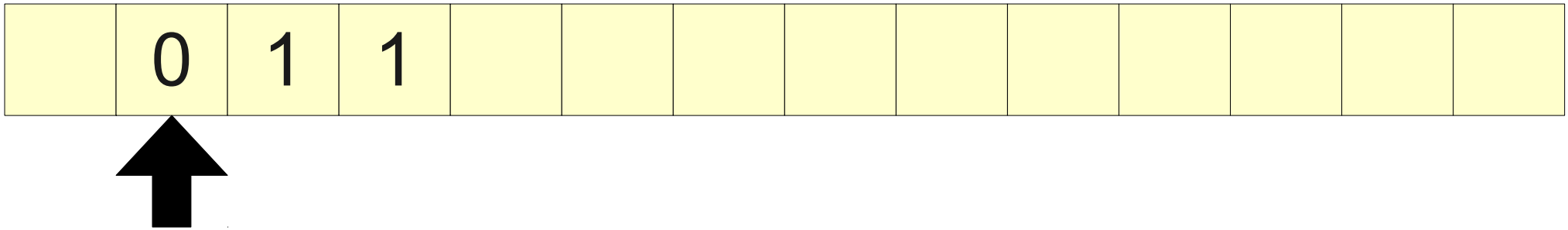
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



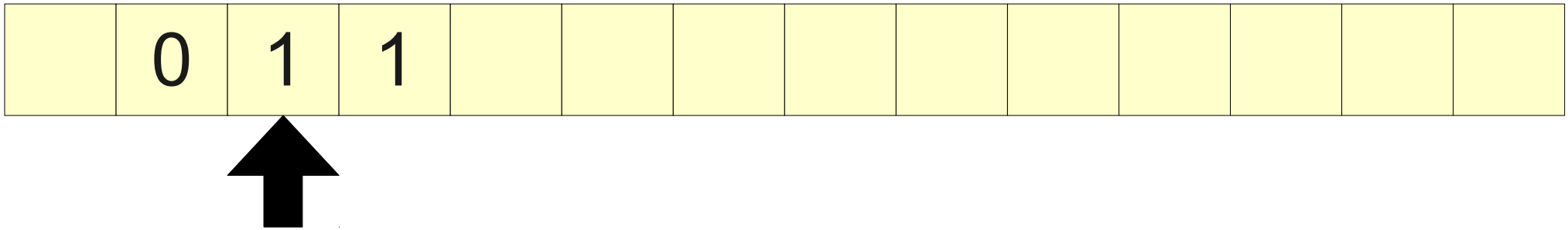
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



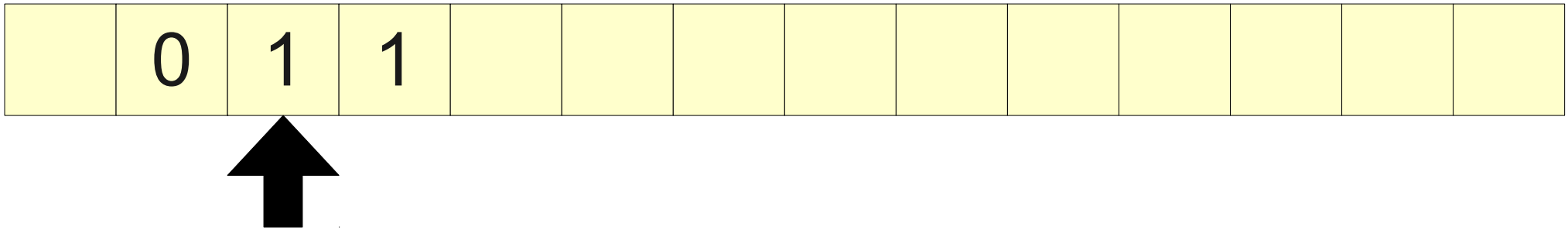
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



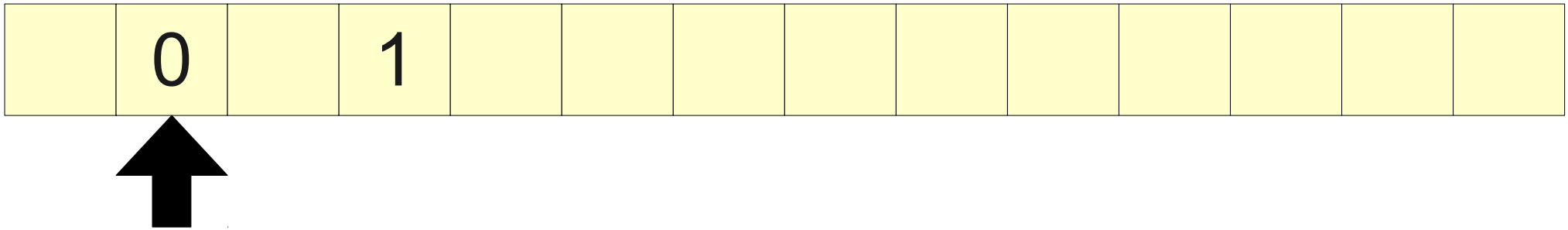
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



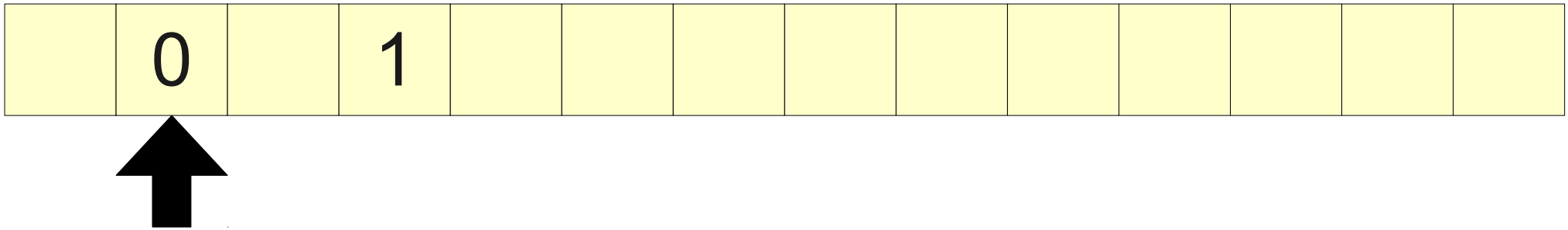
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



