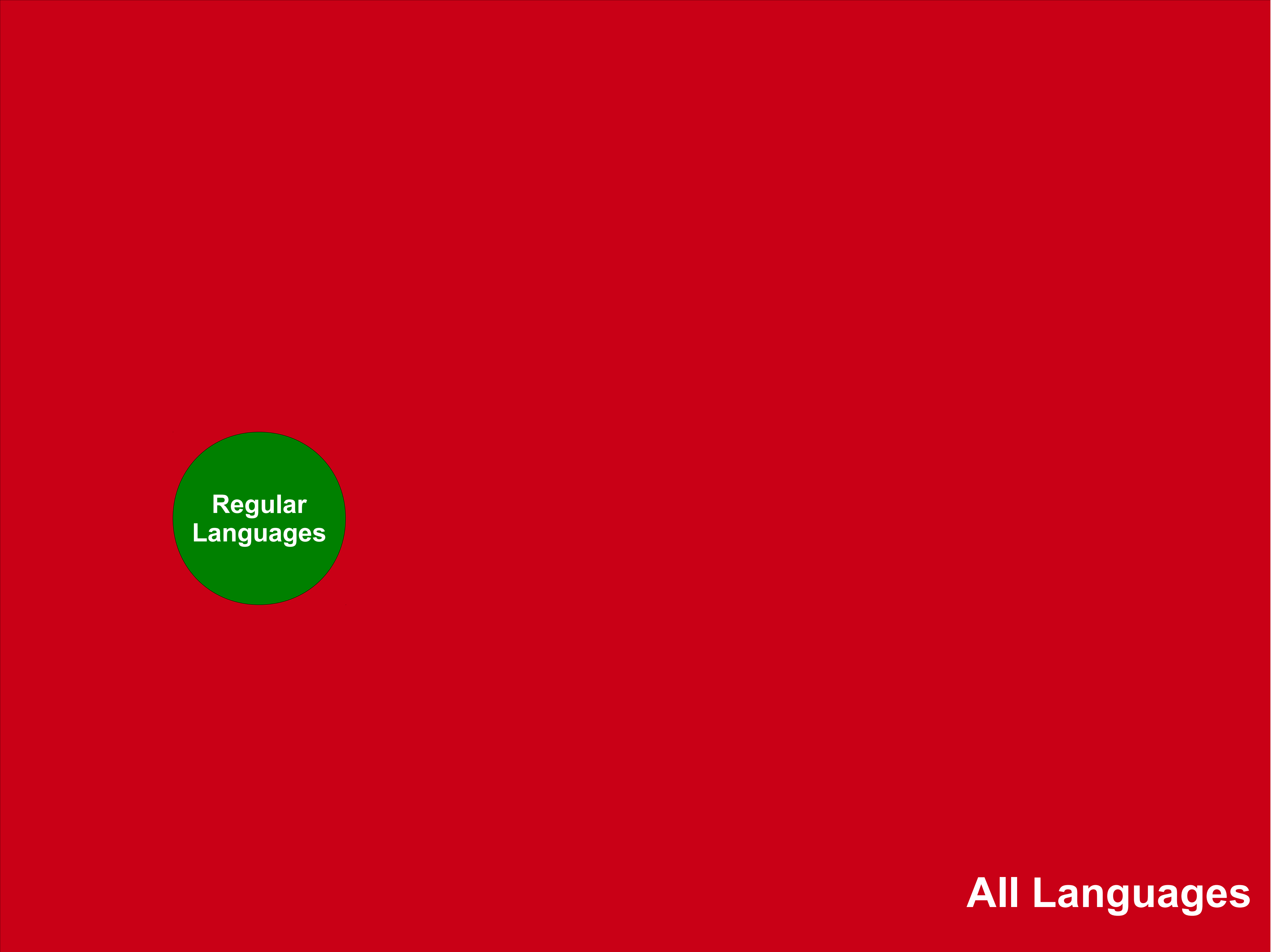
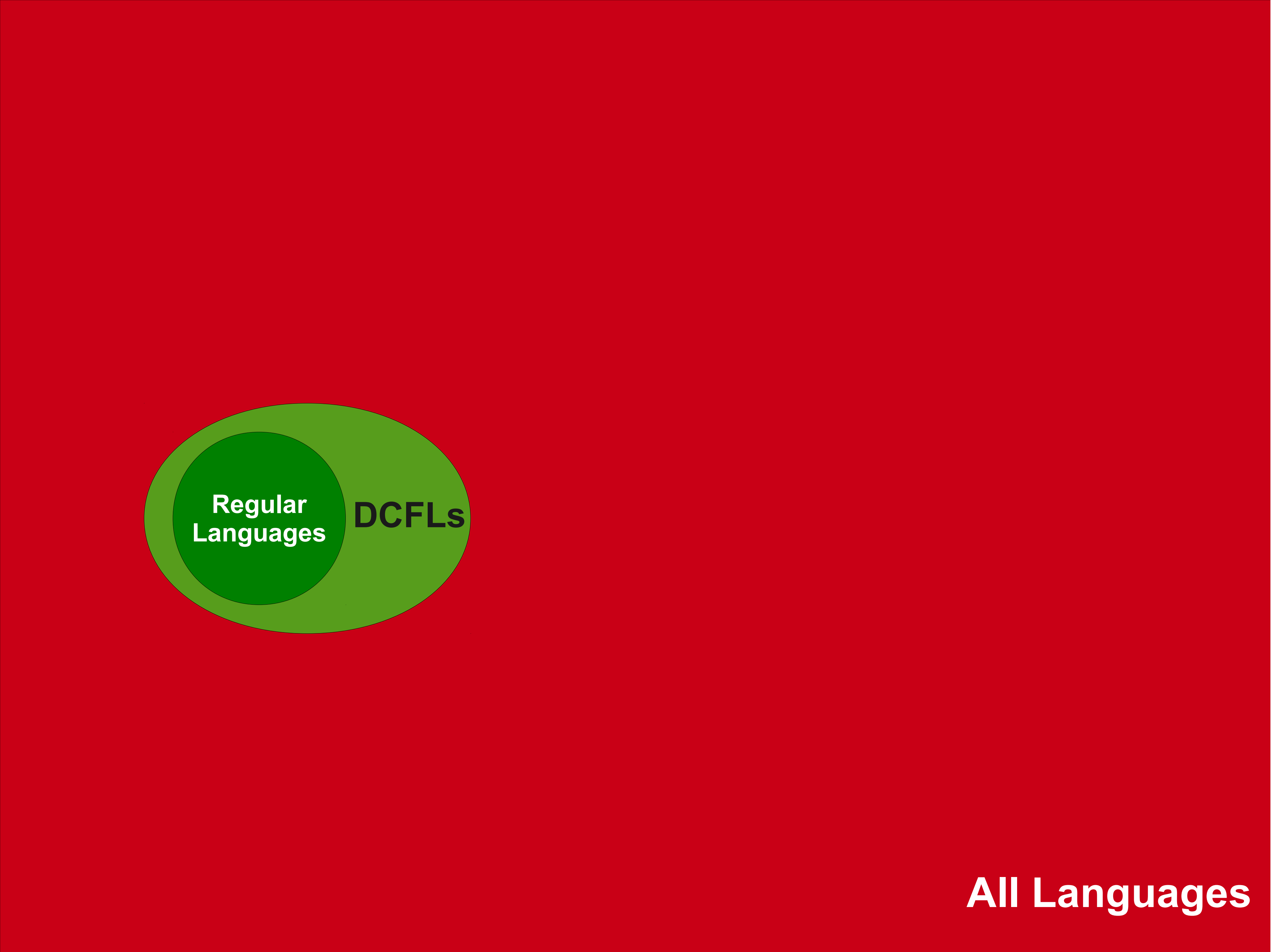
The **Limits** of Context-Free Languages

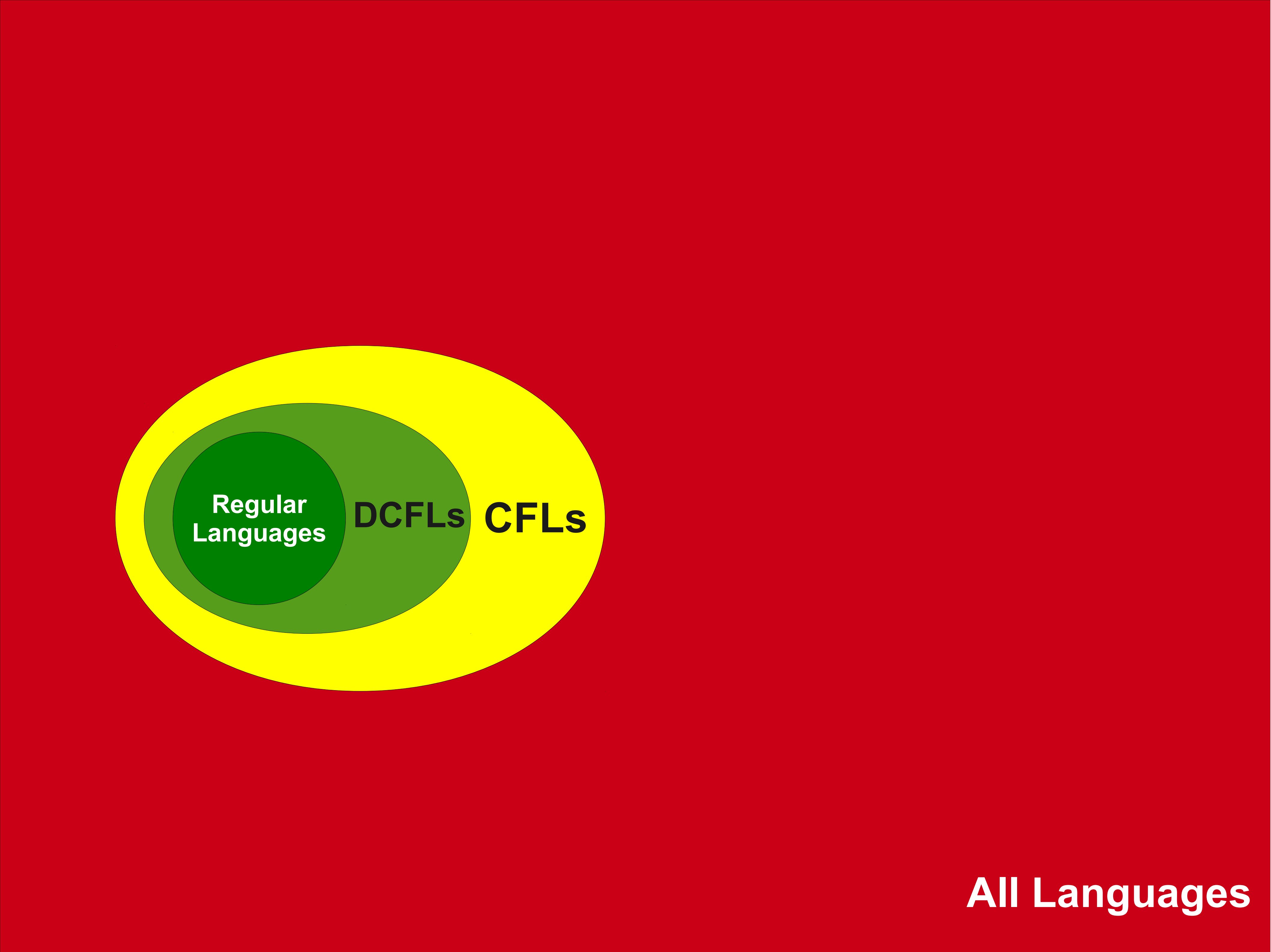
Announcements

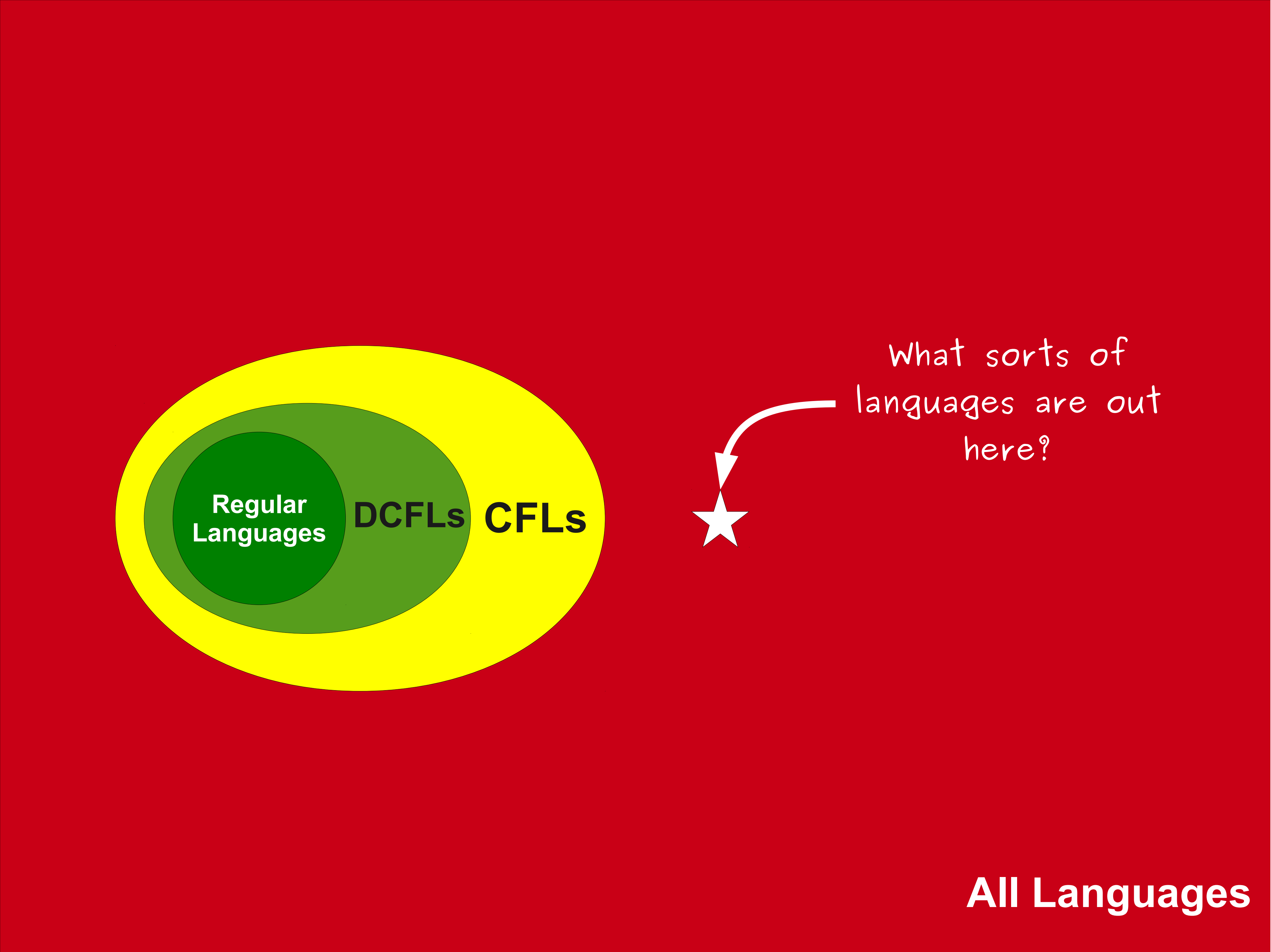
- Problem Set 6 due right now.
- Problem Set 7 out, due next Friday, November 18.
  - Covers context-free languages, CFGs, PDAs, and the limits of CFLs (from today's lecture)
  - Ask questions in office hours!
  - Email questions to **cs103@cs.stanford.edu**
- Problem Set 4 graded, will be returned at end of class.

All Languages


Regular
Languages
All Languages


Regular Languages
DCFLs
All Languages


Regular Languages
DCFLs
CFLs
All Languages


Regular Languages
DCFLs
CFLs
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

$R \rightarrow a \mid b \mid c \mid \ldots$

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

Recall: Parse Trees

R → a | b | c | …

R → “ε”

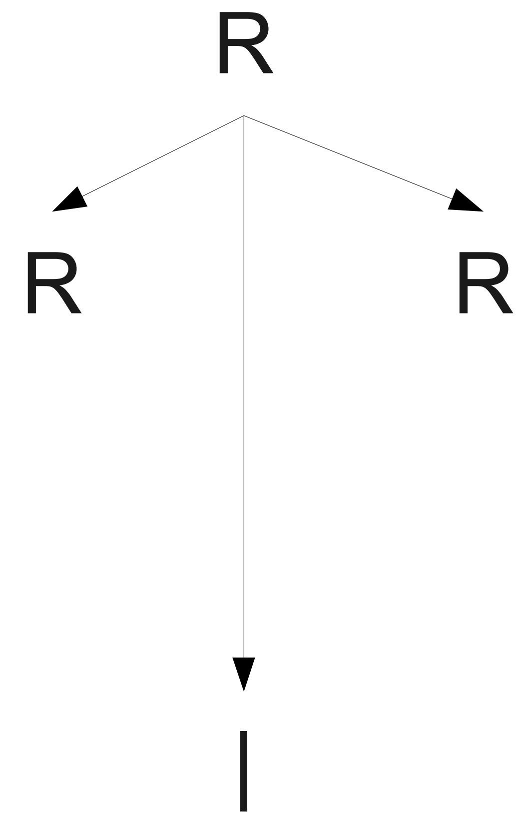
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → “ε”

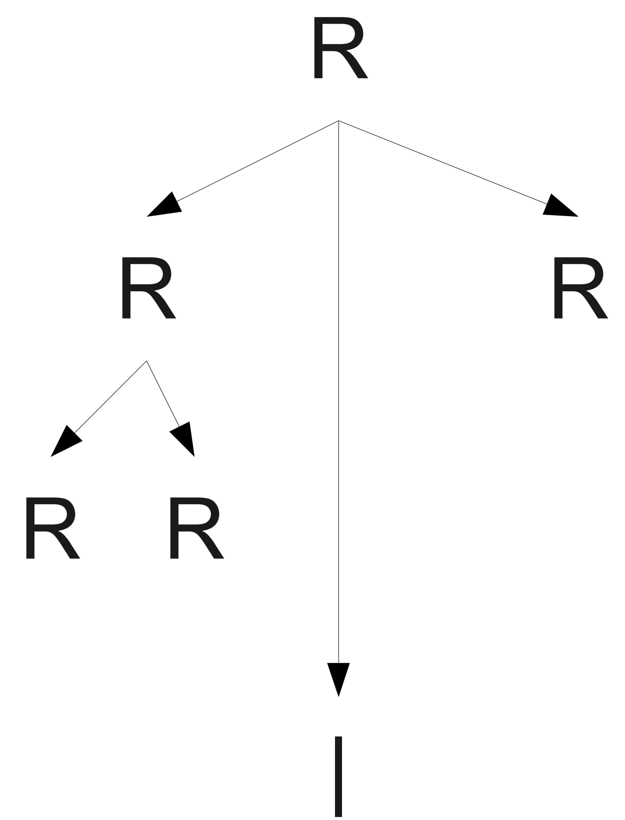
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → "ε"

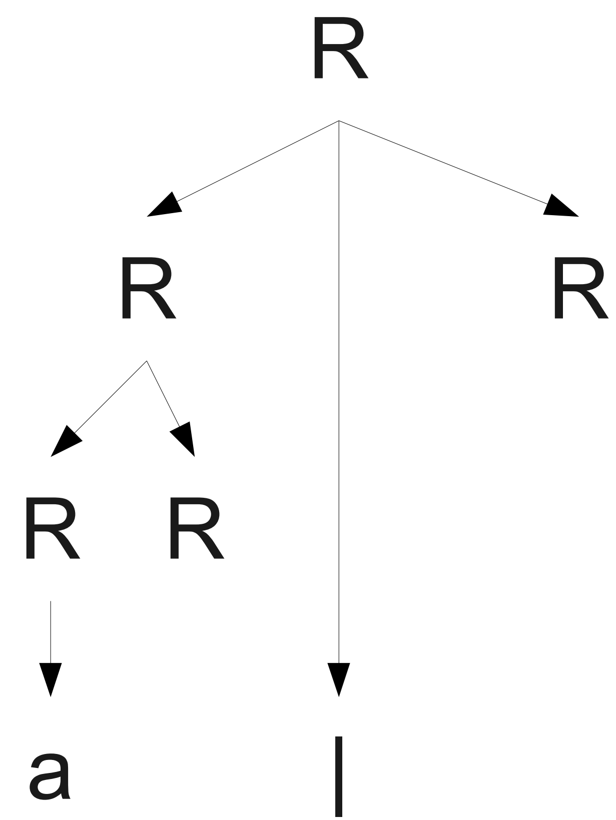
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → "ε"

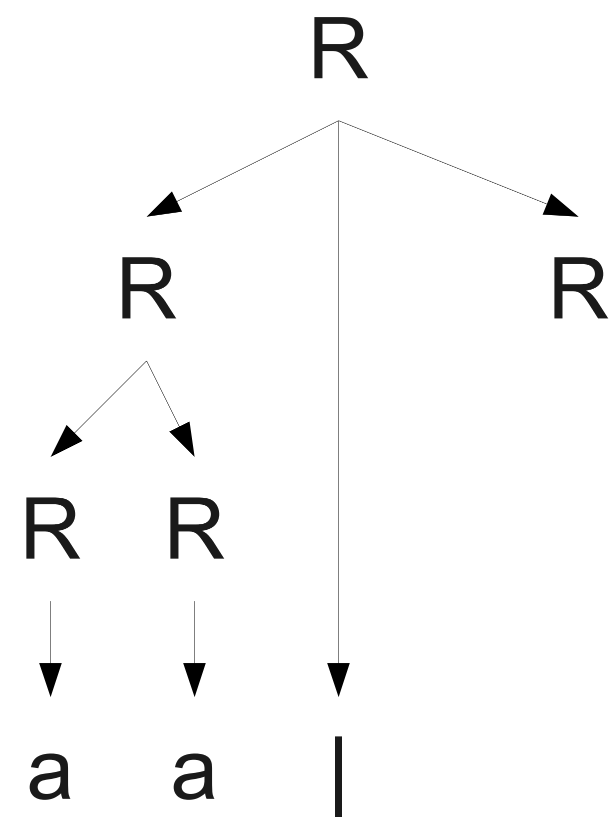
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → "ε"

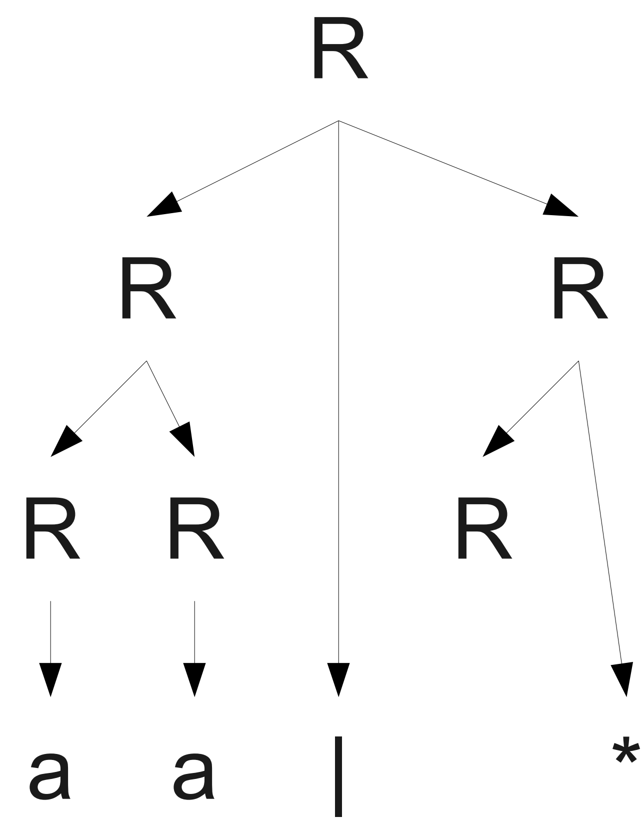
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → “ε”

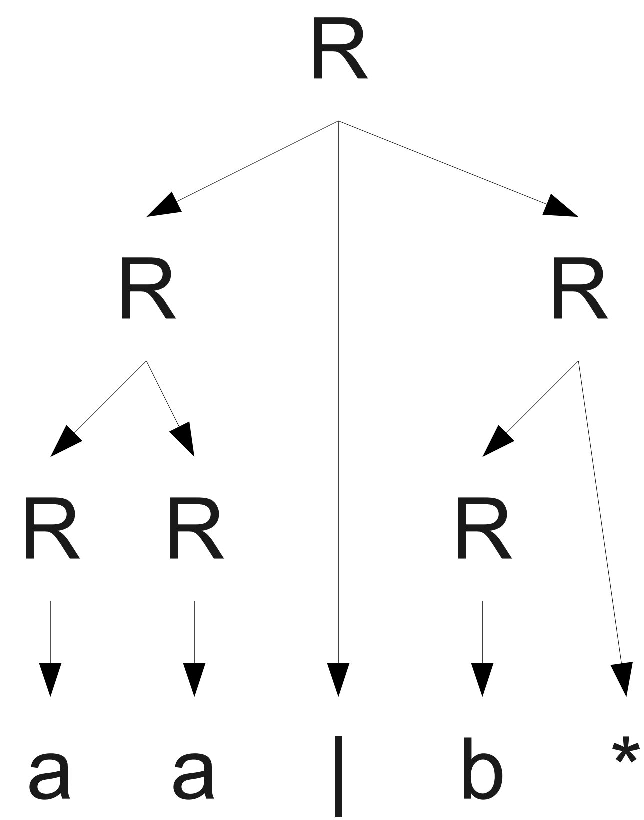
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Recall: Parse Trees

R → a | b | c | …

R → “ε”

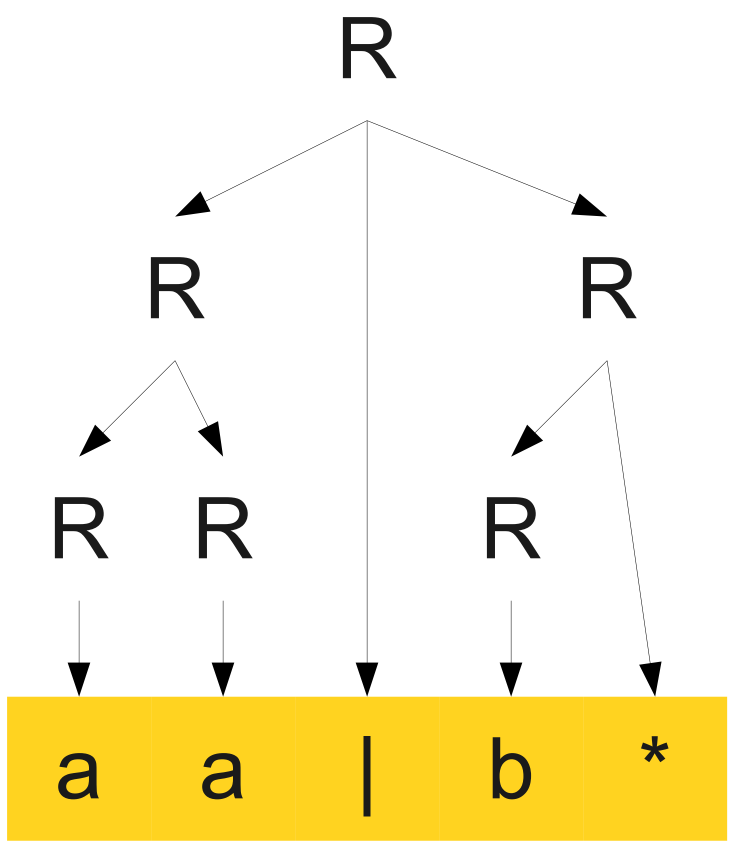
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



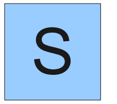
Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

Parse Trees Revisited



S → [P]

P → RR | a

R → (P) | b

Parse Trees Revisited

S → [P]