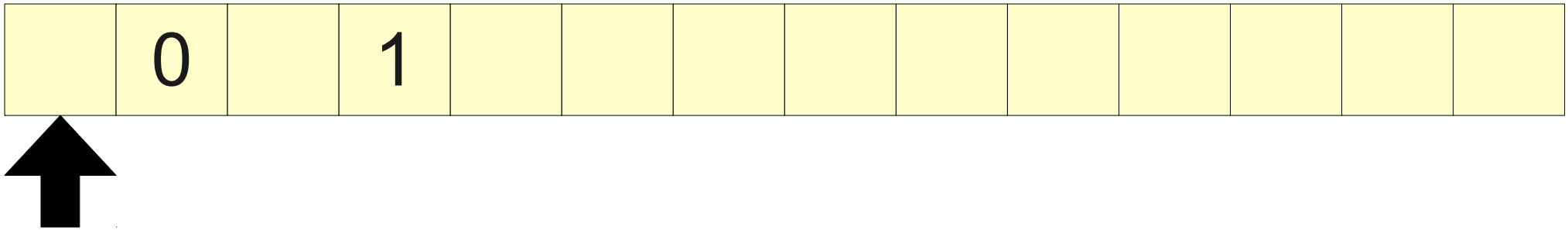
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



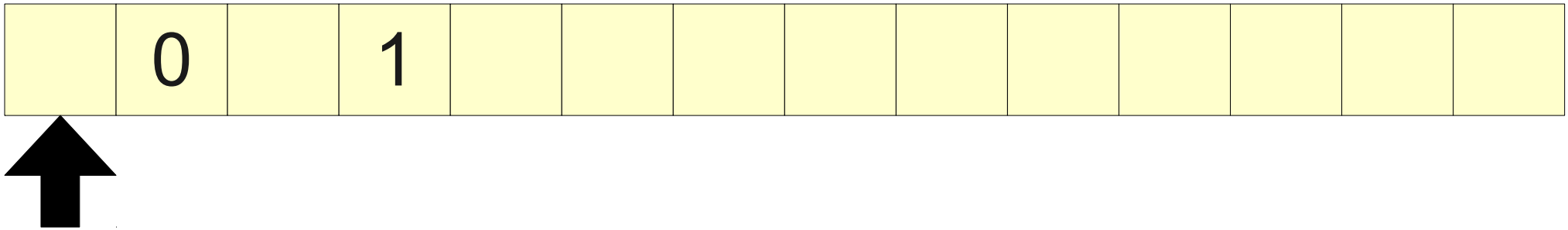
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



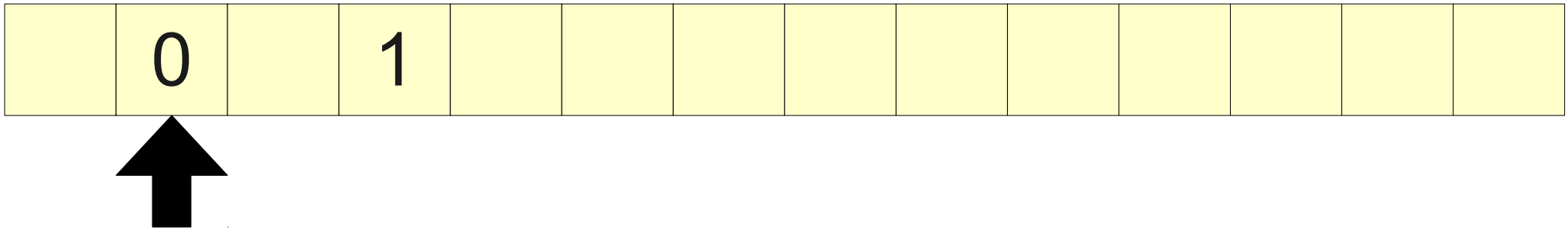
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



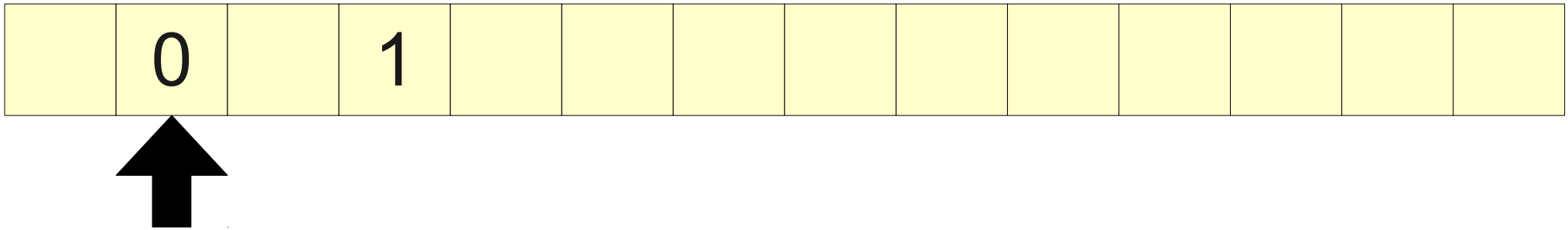
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



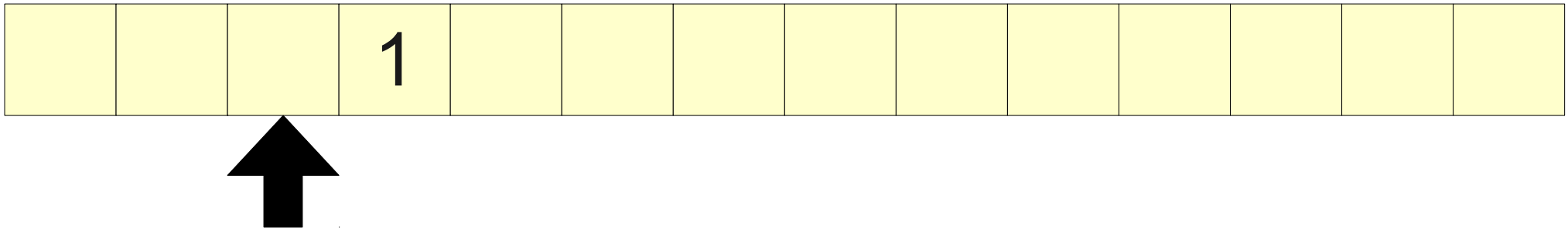
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



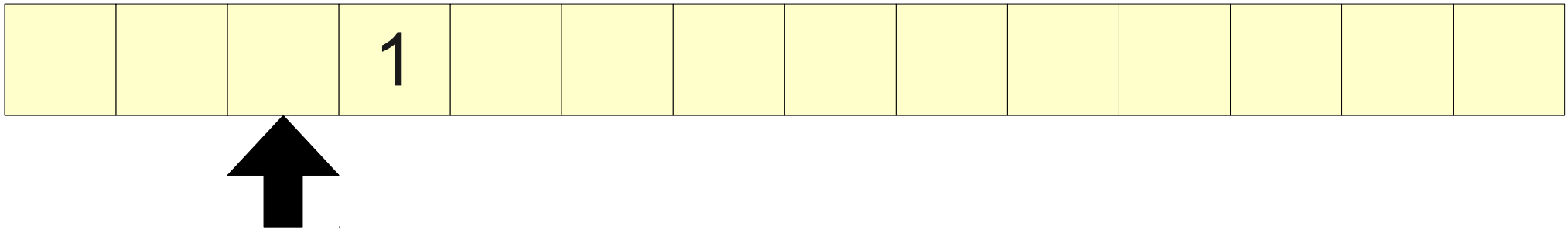
	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*



	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	B	L	q_{ret}	Reject		
q_{m1}	B	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

	0			1			B		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept		
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject		
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject		
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}

TM for *BALANCE*

	0			1			B			x
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	

TM for *BALANCE*

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject					
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject					
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}			

TM for *BALANCE*

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject					
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}			

TM for *BALANCE*

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}			

TM for *BALANCE*

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

0	0	1	1										

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

0	0	1	1									

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

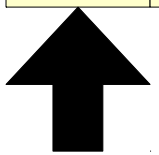
TM for *BALANCE*

0	0	1	1										

	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

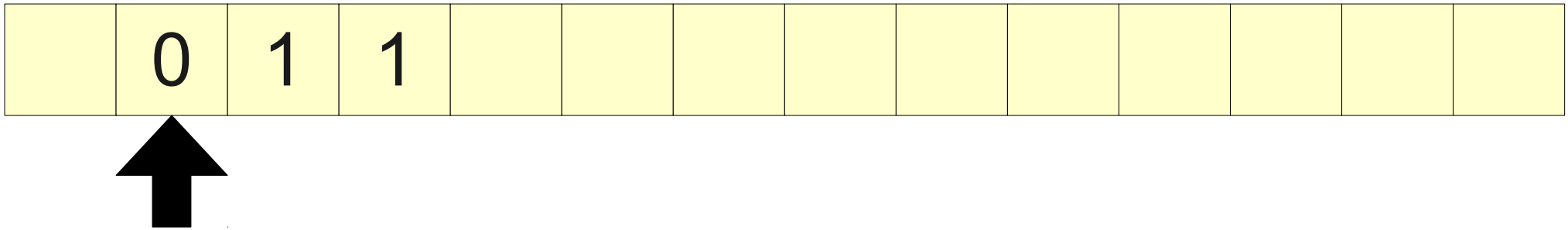
TM for *BALANCE*

0	0	1	1										



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

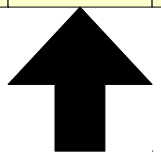
TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

	0	1	1									



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

	0	1	1									

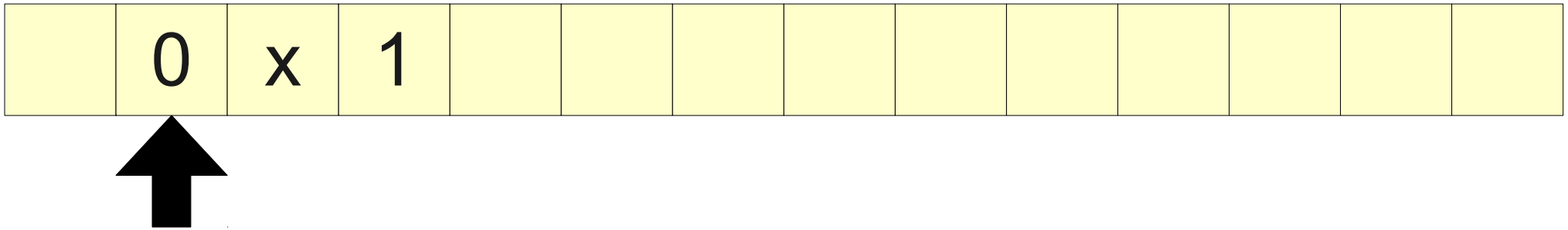
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

	0	1	1									

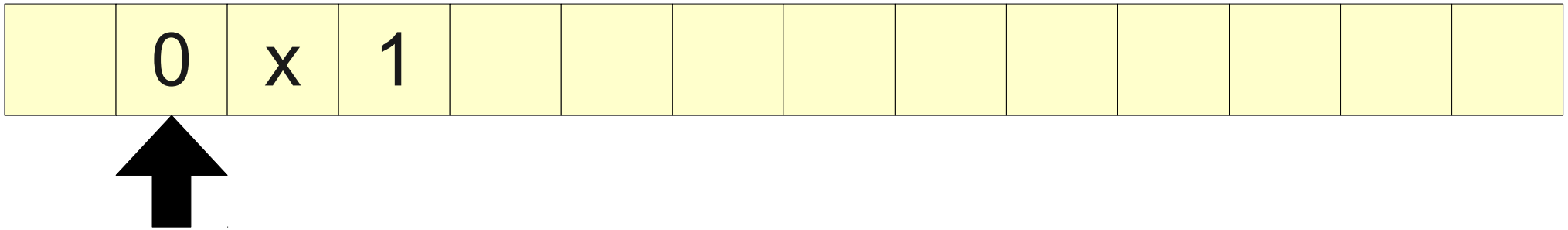
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

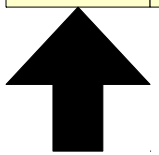
TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*

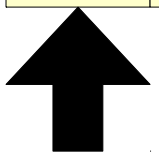
	0	x	1									



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

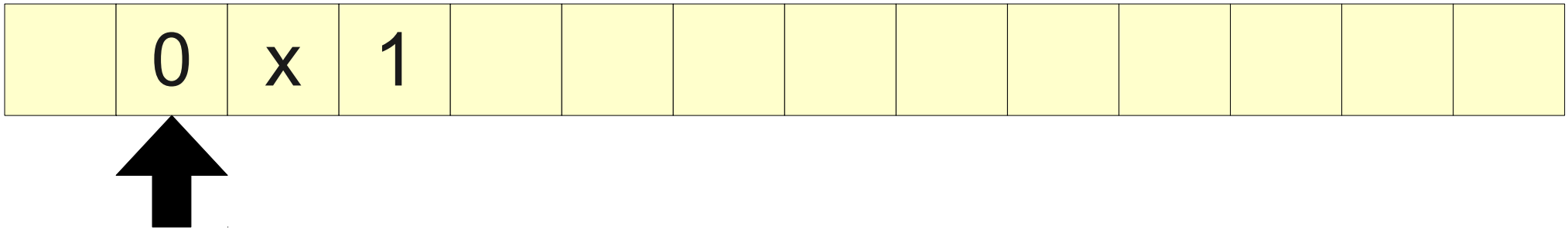
TM for *BALANCE*

	0	x	1									



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

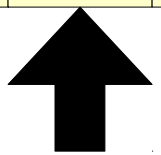
TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