P → RR | a

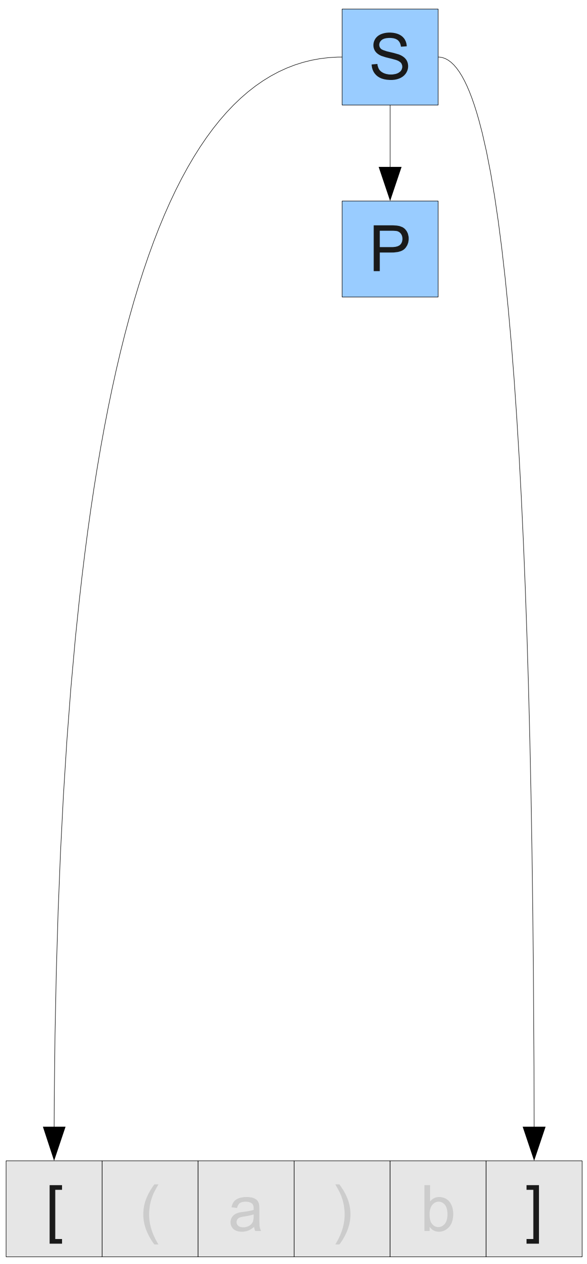
R → (P) | b

Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

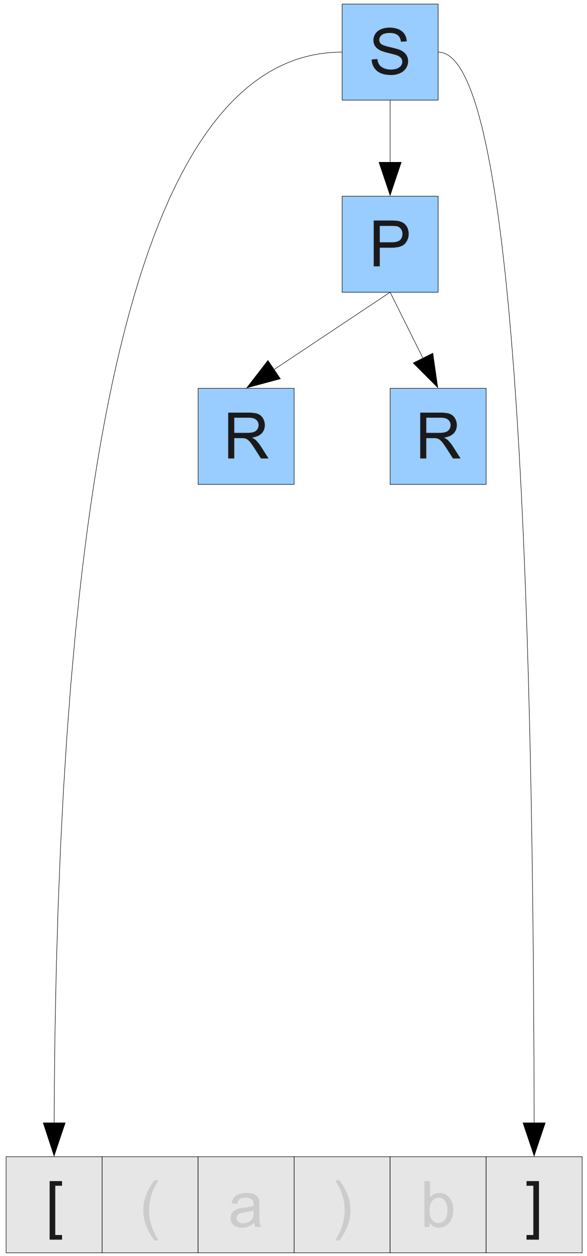


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

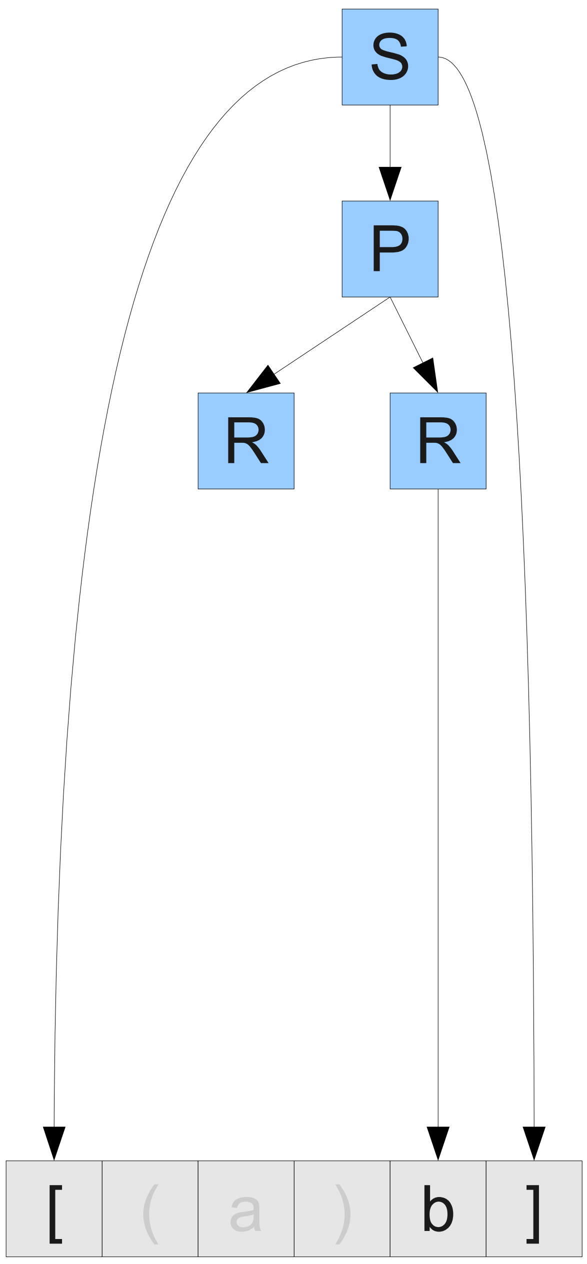


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

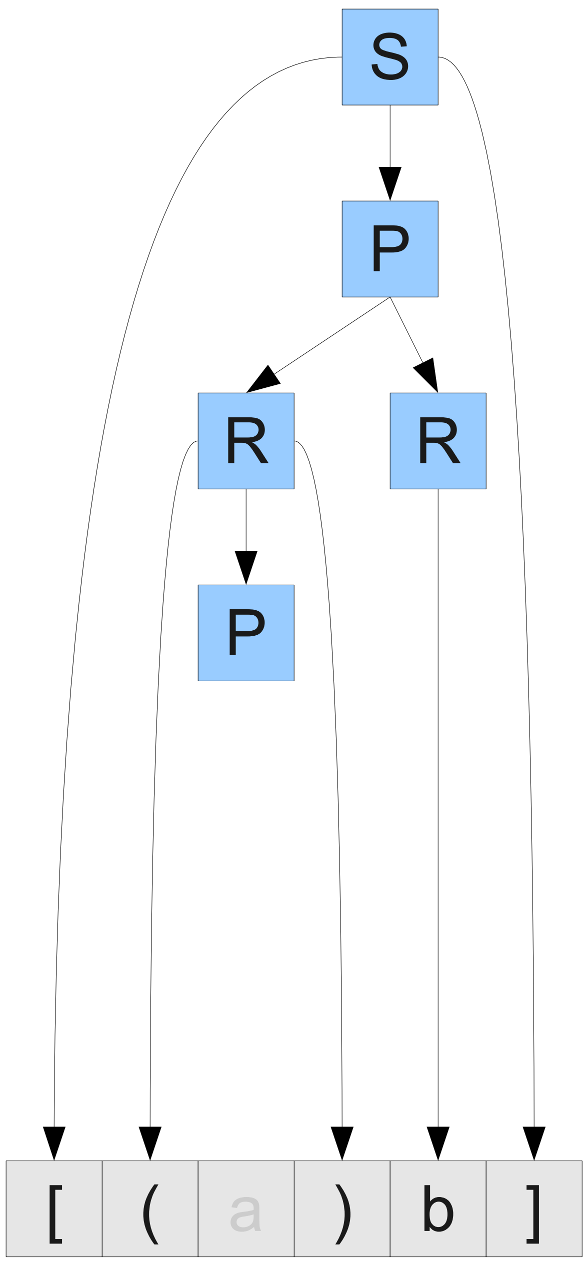


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

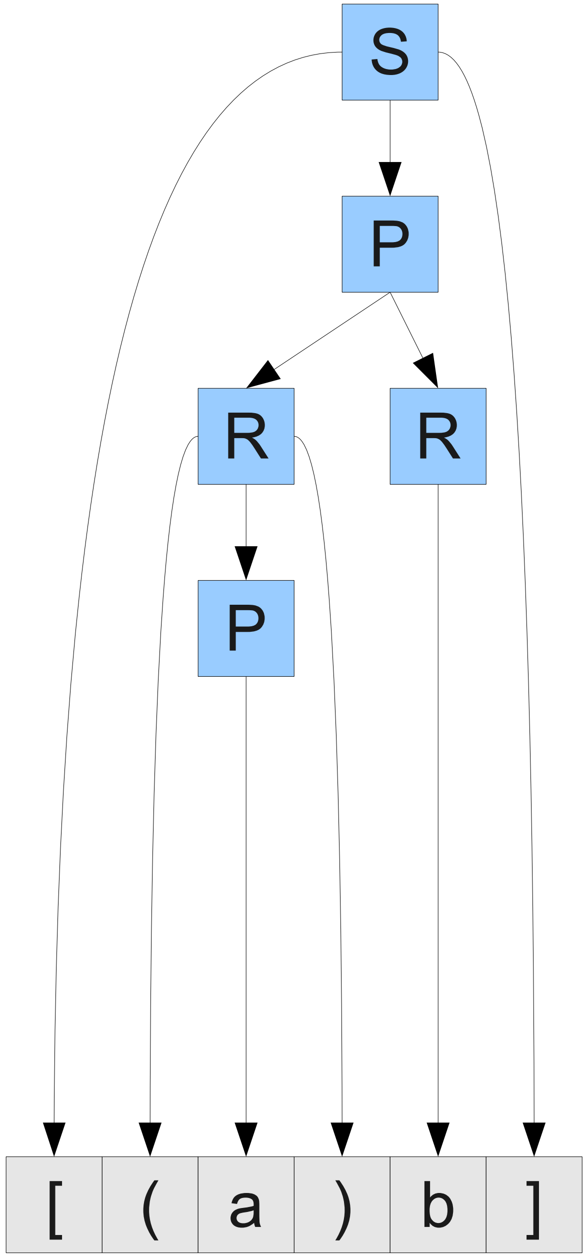


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

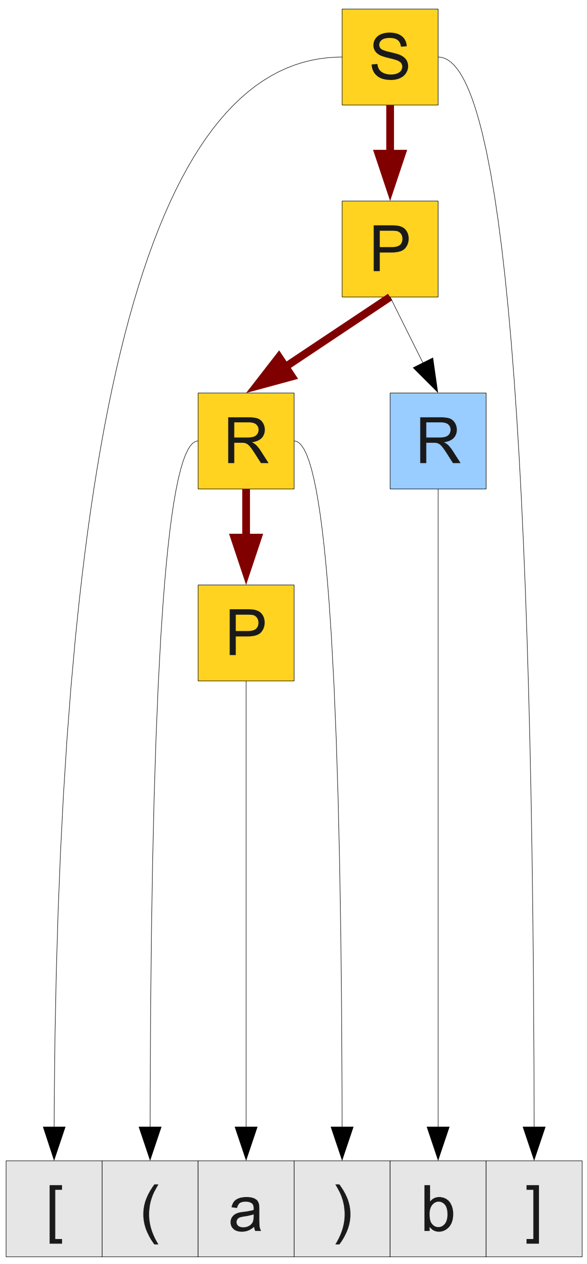


Parse Trees Revisited

$S \to [P]$

$P \to RR \mid a$

$R \to (P) \mid b$

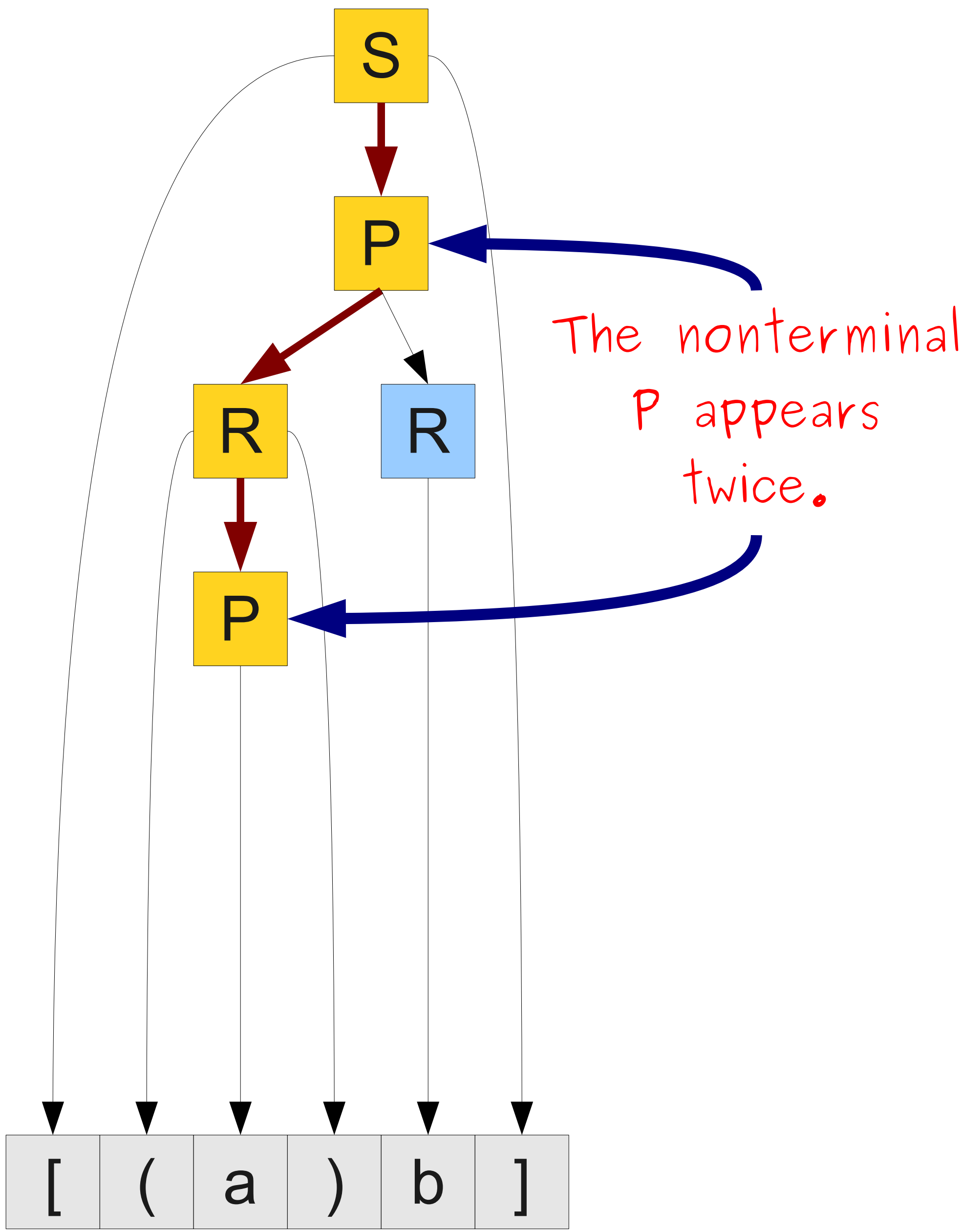


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

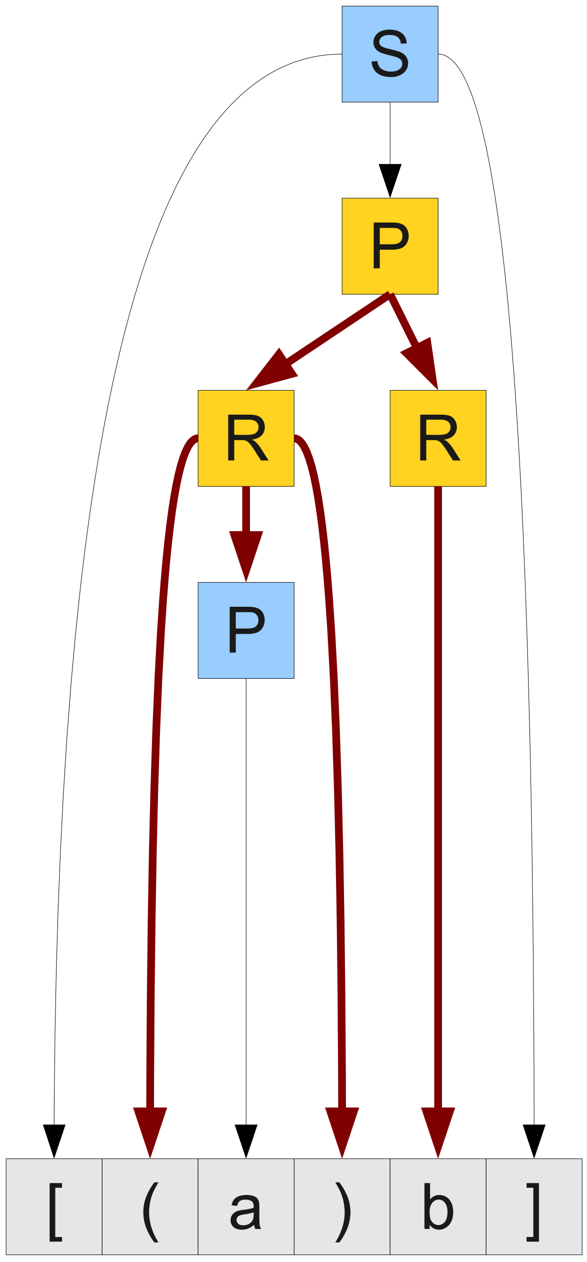


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

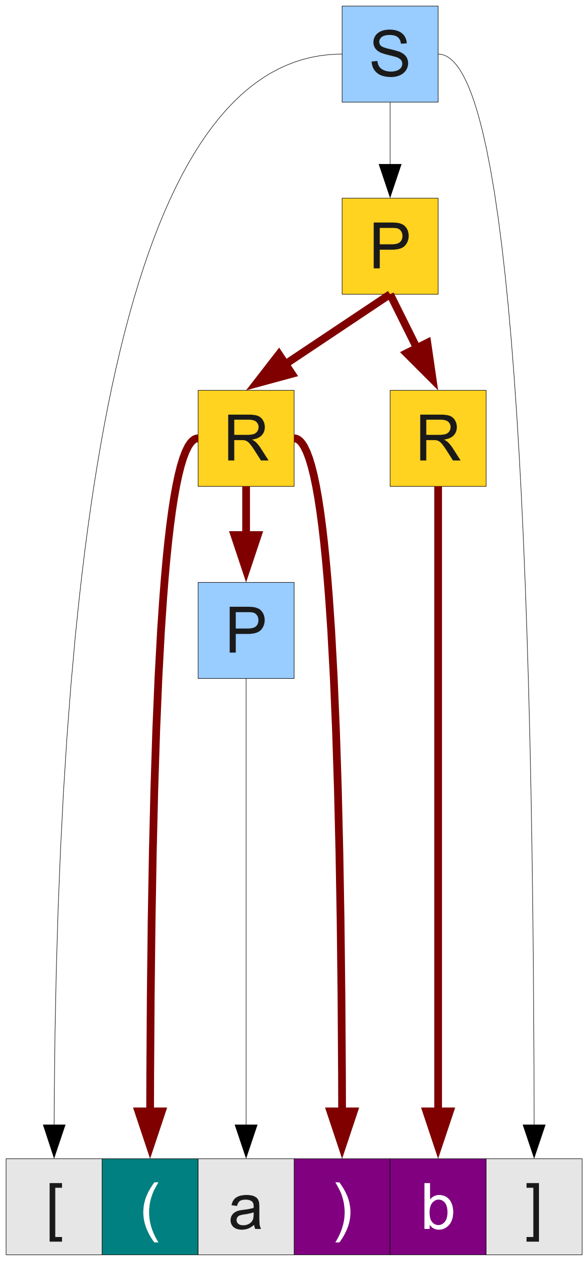