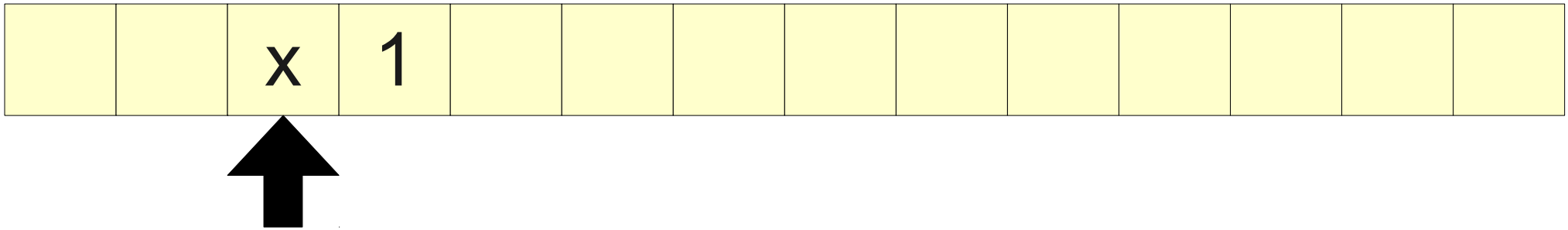
TM for *BALANCE*

	0	x	1									



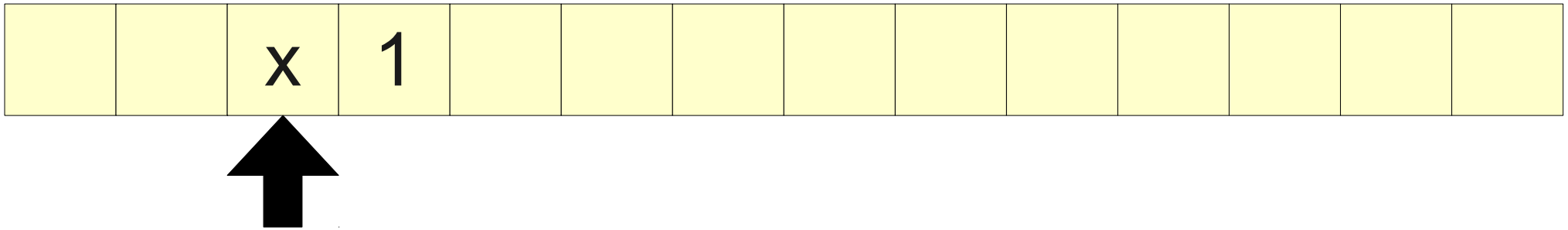
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



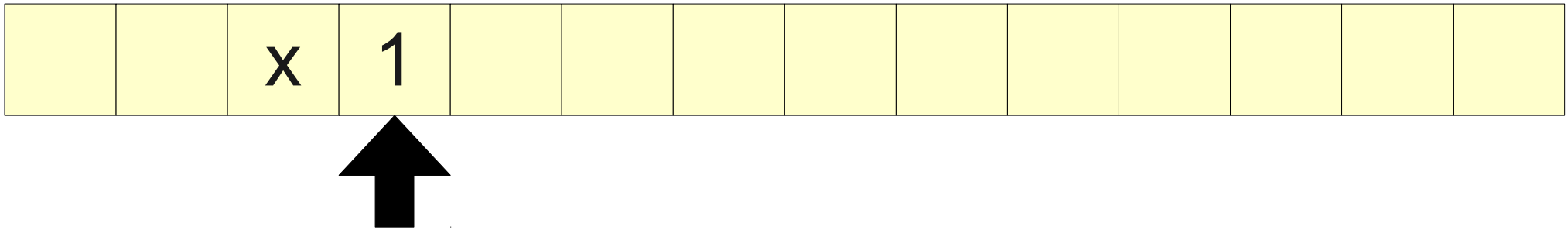
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



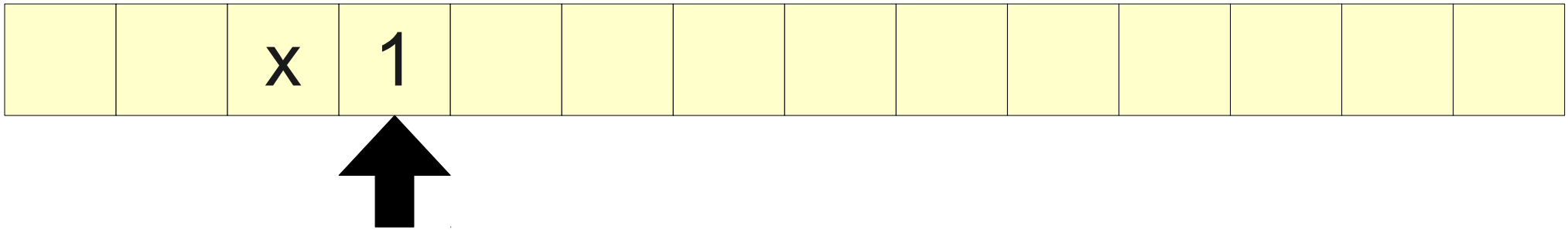
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



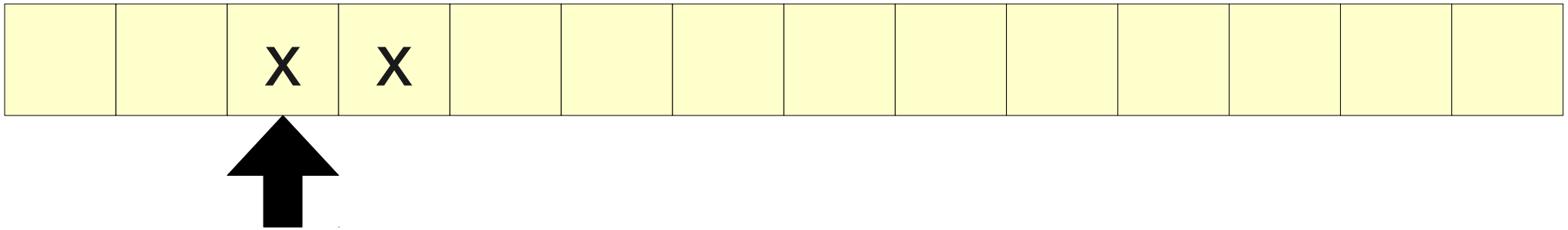
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



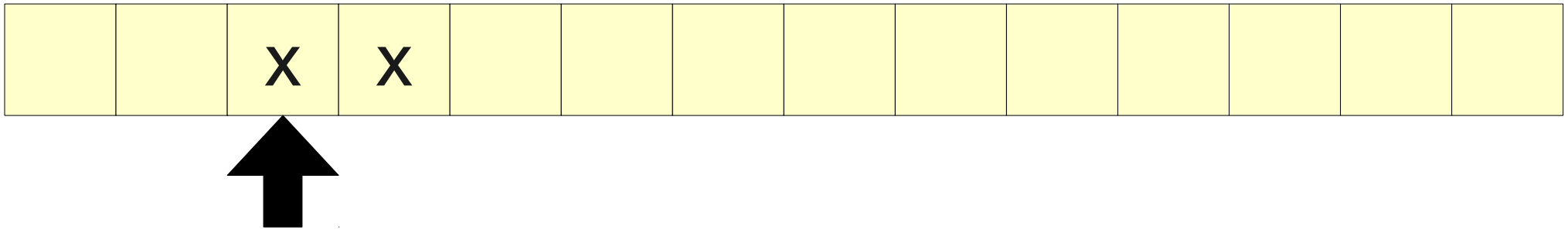
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



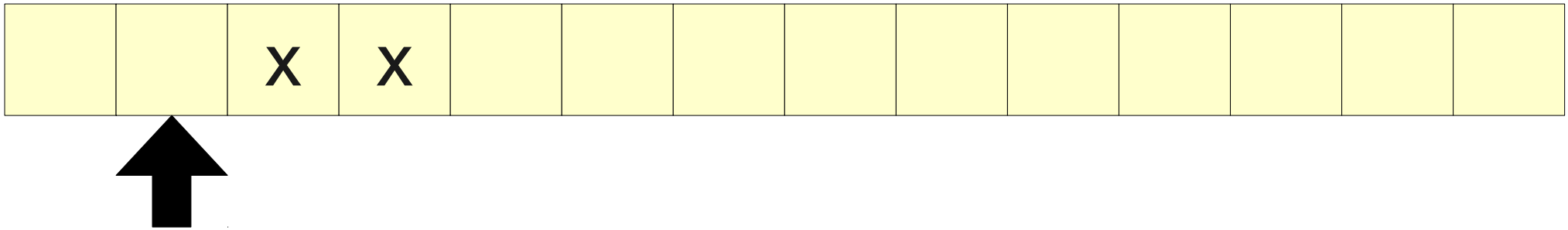
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



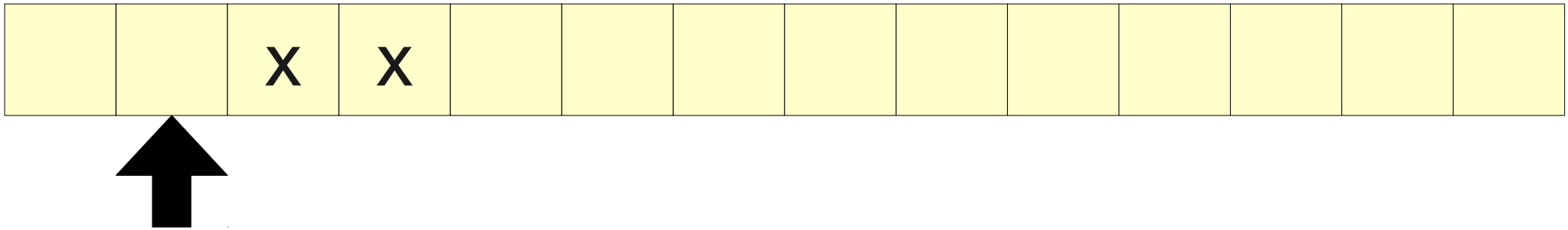
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



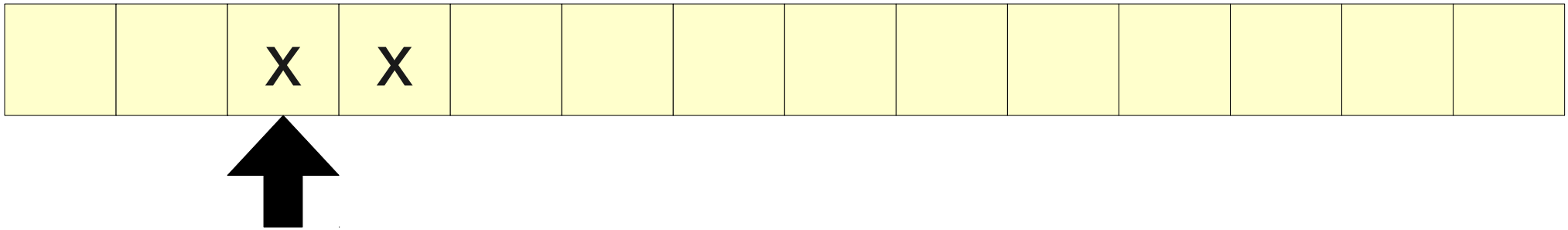
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



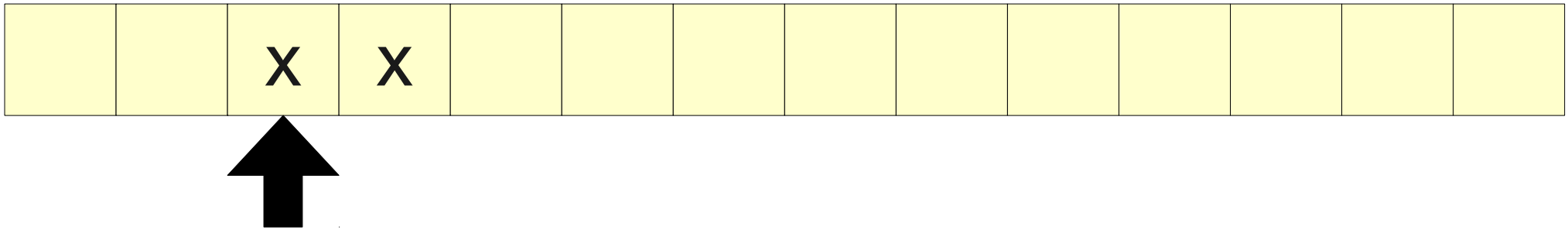
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