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

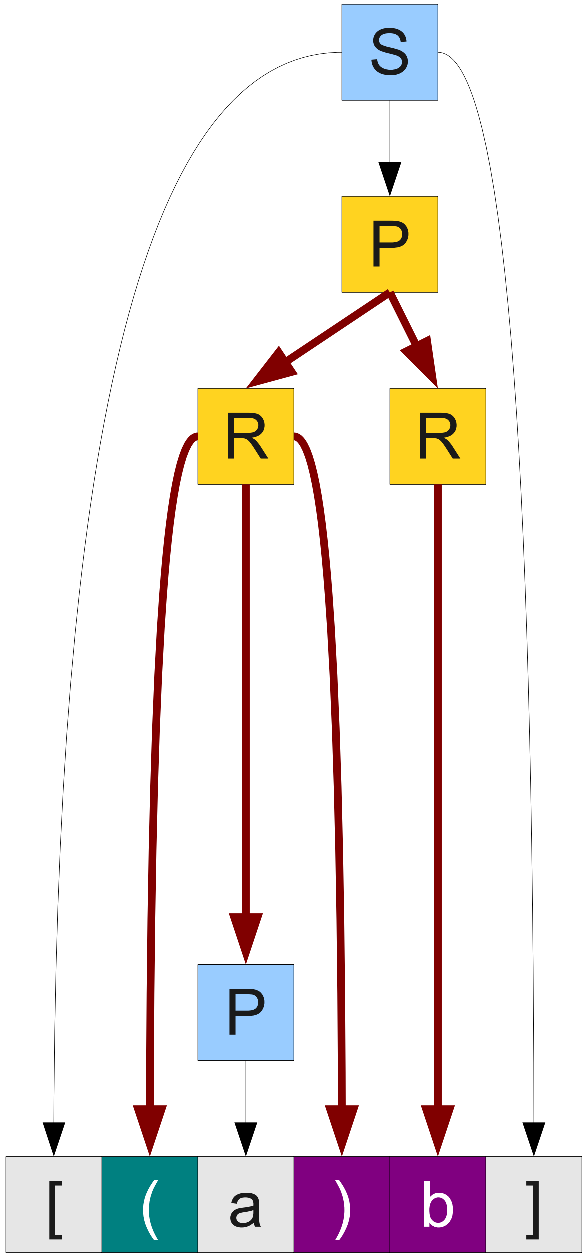


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

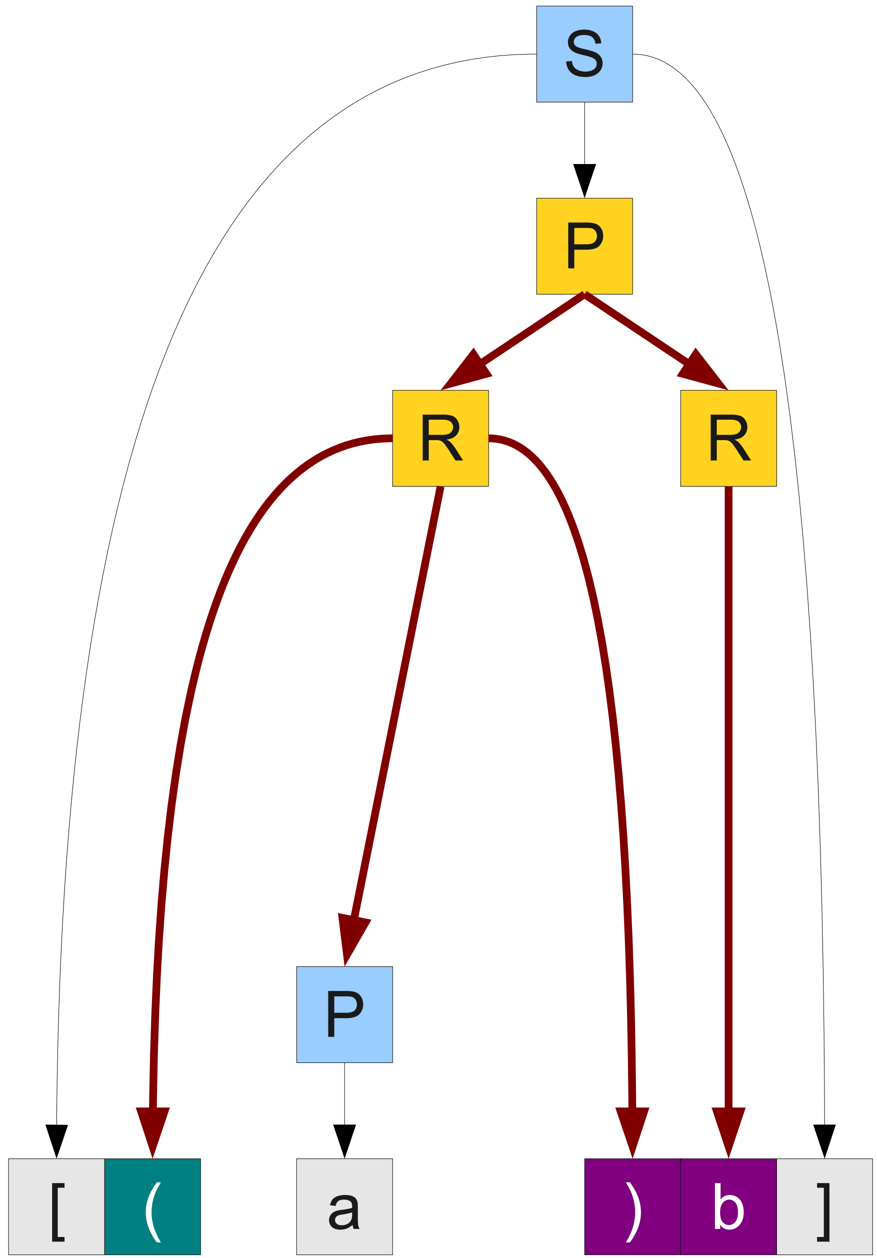


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

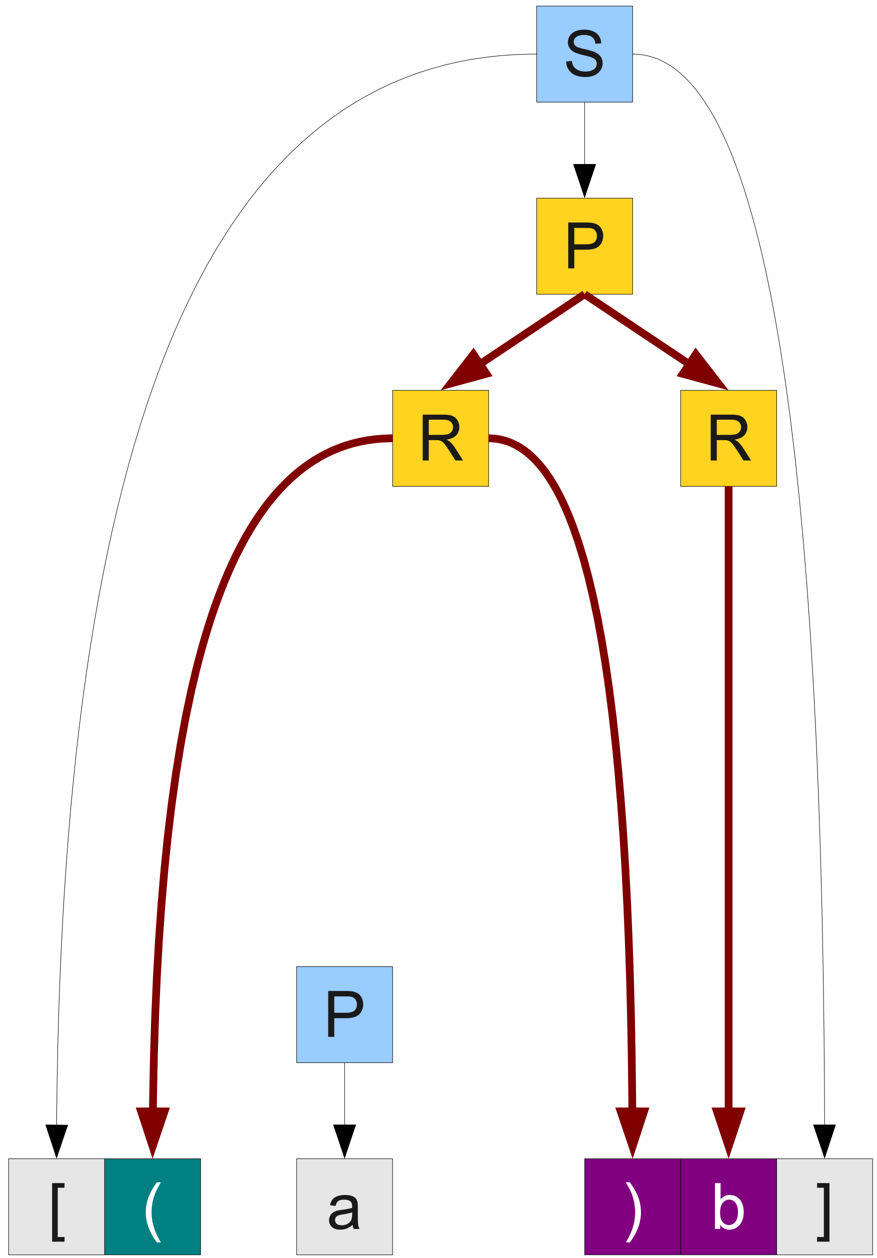


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

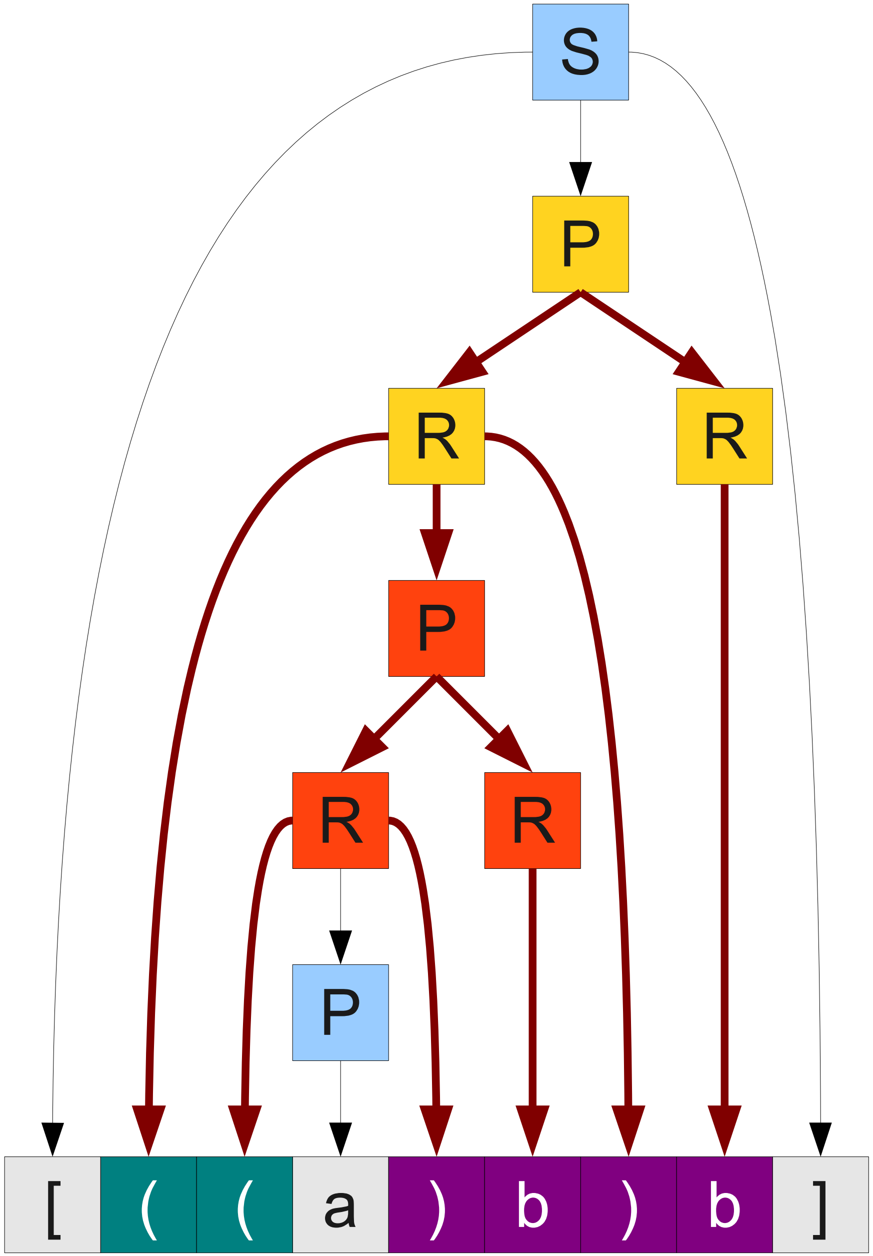


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

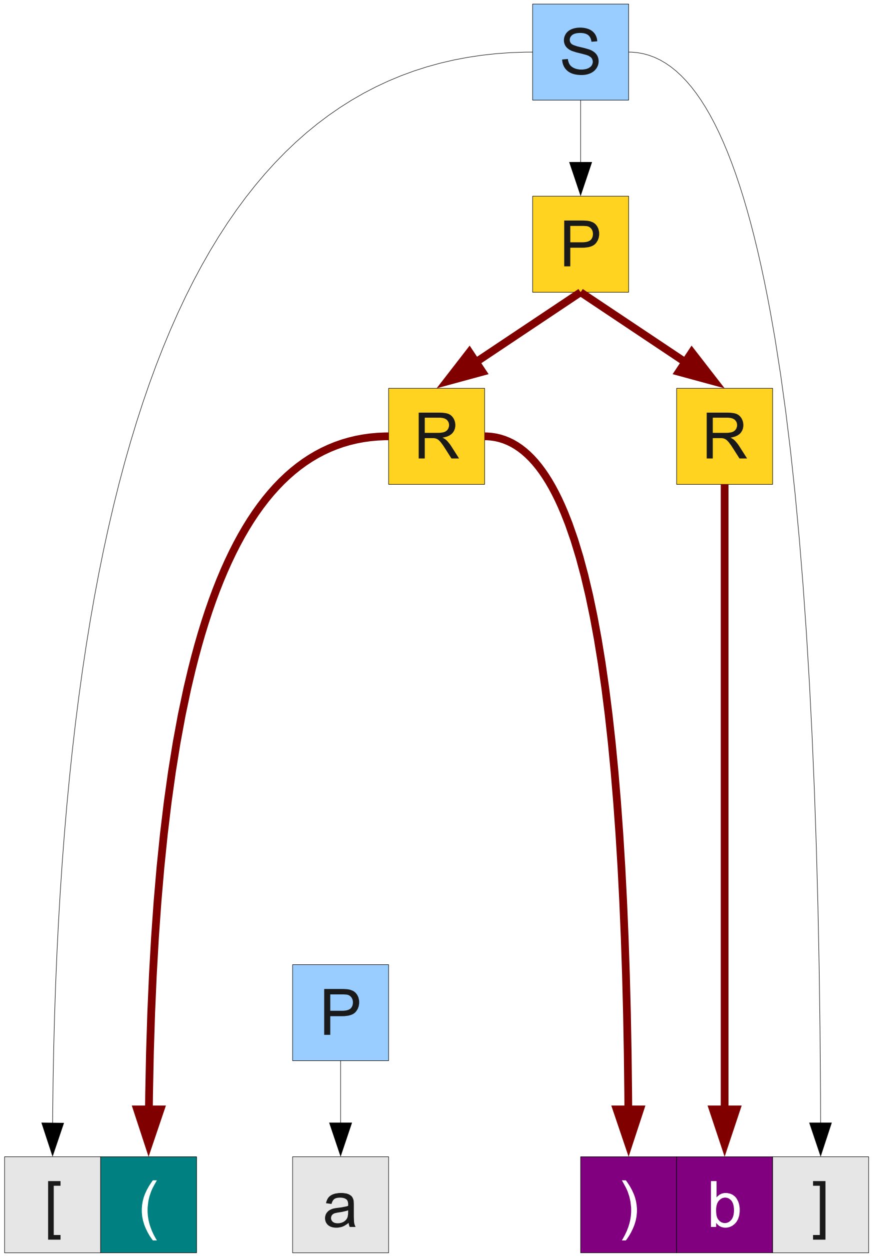


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

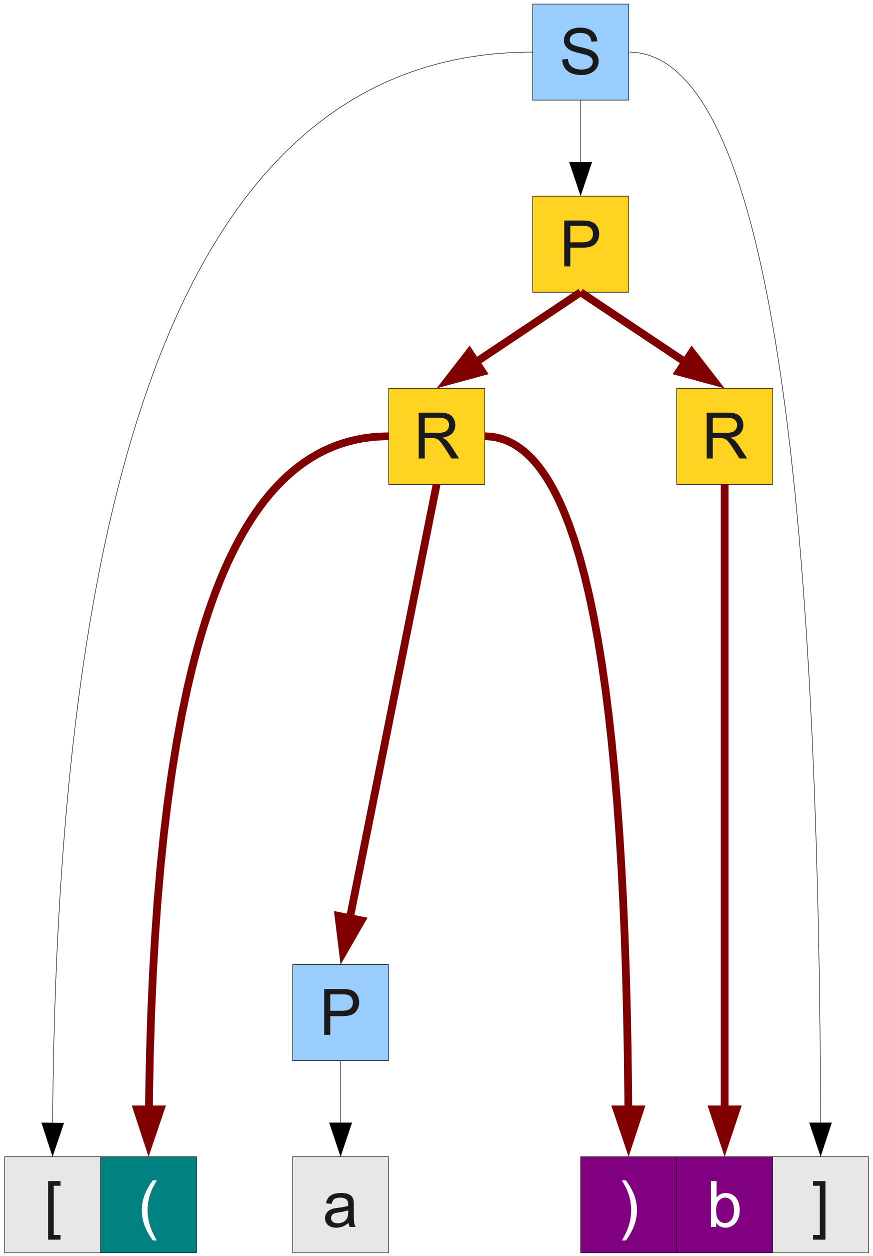


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

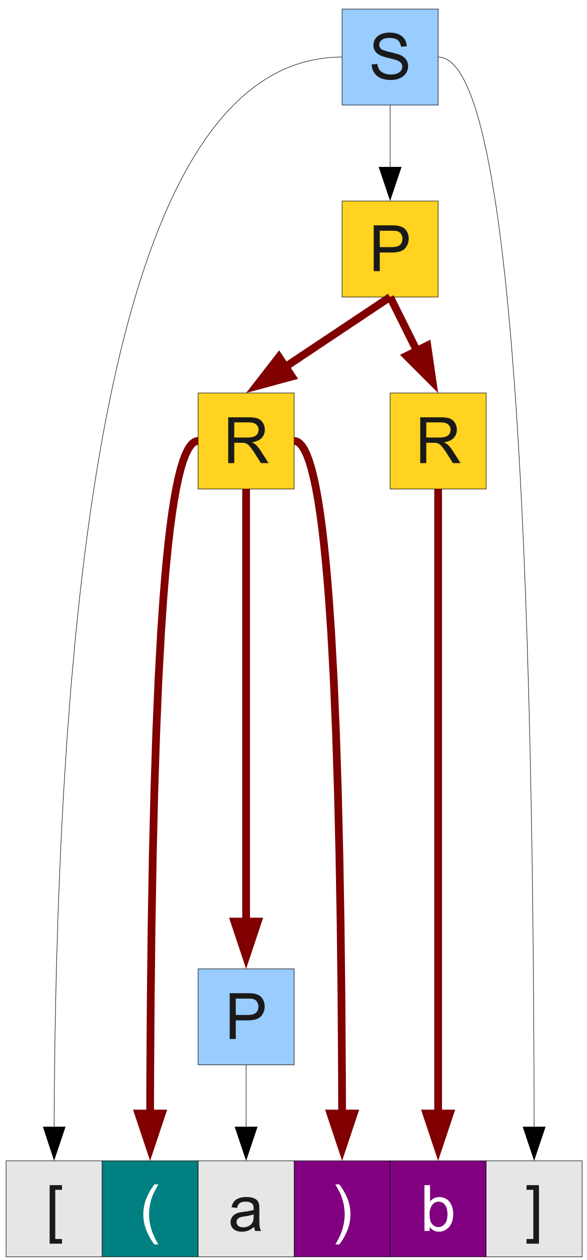


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

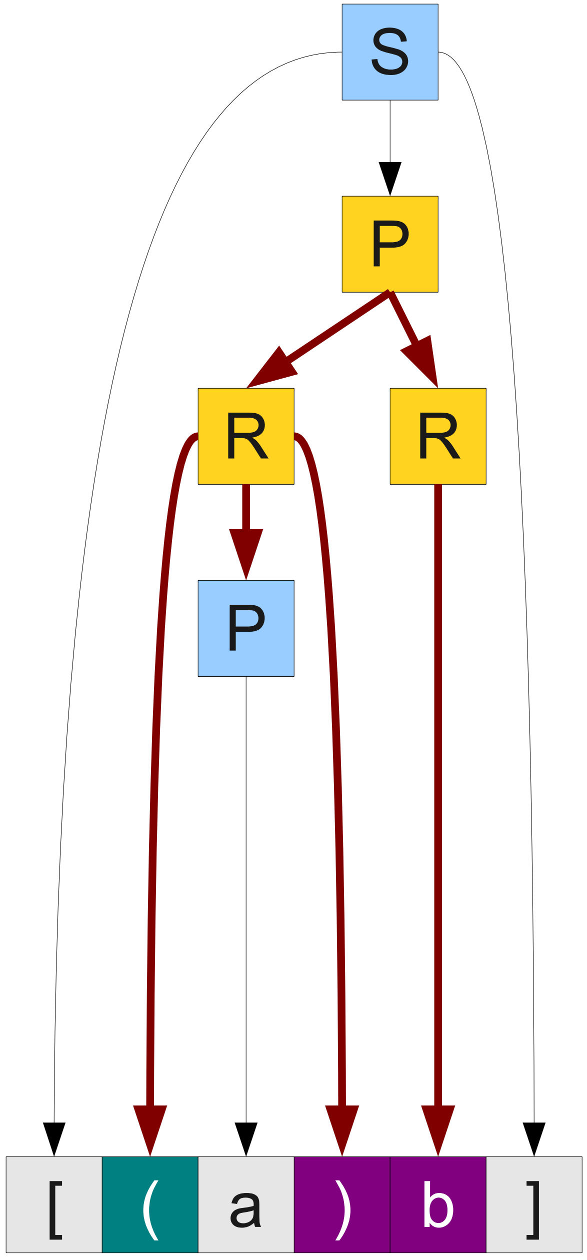


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

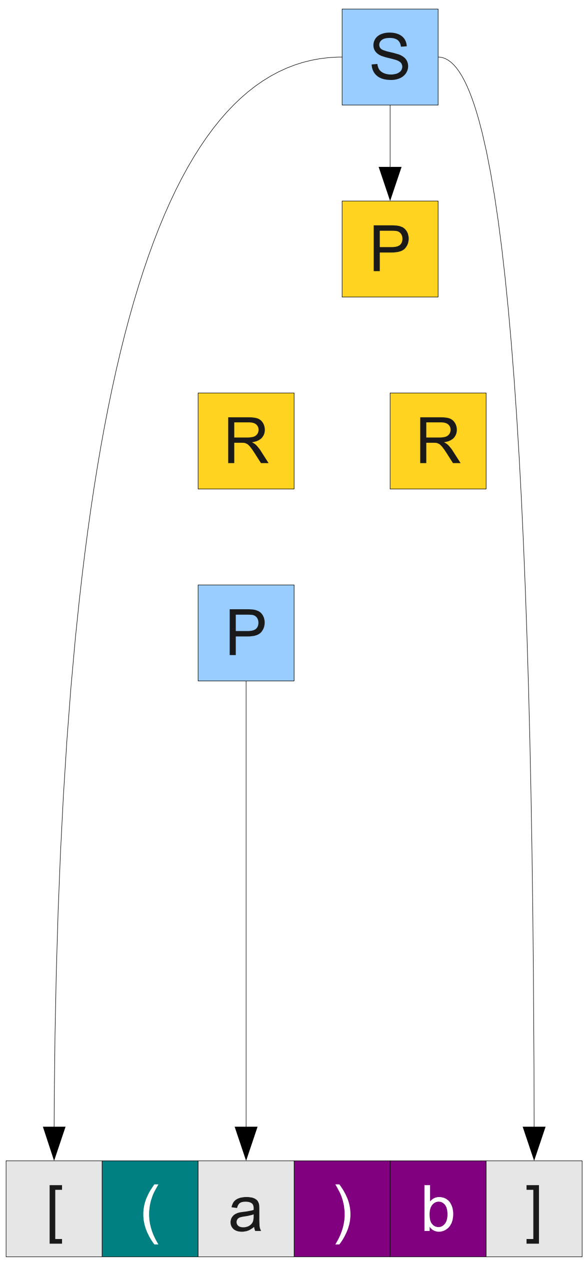


Parse Trees Revisited

S → [P]