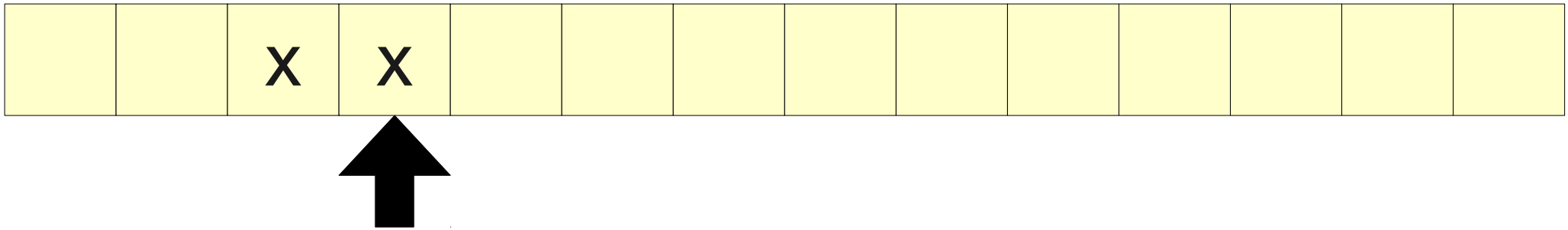
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



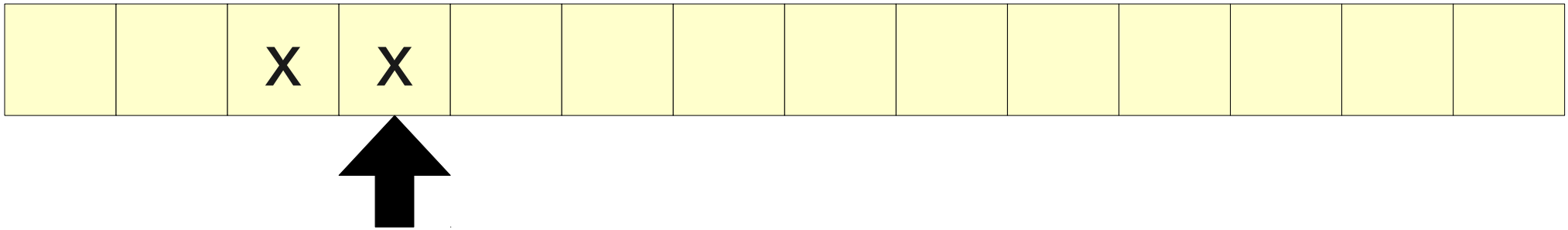
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



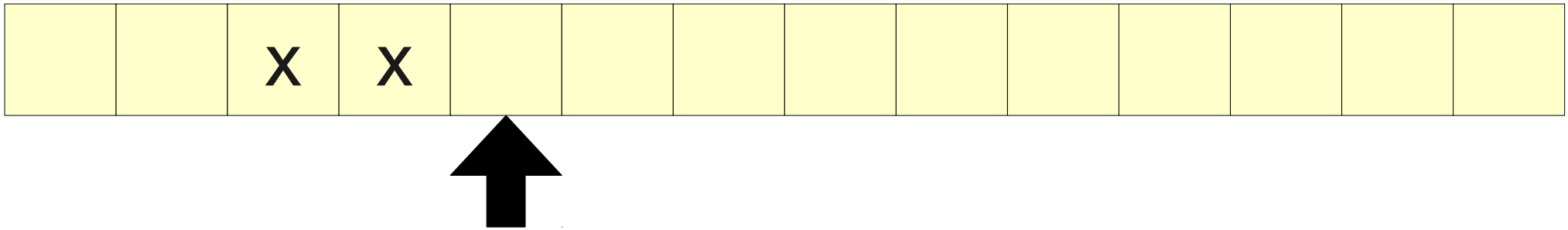
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



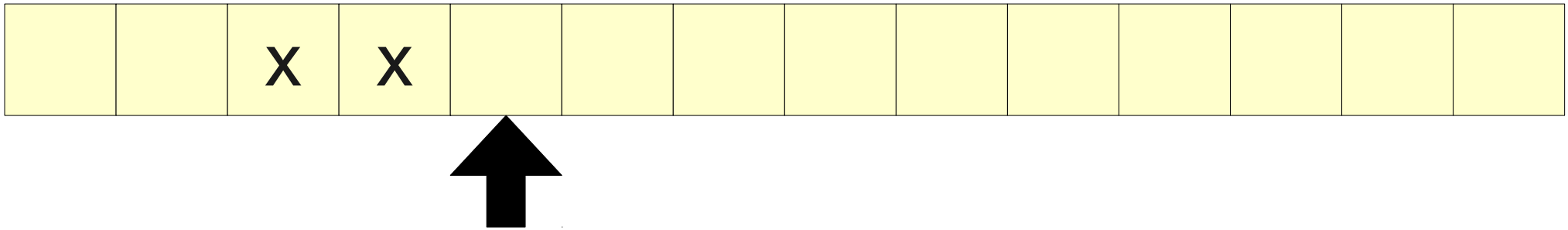
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



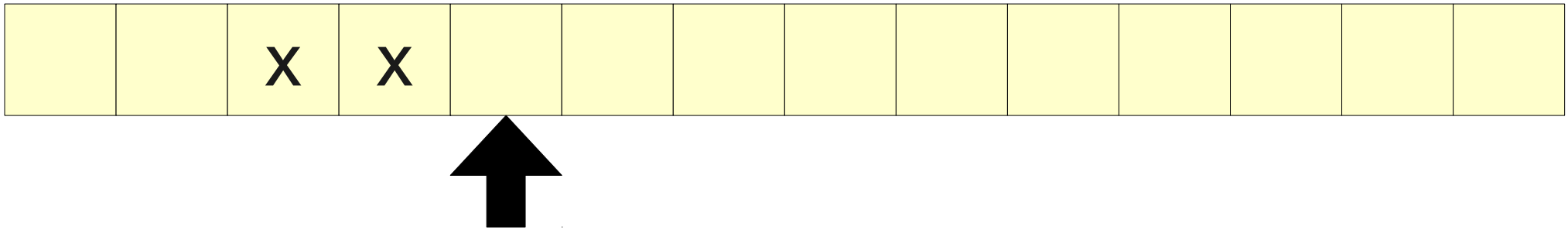
	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

TM for *BALANCE*



	0			1			B			x		
q_{st}	B	R	q_{m0}	B	R	q_{m1}	Accept			x	R	q_{st}
q_{m0}	0	R	q_{m0}	x	L	q_{ret}	Reject			x	R	q_{m0}
q_{m1}	x	L	q_{ret}	1	R	q_{m1}	Reject			x	R	q_{m1}
q_{ret}	0	L	q_{ret}	1	L	q_{ret}	B	R	q_{st}	x	L	q_{ret}

The Key Insight

- Our construction worked because we could make the finite-state control hold extra information (which symbol we had matched).
- ***General TM design trick***: Treat the finite state control as a combination control/finite memory.
- Can hold any finite amount of information by just replicating important states the appropriate number of times.