P → RR | a

R → (P) | b

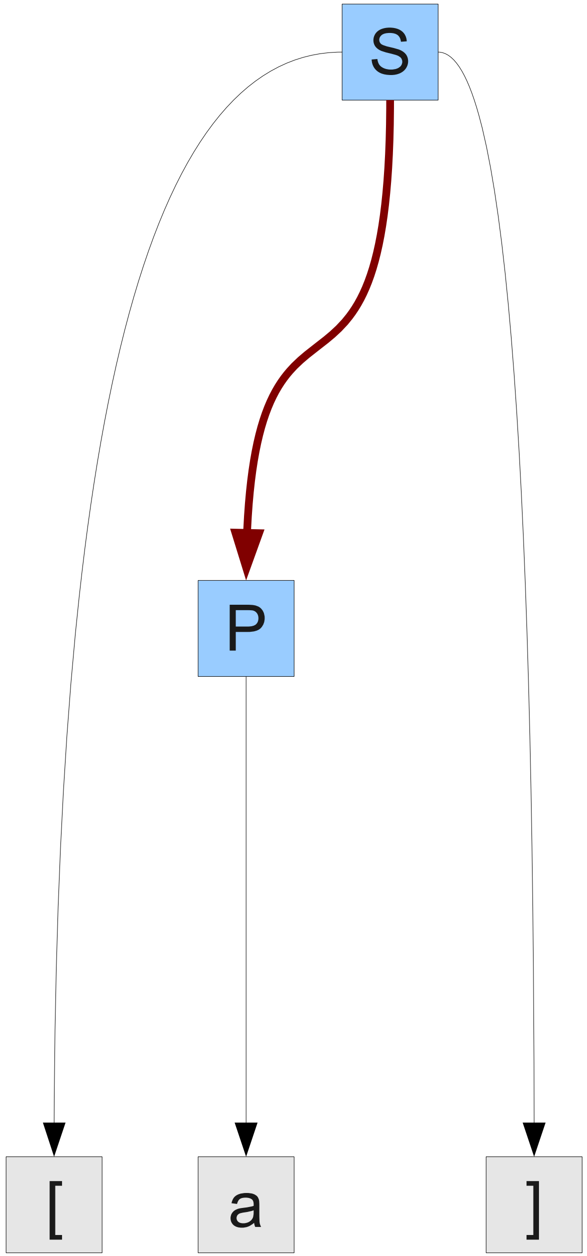


Parse Trees Revisited

$S \rightarrow [P]$

$P \rightarrow RR \mid a$

$R \rightarrow (P) \mid b$

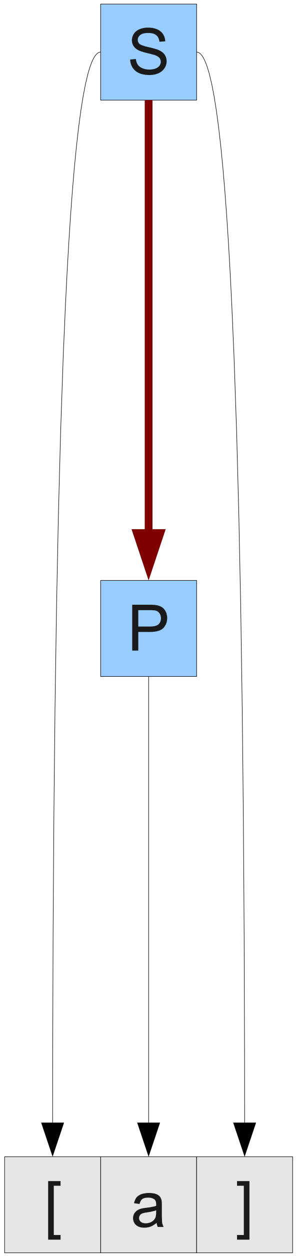


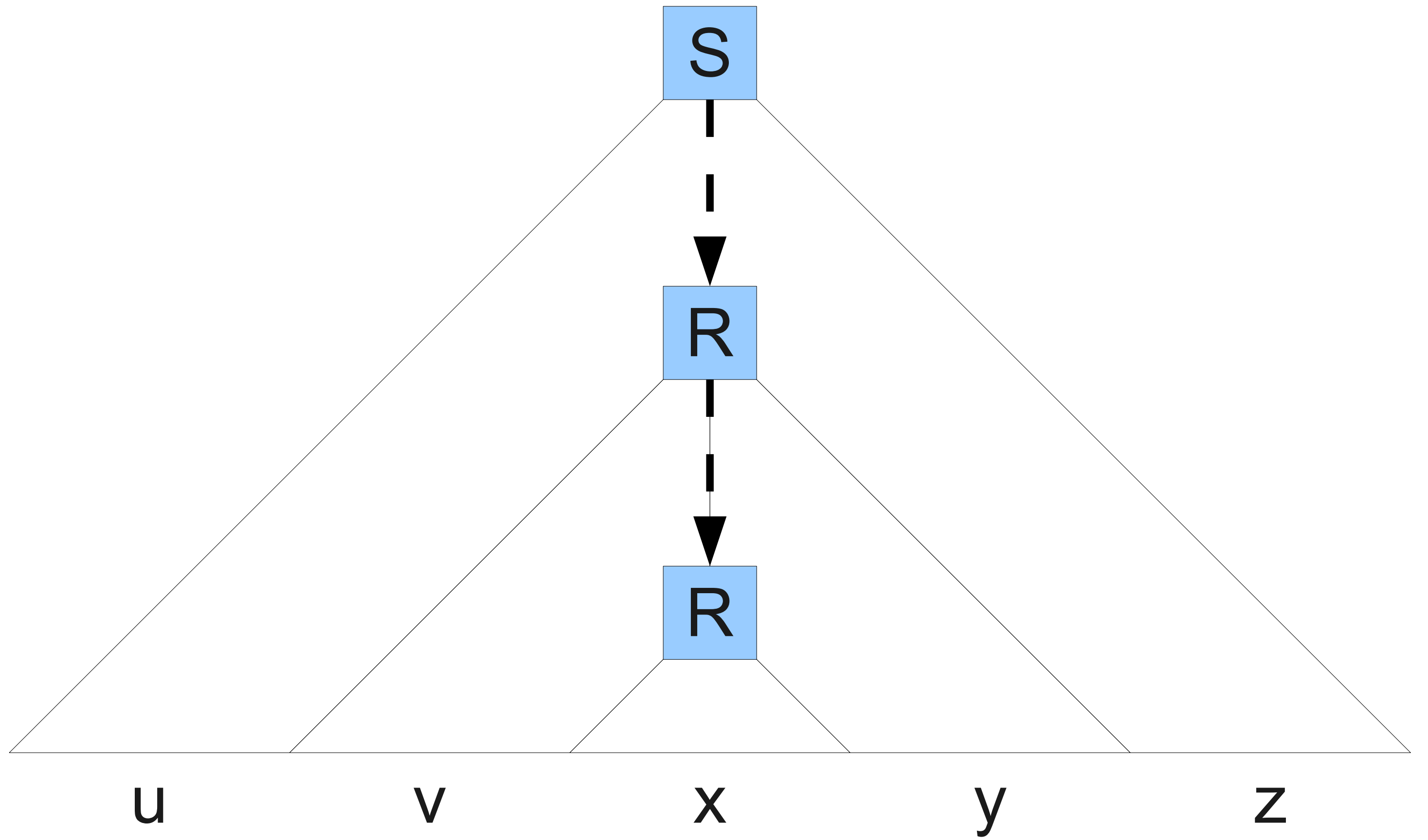
Parse Trees Revisited

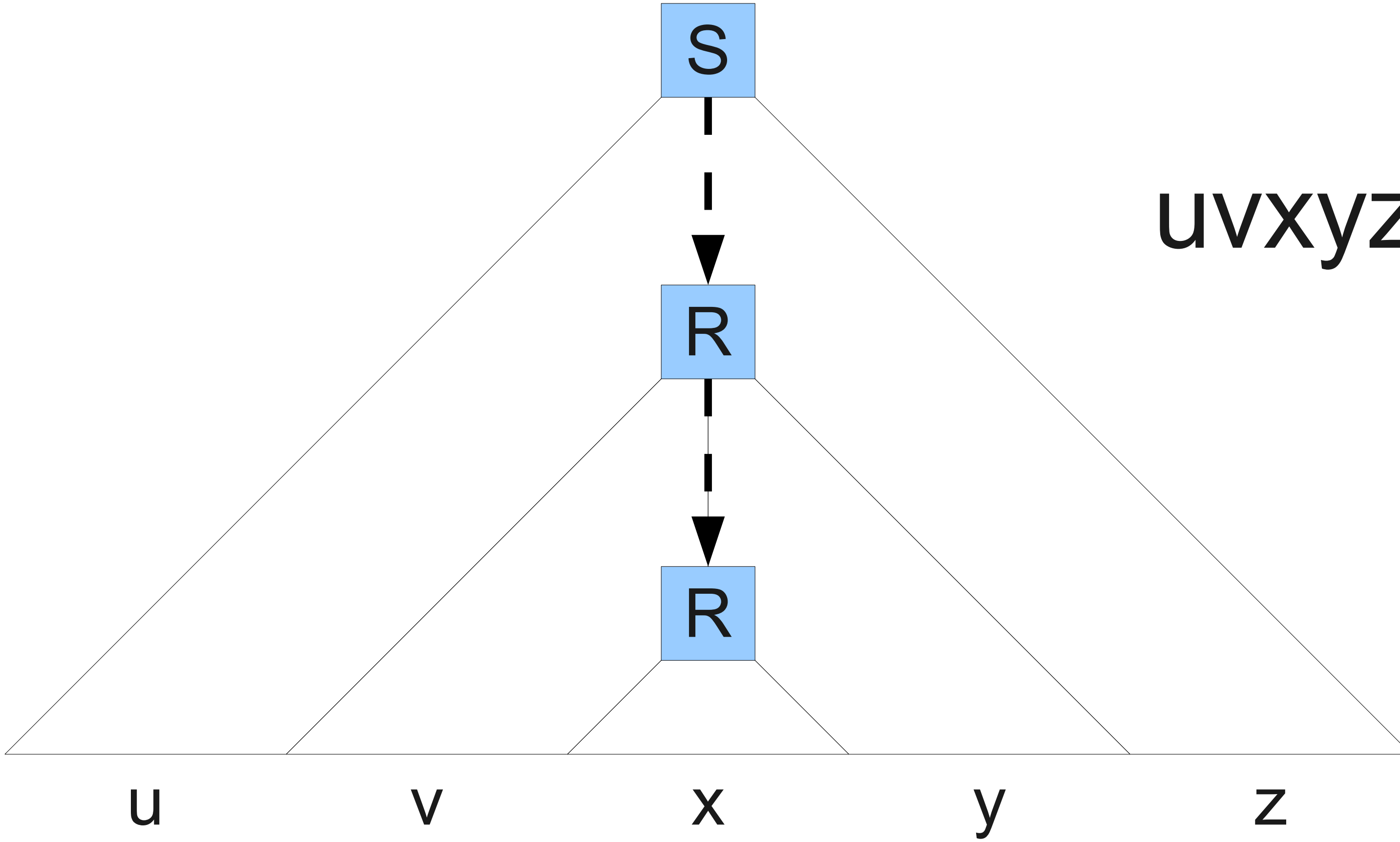
$S \rightarrow [P]$

$P \rightarrow RR \mid a$

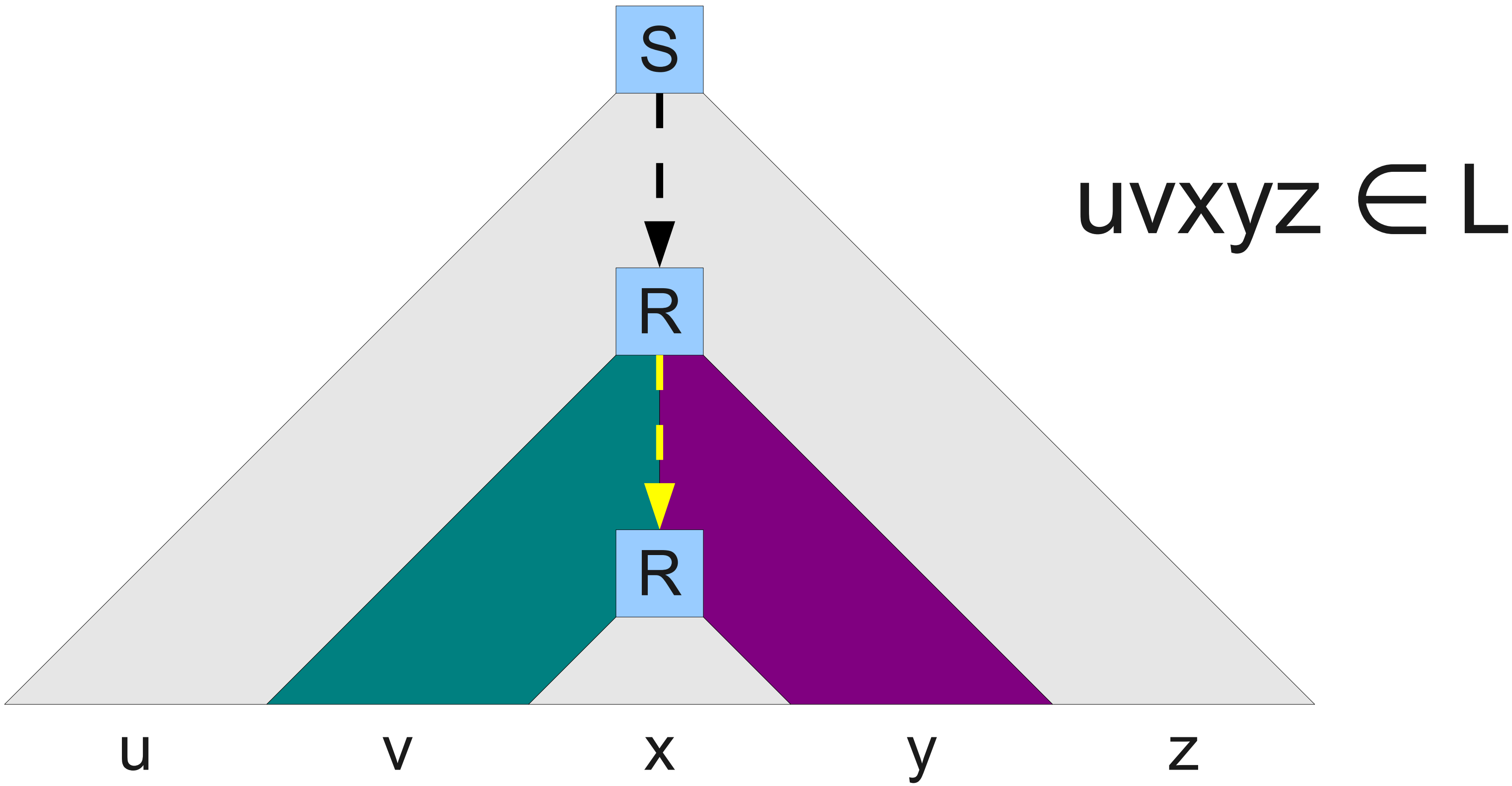
$R \rightarrow (P) \mid b$

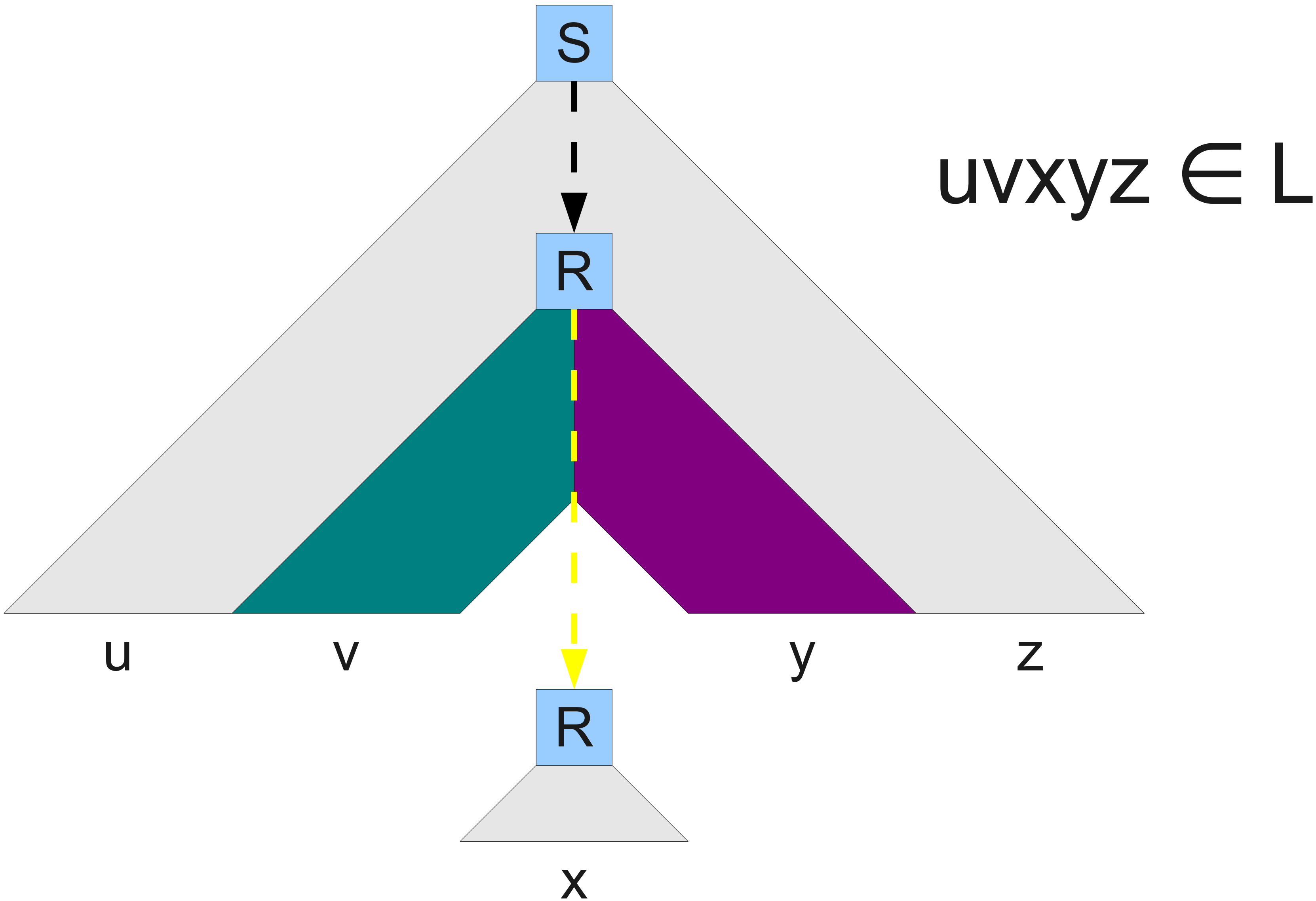


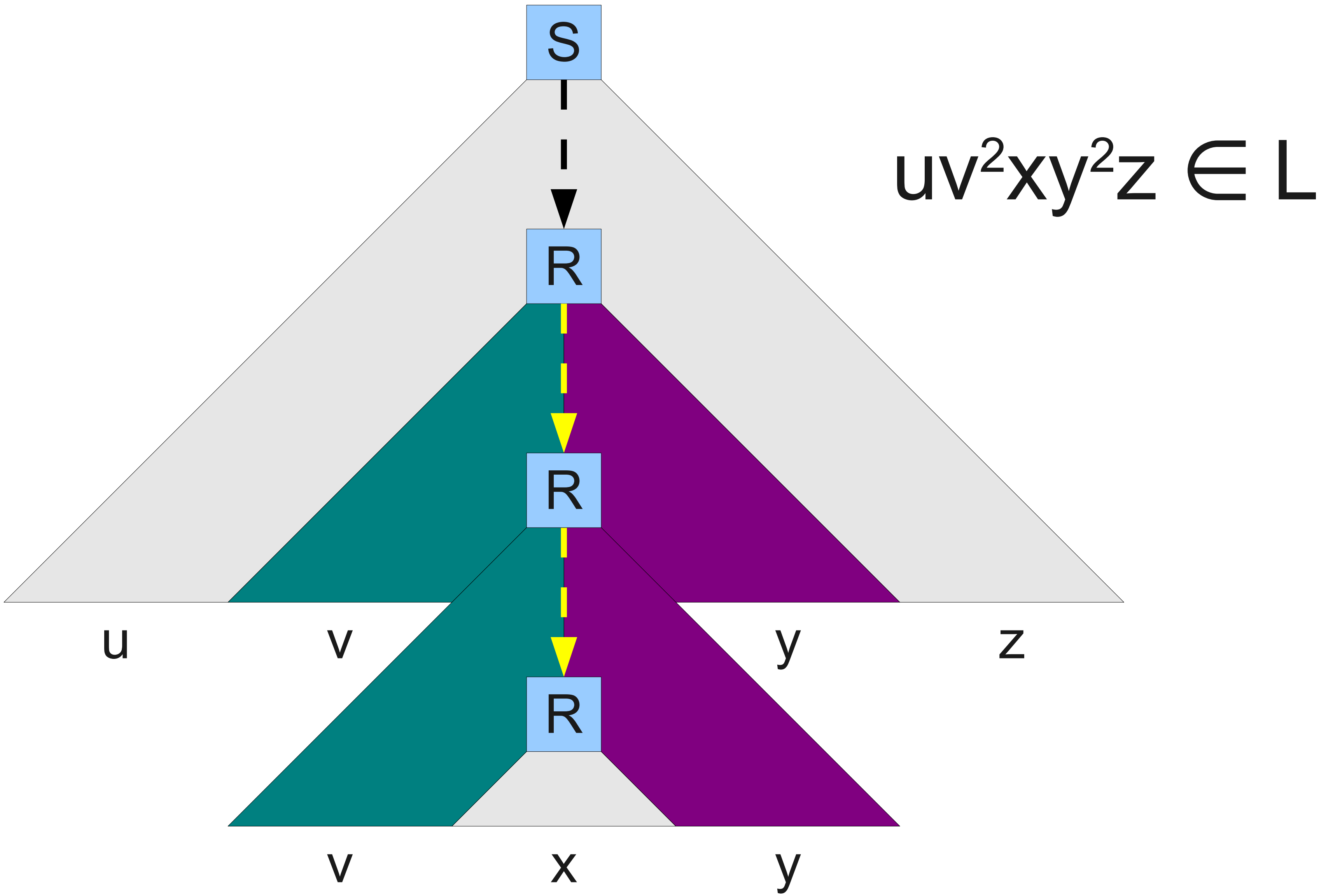

S
R
R
u
v
x
y
z



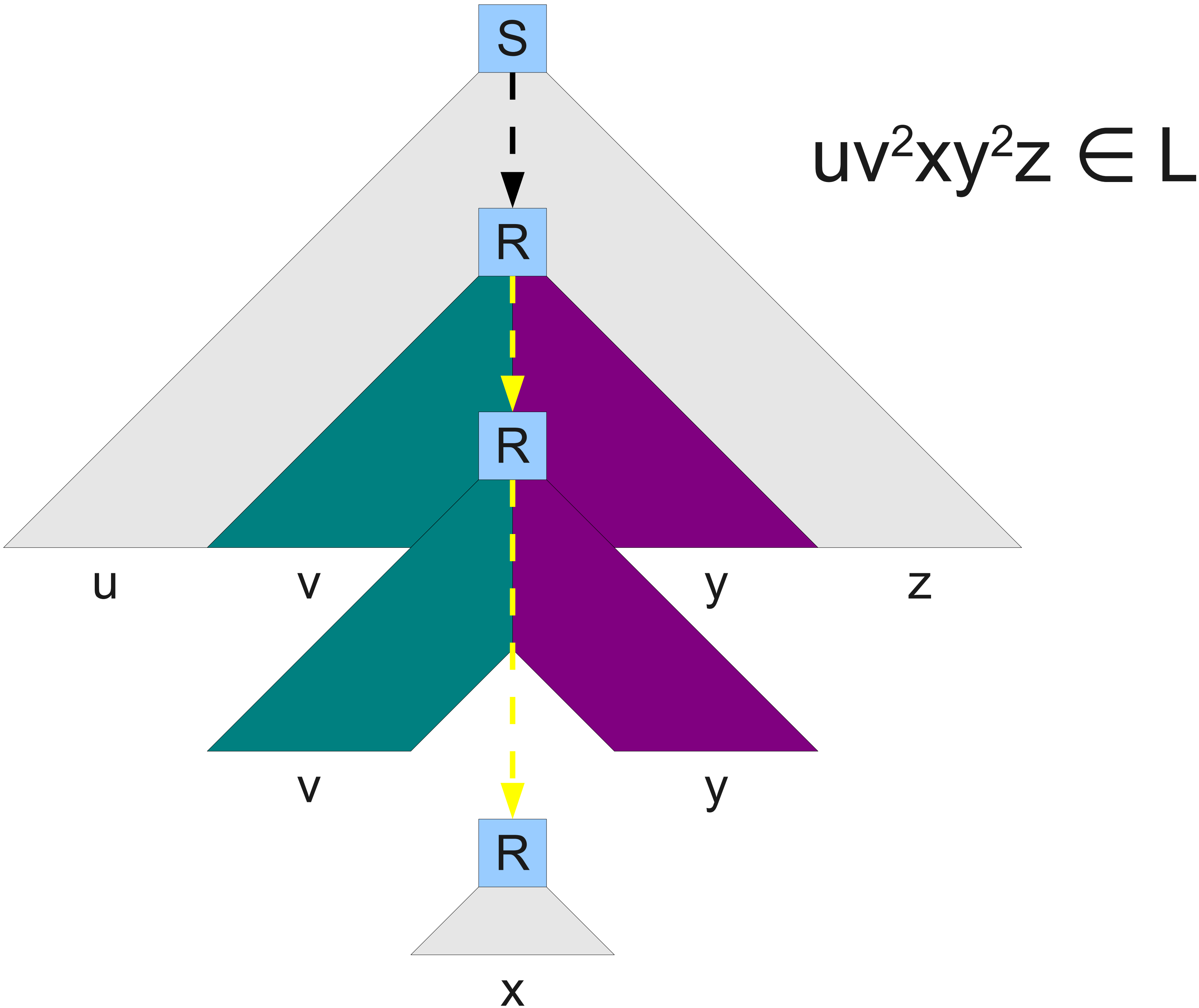
uvxyz ∈ L

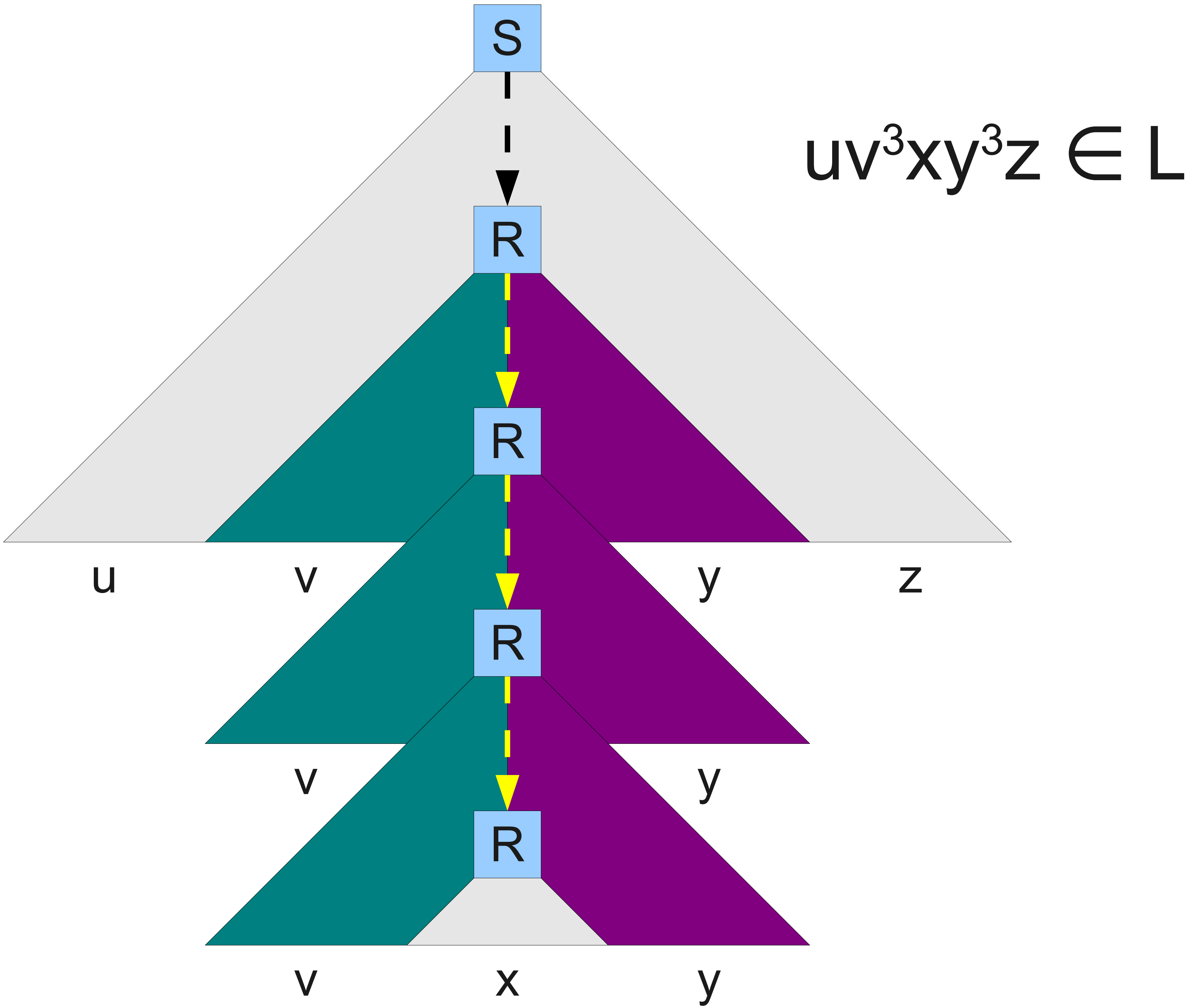

S
R
R
u
v
x
y
z
uvxyz ∈ L


S
R
uvxyz ∈ L
u
v
y
z
R
x

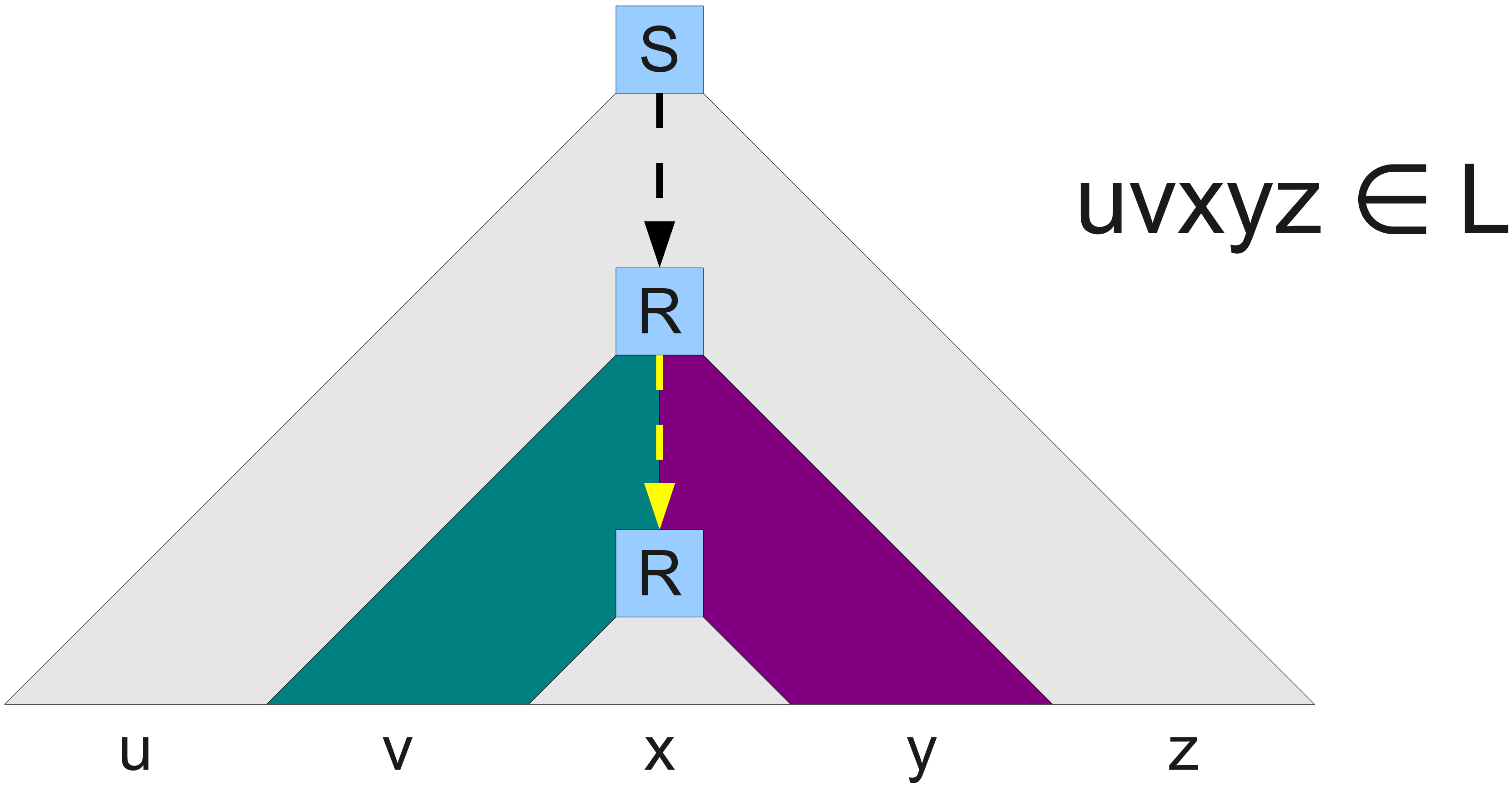


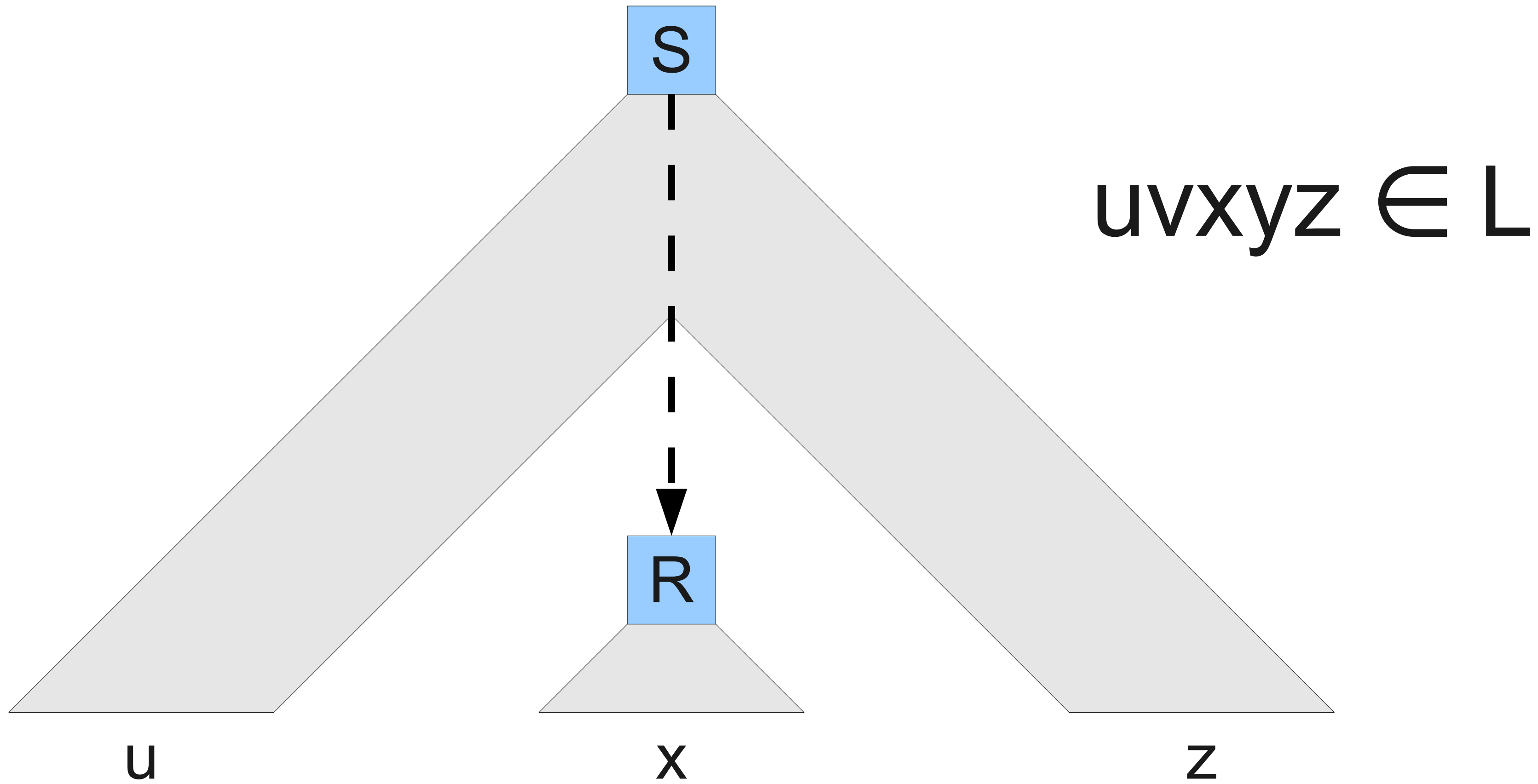
S
R
R
R
u
v
y
z
v
x
y
uv²xy²z ∈ L

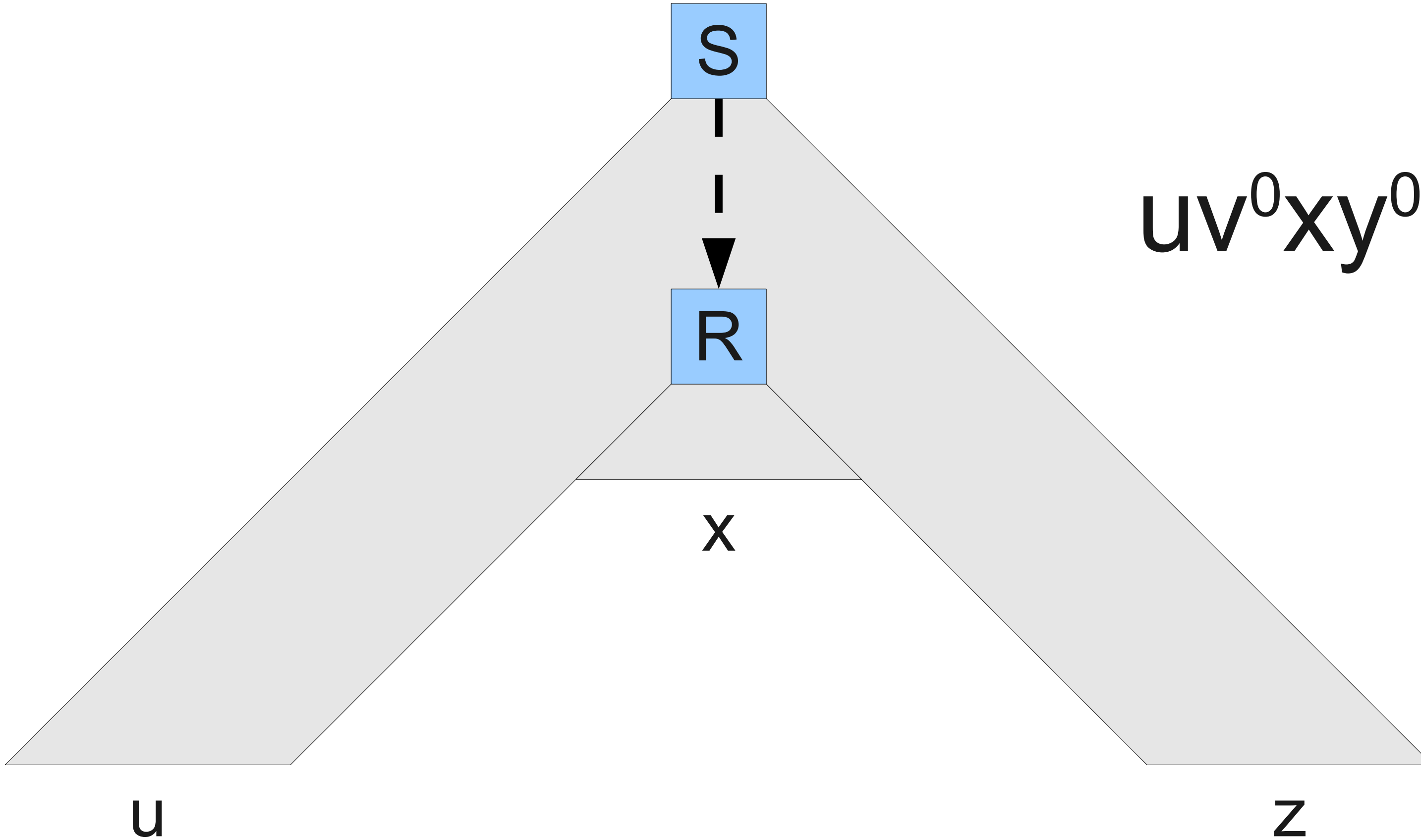

S
R
R
R
u
v
y
z
v
y
x
$uv^2xy^2z \in L$



$uv^3xy^3z \in L$
S
R
R
R
R
u
v
y
z
v
y
v
x
y


S
R
R
uvxyz ∈ L
u
v
x
y
z


S
uvxyz ∈ L
R
u
x
z



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

The Pumping Lemma for CFLS

For any context-free language L,

There exists a positive natural number n such that

For any w ∈ L with |w| ≥ n,

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

For any natural number i,

$w = uvxyz$,

$|vxy| \leq n$,

$|vy| > 0$

$uv^ixy^iz \in L$

The Pumping Lemma for CFLS

For any context-free language L,

There exists a positive natural number n such that

For any w ∈ L with |w| ≥ 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| ≤ n,

|vy| > 0

$uv^ixy^iz \in L$

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,

$|vy| > 0$

$uv^ixy^iz \in L$

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

$uv^ixy^iz \in L$

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.

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

The pumping length is not simple; see Sipser for details.

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 Game

ADVERSARY	YOU

The Pumping Lemma Game

ADVERSARY	YOU
Maliciously choose pumping length n.	

The Pumping Lemma Game

ADVERSARY	YOU
Maliciously choose pumping length n.	Cleverly choose a string $w \in L$, $\|w\| \geq n$

The Pumping Lemma Game

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 $\lvert vxy \rvert \leq n$	

The Pumping Lemma Game

ADVERSARY	YOU
Maliciously choose pumping length n.	
	Cleverly choose a string $w \in L$, $\|w\| \geq n$
Maliciously split $w = uvxyz$, with $\|vy\| > 0$ and $\|vxy\| \leq n$	
	Cleverly choose i such that $uv^ixy^iz \notin L$

The Pumping Lemma Game

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

The Pumping Lemma Game

$L = \{ 0^n 1^n 2^n \mid n \in \mathbb{N} \}$

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 i such that $uv^ixy^iz \notin L$
Grrr! Aaaargh!	

The Pumping Lemma Game

$L = \{ 0^n 1^n 2^n \mid n \in \mathbb{N} \}$

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

000111000

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free.

For any context-free language L,

There exists a positive natural number n such that

For any w ∈ L with |w| ≥ n,

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

For any natural number i,

w = uvxyz,

|vxy| ≤ n,

|vy| > 0

$uv^ixy^iz \in L$

Theorem: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

Proof: By contradiction; assume that L is context-free. Let n be the 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$.

For any context-free language L,

There exists a positive natural number n such that

For any w ∈ L with |w| ≥ n,

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

For any natural number i,

w = uvxyz,

|vxy| ≤ n,

|vy| > 0

$uv^ixy^iz \in L$

Theorem: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then w ∈ L and |w| = 3n ≥ 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.

Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. We consider two cases for vxy:

Case 1: vxy is completely contained in 0^n, 1^n, or 2^n.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

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

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

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

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

Case 2: vxy either consists of 0s and 1s or of 1s and 2s (it cannot consist of all three symbols, because $|vxy| \leq n$). Then if vxy has no 2s in it, $uv^2xy^2z \notin L$ because it contains more 0s or 1s than 2s, and similarly if vxy has no 0s in it $uv^2xy^2z \notin L$ because it contains more 1s or 2s than 0s. This contradicts 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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

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

In either case we reach a contradiction, so our L is not context-free.

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: $L = \{0^n1^n2^n \mid n \in \mathbb{N}\}$ is not context-free.
Proof: By contradiction; assume that L is context-free. Let n be the length guaranteed by the pumping lemma. Consider the string $w = 0^n1^n2^n$. Then $w \in L$ and $|w| = 3n \geq n$. Then we can write $w = uvxyz$ such that $|vxy| \leq n$, $|vy| > 0$, and for any natural number i, $uv^ixy^iz \in L$. 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. This contradicts the pumping lemma.

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

In either case we reach a contradiction, so our L is not context-free. ■

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) Closure Properties of CFLs

(Non) Closure Properties of CFLs

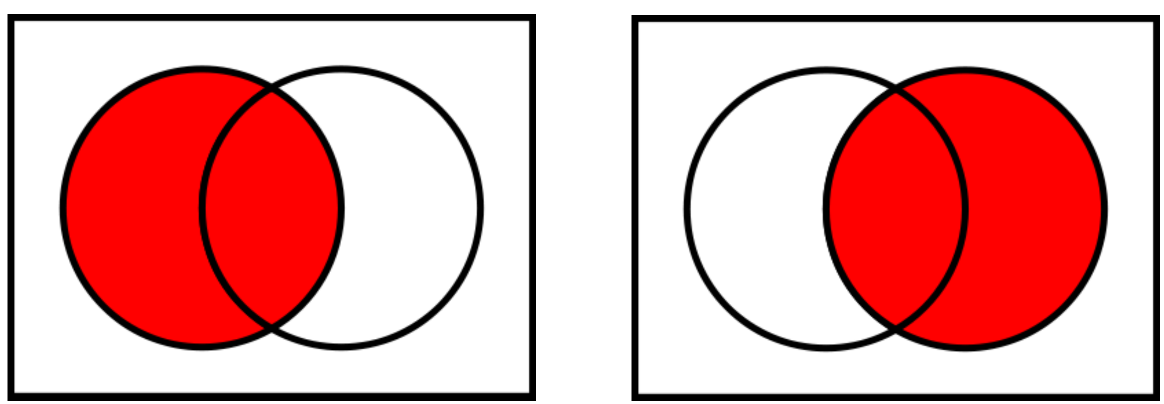
- Now that we have a single non-context-free language, we can prove that CFLs are not closed under certain operations.
- Let $L_1 = \{ 0^n1^n2^m \mid n, m \in \mathbb{N} \}$
- Let $L_2 = \{0^m1^n2^n \mid n, m \in \mathbb{N} \}$
- Both of these languages are context-free.
  - Can either find an explicit CFG, or note that these languages are the concatenation of two CFLs.
- But $L_1 \cap L_2 = \{ 0^n1^n2^n \mid n \in \mathbb{N} \}$, 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

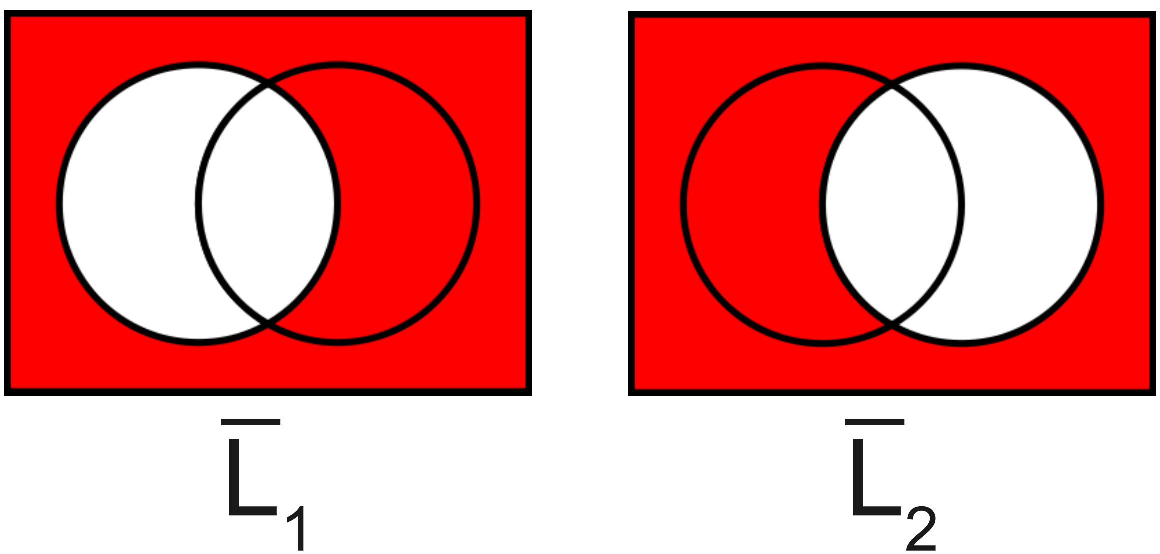
- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



L_1 L_2

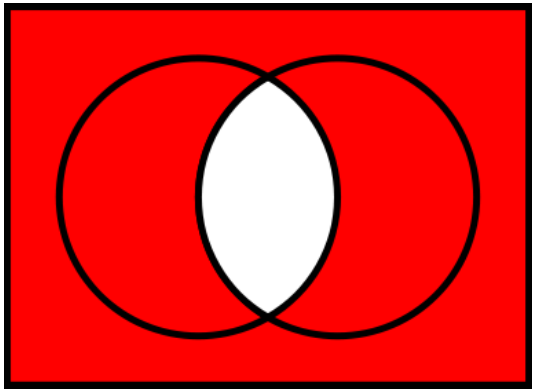
(Non) Closure under Complement

- 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.