A More Elaborate Language

- Consider $\Sigma = \{ \mathtt{1}, \mathtt{\times}, \mathtt{=} \}$ and the language

$$\text{MULTIPLY} = \{ \mathtt{1}^n \mathtt{\times} \mathtt{1}^m \mathtt{=} \mathtt{1}^{mn} \mid m, n \in \mathbb{N} \}$$

- This language is neither regular nor context-free, but it is recursively enumerable.
- How would we build a TM for it?

A Turing Machine Subroutine

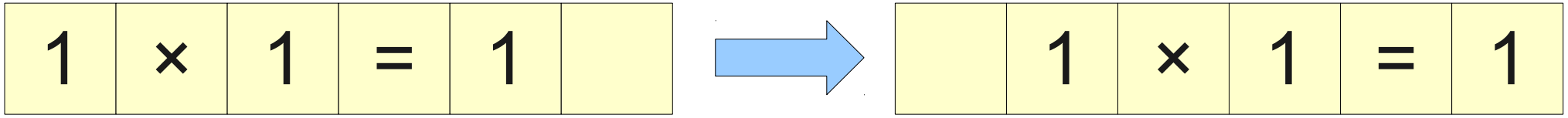
- A **subroutine** in a TM is state that, when entered:
  - Performs some specific task on the tape, then
  - Terminates in a well-specified state.
- Complex Turing machines can be broken down into smaller subroutines as follows:
  - The start state fires off the first subroutine.
  - After the first subroutine terminates, the next begins.
  - (etc.)
  - The machine may accept or reject at any point.

Key Idea: Subroutines

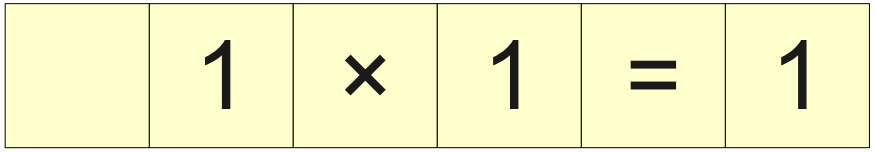
- Checking whether a string is in *MULTIPLY* requires several different steps:
  - Check that the string is formatted correctly.
  - Compute $m \times n$.
  - Confirm that $m \times n$ matches what's given.
- Let's design a subroutine for each of these.

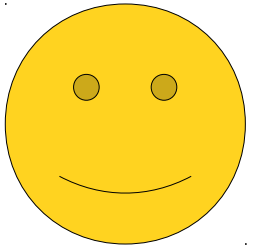
Validating the Input

- High-level idea:
  - Shift the input over by one step.

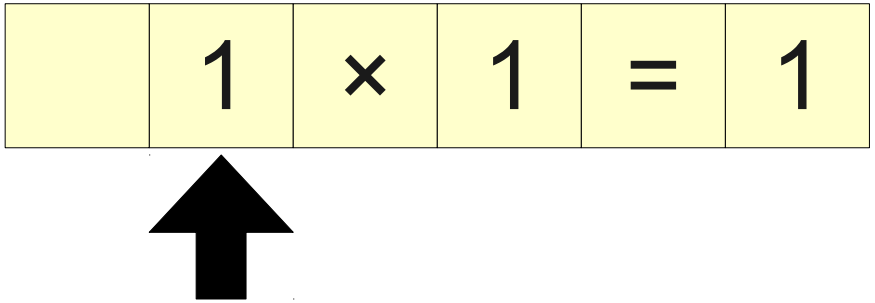


  - Check the structure of the input.





  - End up in a new state looking at the first character of the input if successful.



Step One: Shift the Input

Step One: Shift the Input

	1	×	=	B
q_s	?	?	?	?

Step One: Shift the Input

	1			×	=	B
q_s	B	R	q_1	?	?	?

Step One: Shift the Input

	1			×			=	B
q_s	B	R	q_1	B	R	$q_\times$	?	?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	?			?			?			?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	?			?			?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	?			?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	?
$q_\times$	?			?			?			?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	?
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	?
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	?
$q_=$	?			?			?			?

Step One: Shift the Input

	1			×			=			B
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	?
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	?
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	?

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	?			?			?			?		

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	?			?			?		

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	?			?		

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	?		

Step One: Shift the Input

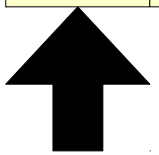
	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	?		
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

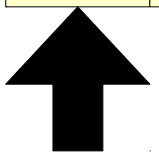
1	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

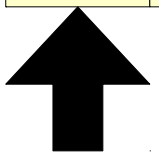
1	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

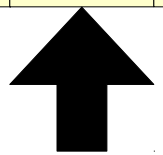
1	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

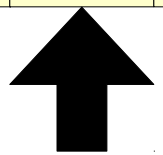
	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

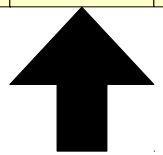
	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

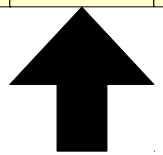
	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

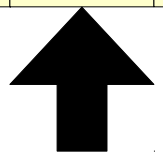
	1	×	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

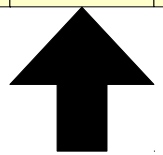
	1	1	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	1	1	=	1	1	1	1			



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	=	1	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	=	1	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	=	1	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

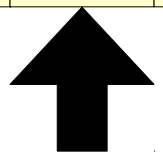
Step One: Shift the Input

	1	1	×	1	=	1	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

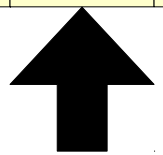
	1	1	×	1	1	1	1	1	1				



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	1	1	1	1				



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1			

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

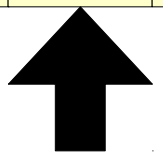
Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

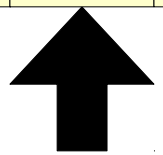
Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

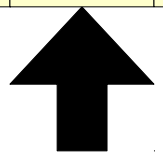
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

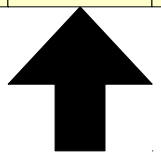
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

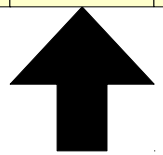
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

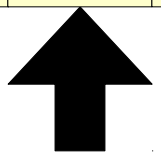
Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