(Non) Closure under Complement

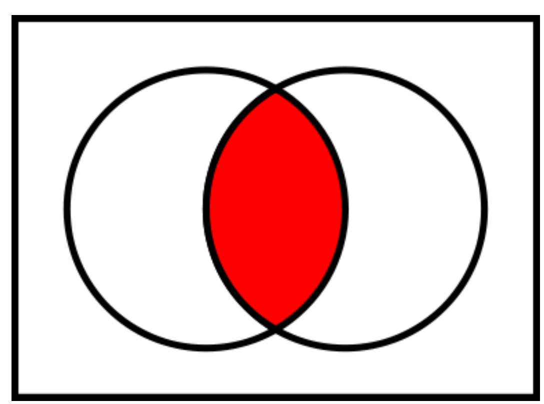
- Recall that if L is regular, $\overline{L}$ is regular as well.
- However, if L is context-free, $\overline{L}$ may not be a context-free language.
- Intuition: Using union and complement, we can construct the intersection.



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

(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

- Recall that 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.

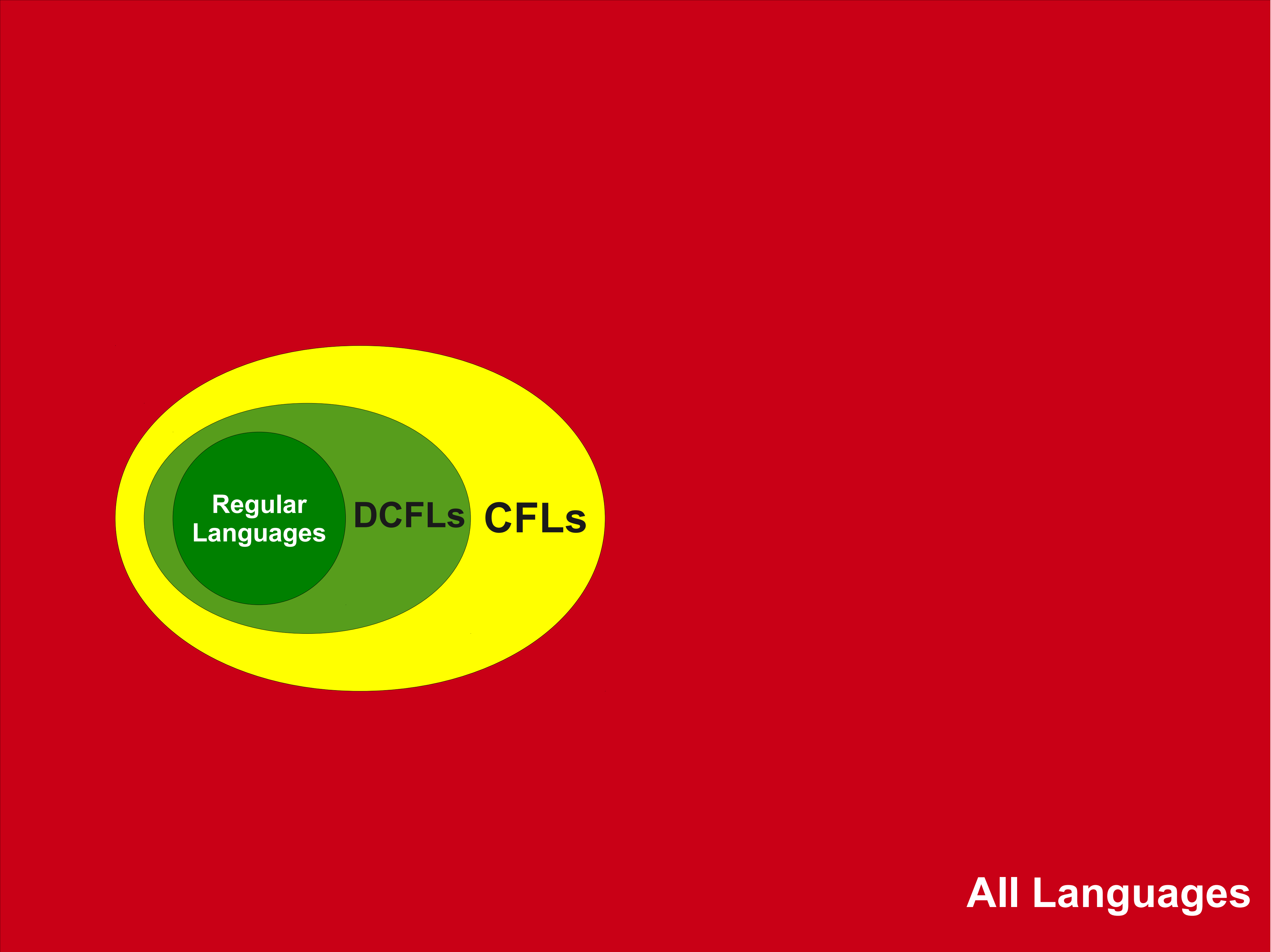
Summary of CFLs

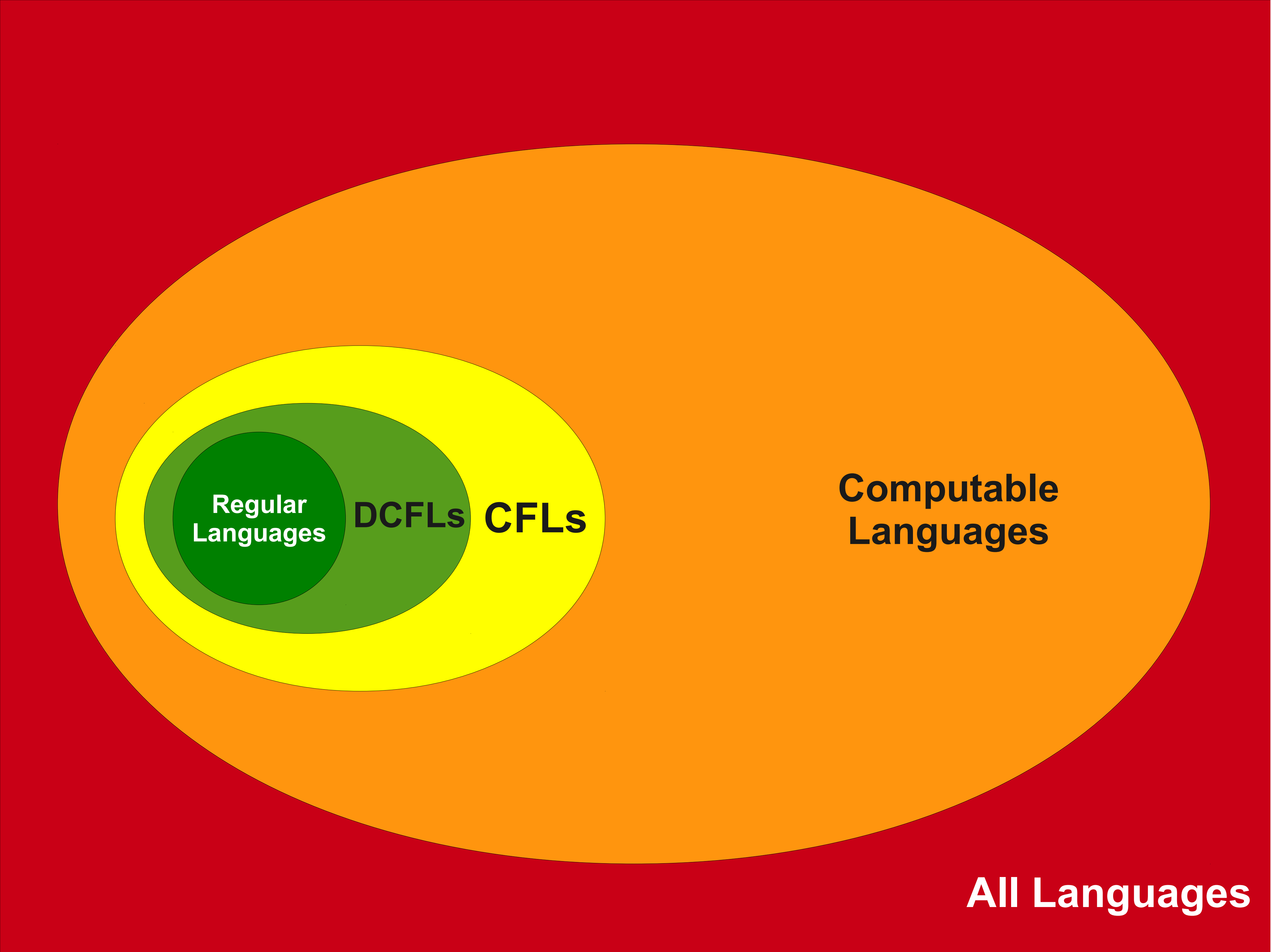
- CFLs are strictly more powerful than the regular languages.
- CFLs can be described by CFGs (generation) or PDAs (recognition).
- CFGs encompass two classes of languages – deterministic and nondeterministic CFLs.
- Context-free languages have a pumping lemma just as regular languages do.

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.


Regular Languages
DCFLs
CFLs
All Languages


Regular Languages
DCFLs
CFLs
Computable Languages
All Languages

That same drawing, to scale.

Defining Computability

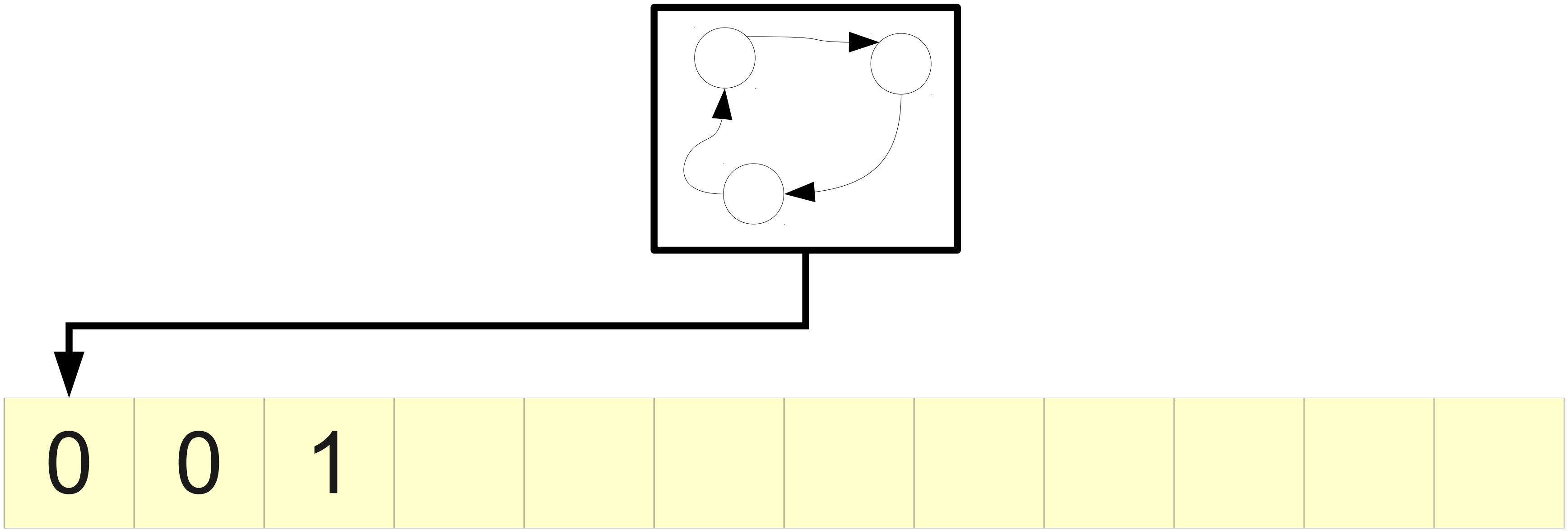
- In order to talk about what languages we could ever hope to recognize with a computer, we need to formalize our model of computation with an automaton.
- The standard automaton for this job is the **Turing machine**, named after Alan Turing, the “Father of Computer Science.”

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.

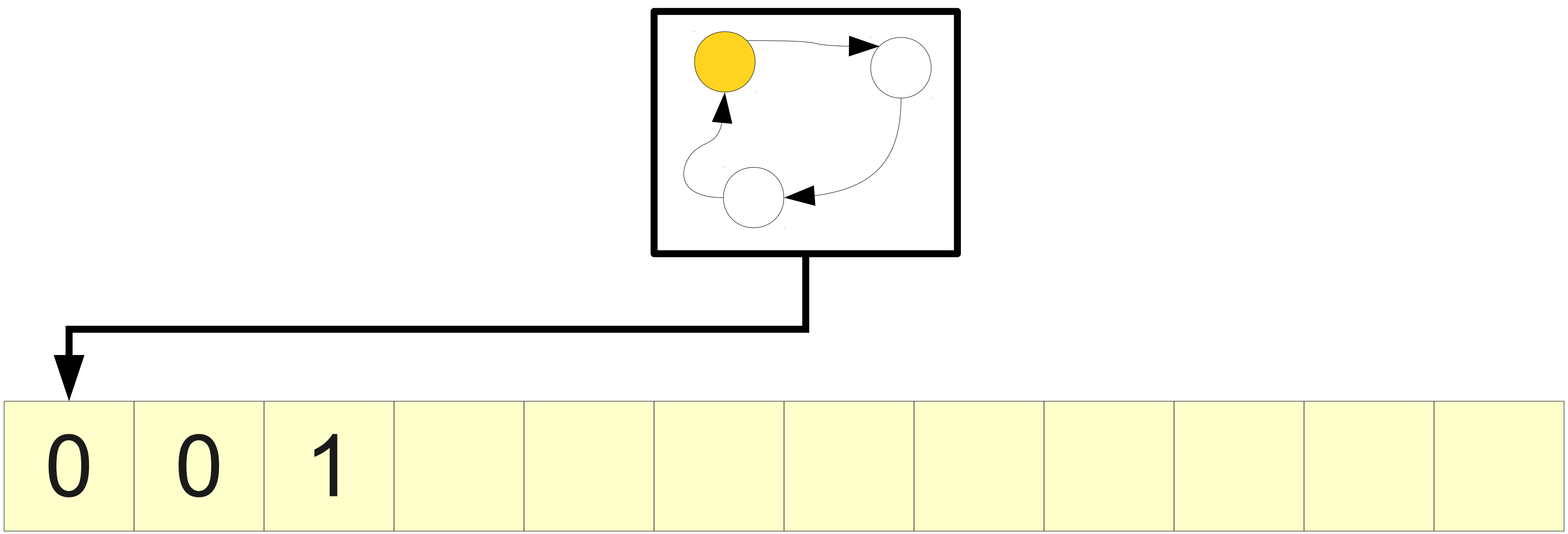
A Better Memory Device

- 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.



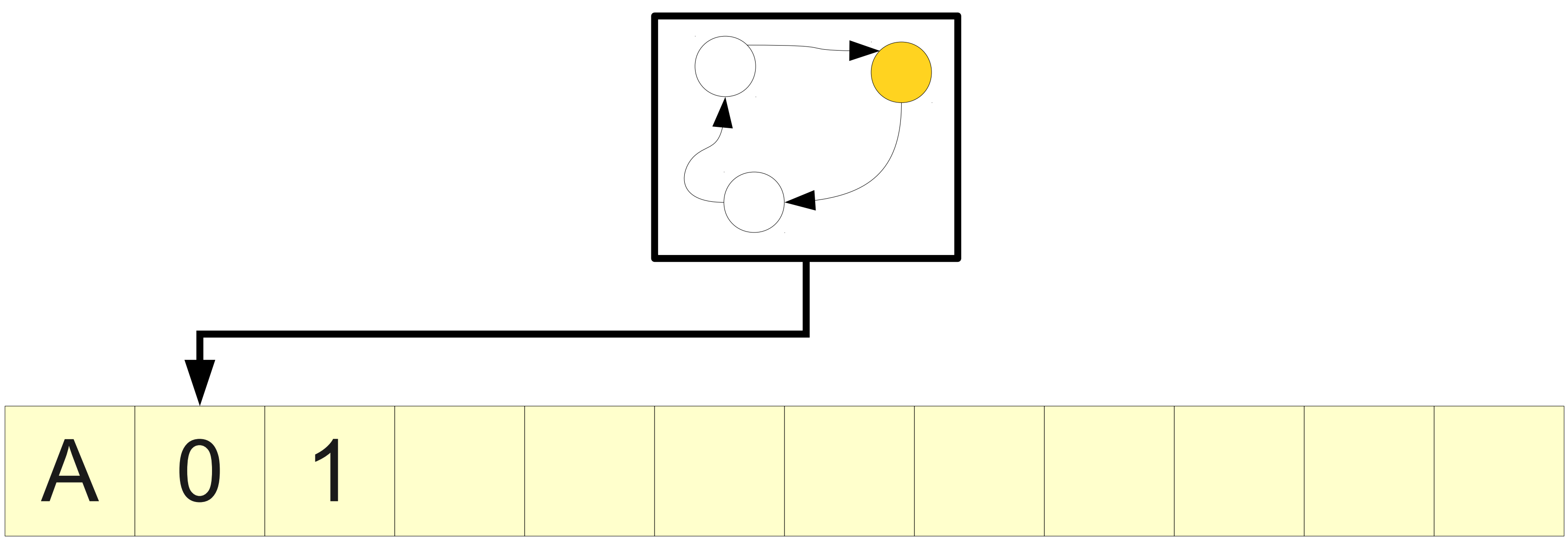
A Better Memory Device

- 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.



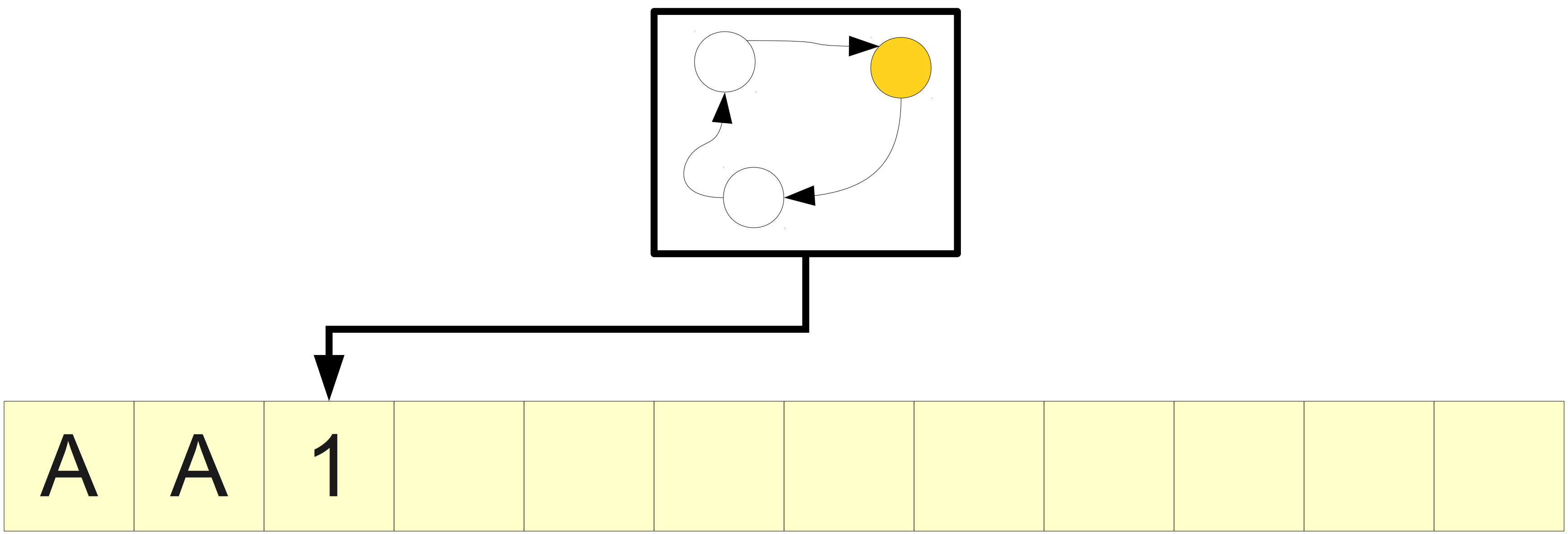
A Better Memory Device

- 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.