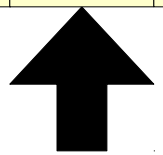
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

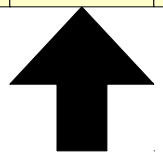
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

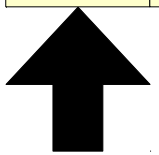
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

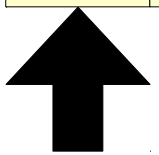
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

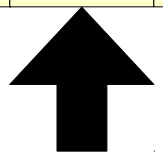
	1	1	×	1	1	=	1	1	1	1		



	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step One: Shift the Input

	1	1	×	1	1	=	1	1	1	1		



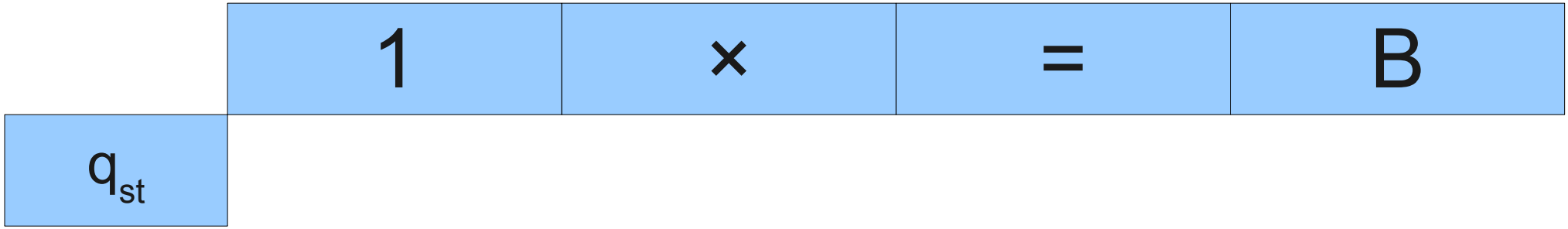
	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	**D**
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	**D**

Step Two: Verify the Input

Step Two: Verify the Input

1	×	=	B

Step Two: Verify the Input



Step Two: Verify the Input

	1	×	=	B
q_{st}	1 R q_{st}			

Step Two: Verify the Input

	1			×	=	B
q_{st}	1	R	q_{st}		Reject	

Step Two: Verify the Input

	1			×	=	B
q_{st}	1	R	q_{st}		Reject	Reject

Step Two: Verify the Input

	1			×			=	B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject	Reject

Step Two: Verify the Input

	1			×			=	B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject	Reject
$q_{\times}$								

Step Two: Verify the Input

	1			×			=	B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject	Reject
$q_{\times}$	1	R	$q_{\times}$					

Step Two: Verify the Input

	1			×			=	B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject	Reject
$q_{\times}$	1	R	$q_{\times}$	Reject				

Step Two: Verify the Input

	1			×			=	B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject	Reject
$q_{\times}$	1	R	$q_{\times}$	Reject				Reject

Step Two: Verify the Input

	1			×			=			B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject

Step Two: Verify the Input

	1			×			=			B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject
$q_{=}$										

Step Two: Verify the Input

	1			×			=			B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject
$q_{=}$	1	R	$q_{=}$							

Step Two: Verify the Input

	1			×			=			B
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject
$q_{=}$	1	R	$q_{=}$	Reject			Reject			

Step Two: Verify the Input

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L

Step Two: Verify the Input

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L			

Step Two: Verify the Input

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

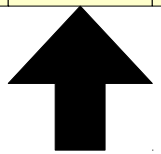
Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

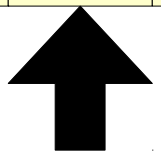
Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

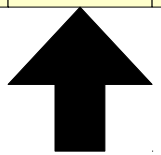
Step Two: Verify the Input

	1	1	×	1	=	1	1					

	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

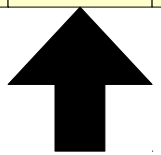
	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

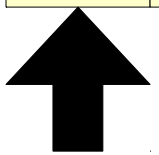
	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

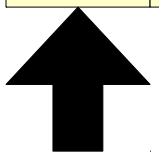
	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

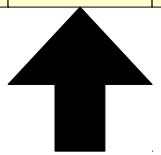
	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Step Two: Verify the Input

	1	1	×	1	=	1	1					



	1			×			=			B		
q_{st}	1	R	q_{st}	×	R	$q_{\times}$	Reject			Reject		
$q_{\times}$	1	R	$q_{\times}$	Reject			=	R	$q_{=}$	Reject		
$q_{=}$	1	R	$q_{=}$	Reject			Reject			B	L	q_L
q_L	1	L	q_L	×	L	q_L	=	L	q_L	B	R	**D**

Putting it Together: Shift/Verify

	1			×			=			B		
q_s	B	R	q_1	B	R	$q_\times$	B	R	$q_=$	B	R	q_{s2}
q_1	1	R	q_1	1	R	$q_\times$	1	R	$q_=$	1	L	q_R
$q_\times$	×	R	q_1	×	R	$q_\times$	×	R	$q_=$	×	L	q_R
$q_=$	=	R	q_1	=	R	$q_\times$	=	R	$q_=$	=	L	q_R
q_R	1	L	q_R	×	L	q_R	=	L	q_R	B	R	q_{s2}
q_{s2}	1	R	q_{s2}	×	R	$q_{\times 2}$	Reject			Reject		
$q_{\times 2}$	1	R	$q_{\times 2}$	Reject			=	R	$q_{=2}$	Reject		
$q_{=2}$	1	R	$q_{=2}$	Reject			Reject			B	L	q_{L2}
q_{L2}	1	L	q_{L2}	×	L	q_{L2}	=	L	q_{L2}	B	R	**D**

Step Three: Doing the Multiply

	1	1	×	1	1	1	=	1							

		1	×	1	1	1	=	1		1	1	1			

			×	1	1	1	=	1		1	1	1	1	1	1

Step Four: Checking the Multiply

			×	1	1	1	=	1	1	1		1	1	1	
			×	1	1	1	=		1	1		1	1		
			×	1	1	1	=			1		1			
			×	1	1	1	=								

Why This Matters

- TMs can solve a large class of problems, but they can be enormously complicated.
- We now have two tricks for designing TMs:
  - Constant storage
  - Subroutines
- We can use these tricks to show that if we can get each individual piece working, we can solve a large problem with a TM.

Next Time

- **Programming Turing Machines**
  - A cleaner way to think about TMs.
- **The Power of Turing Machines**
  - Just how much expressive power do TMs have?