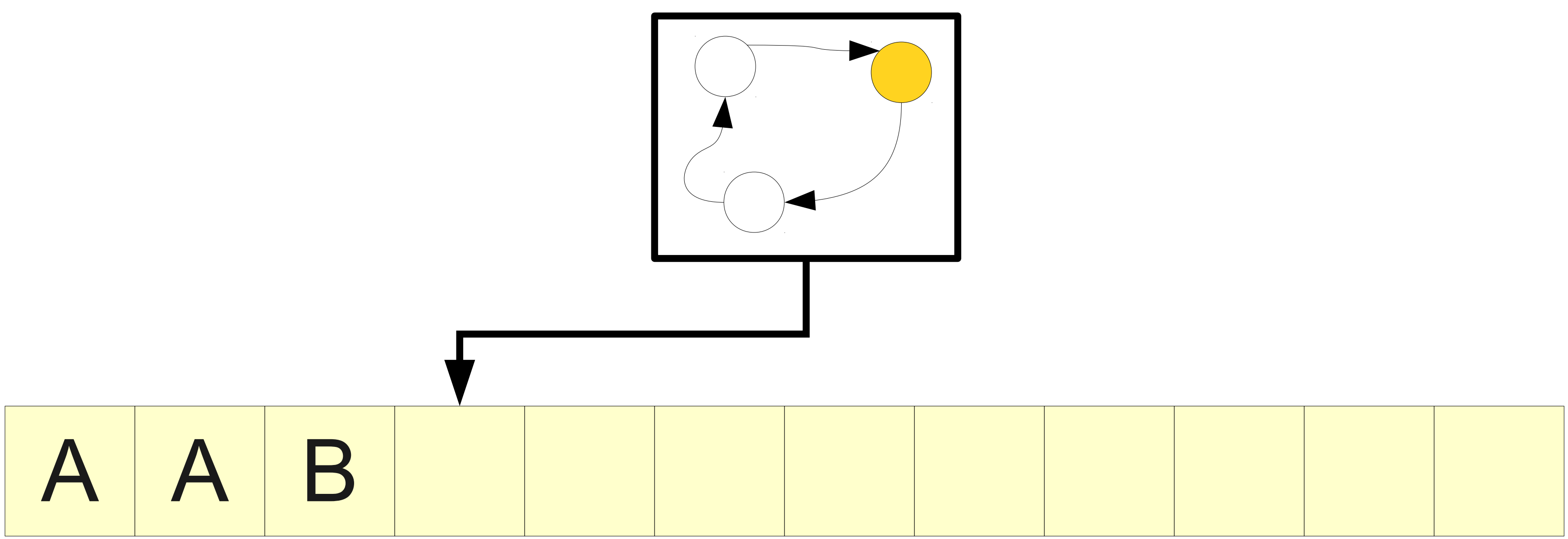
A Better Memory Device

- 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.



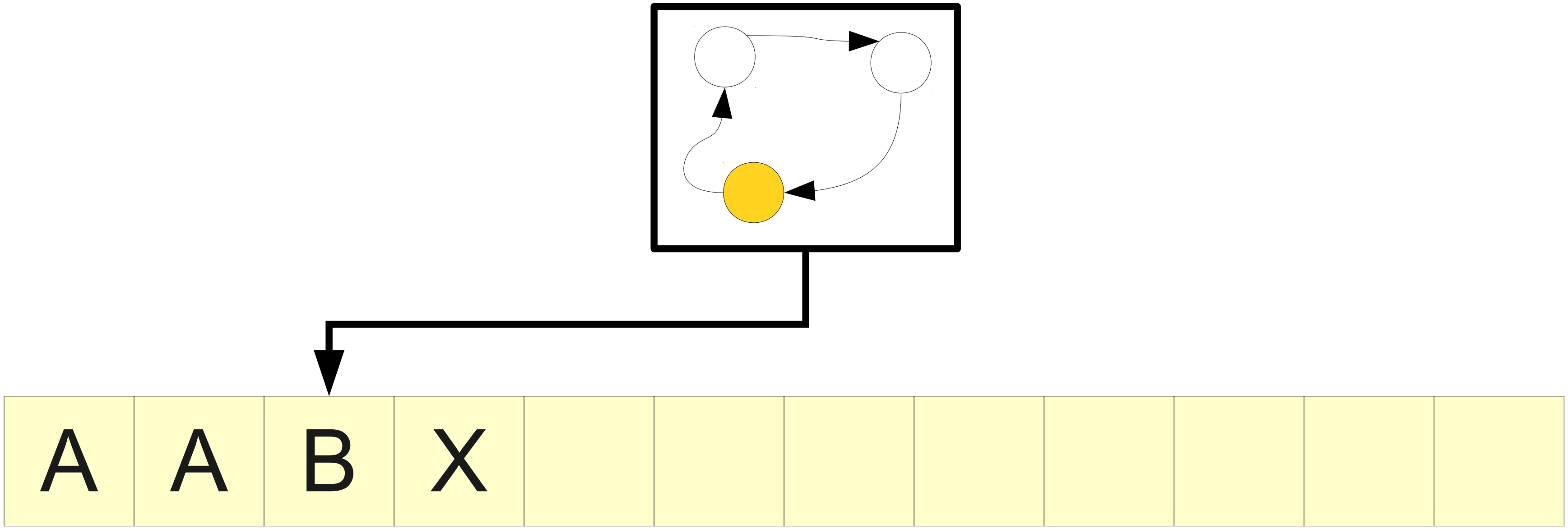
A Better Memory Device

- 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.



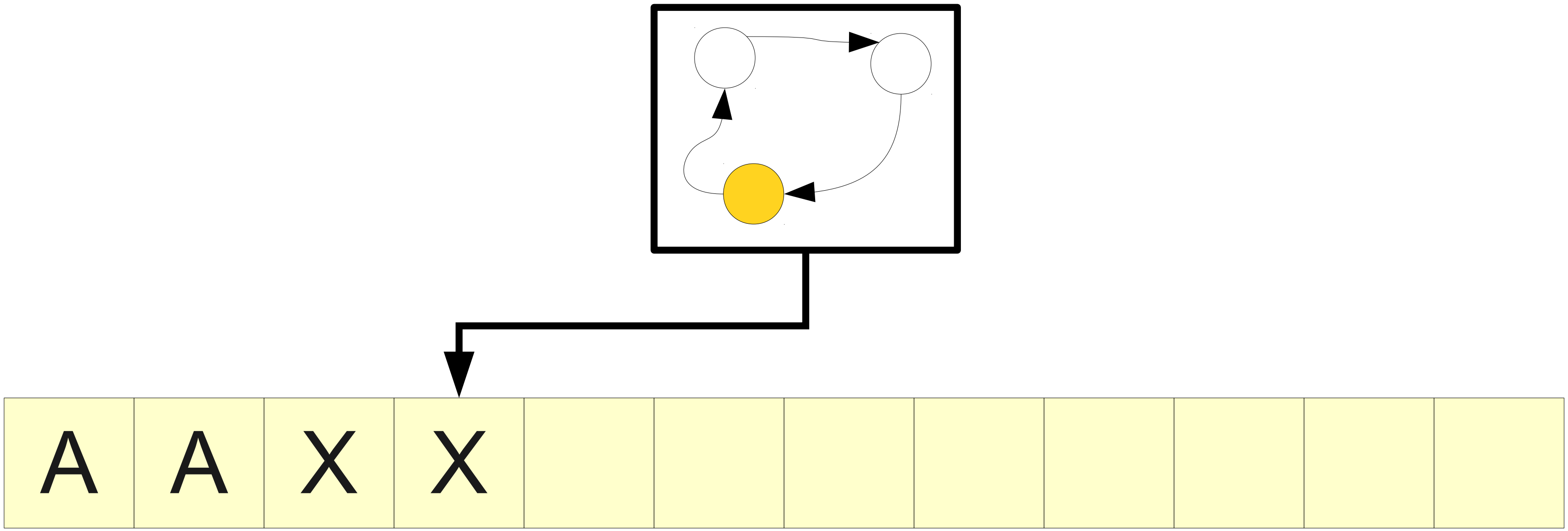
A Better Memory Device

- 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.



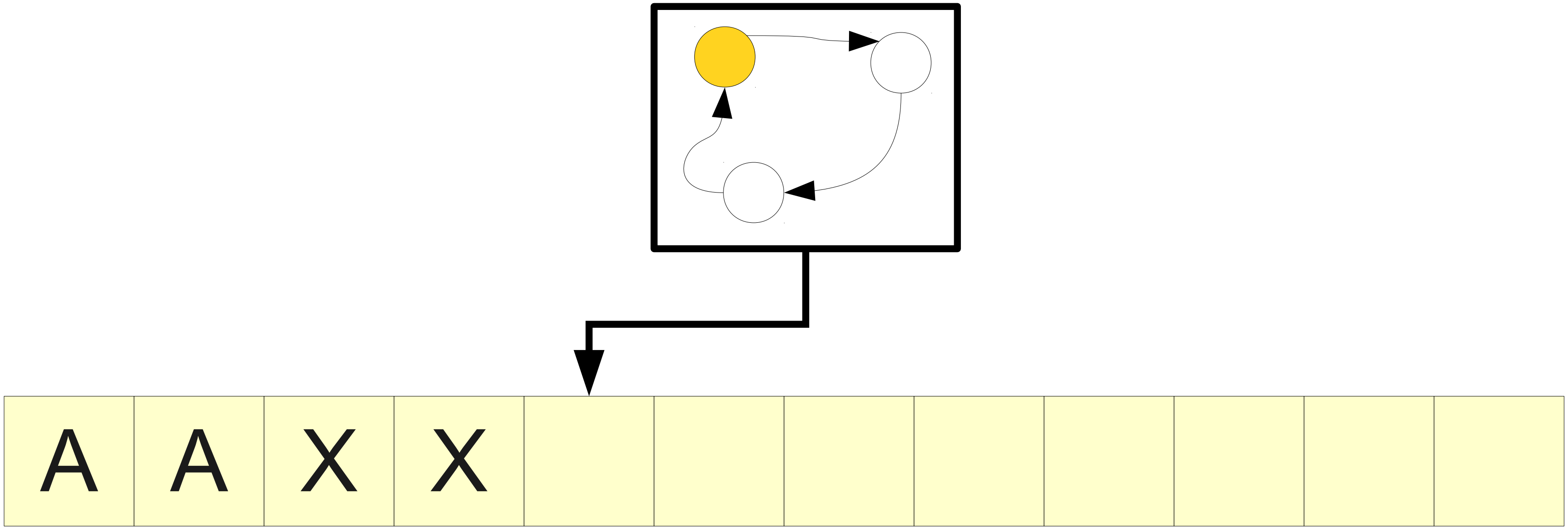
A Better Memory Device

- 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.



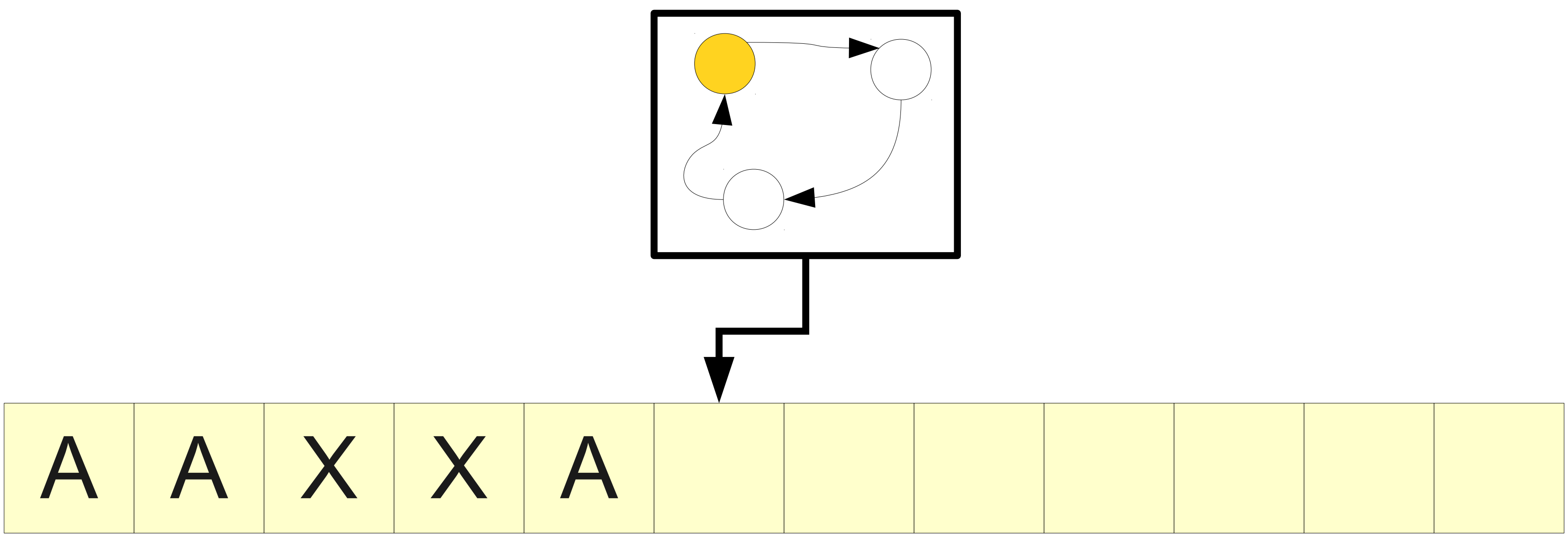
A Better Memory Device

- 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.



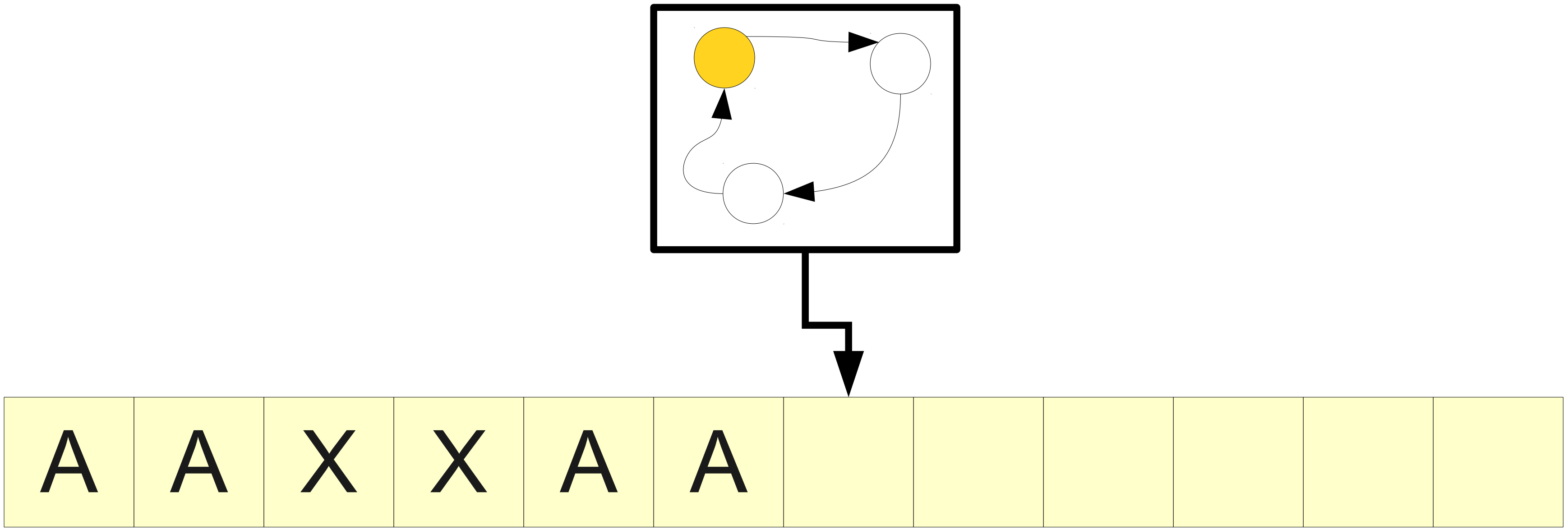
A Better Memory Device

- 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.



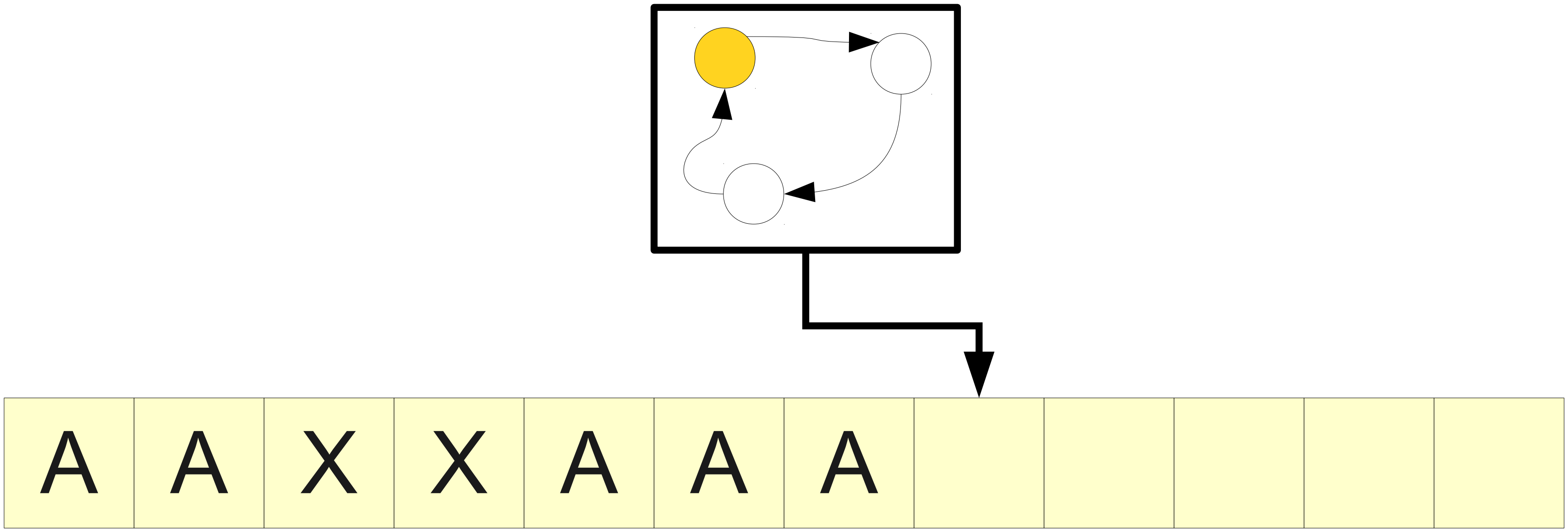
A Better Memory Device

- 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.

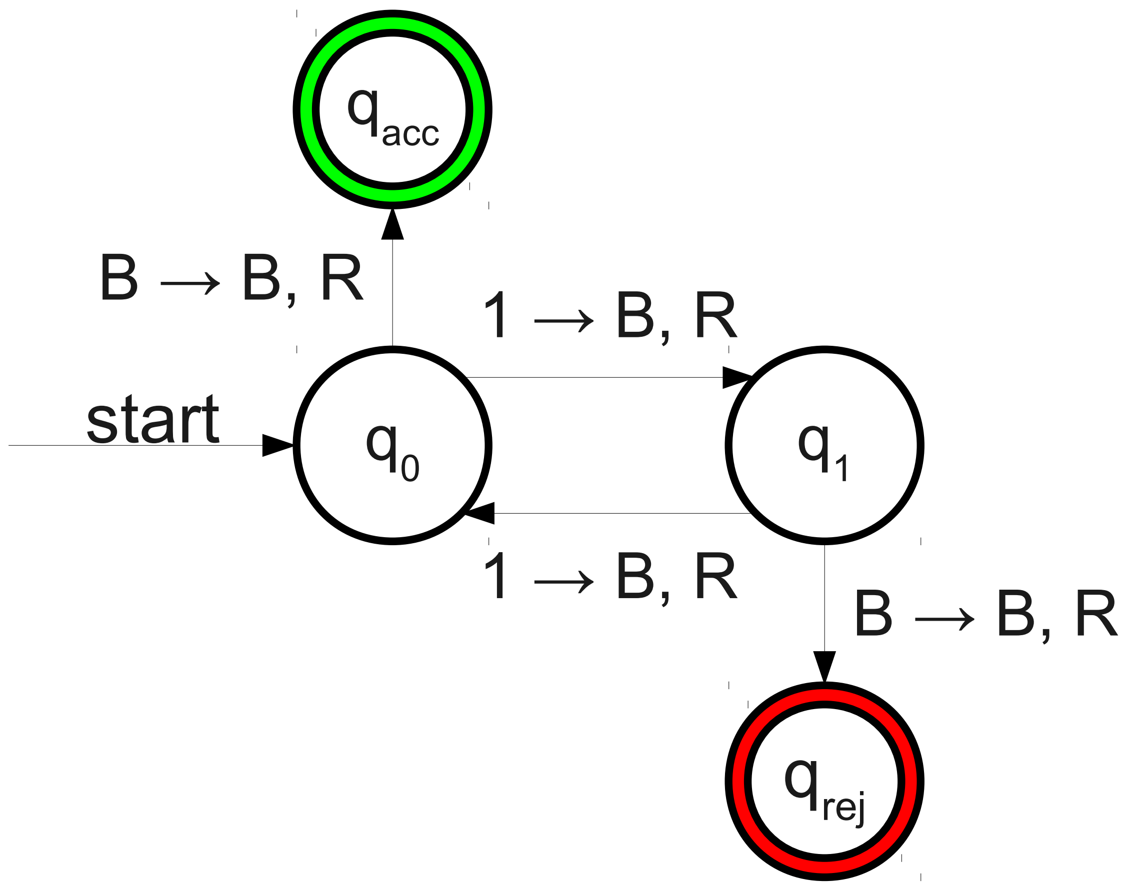


The Turing Machine

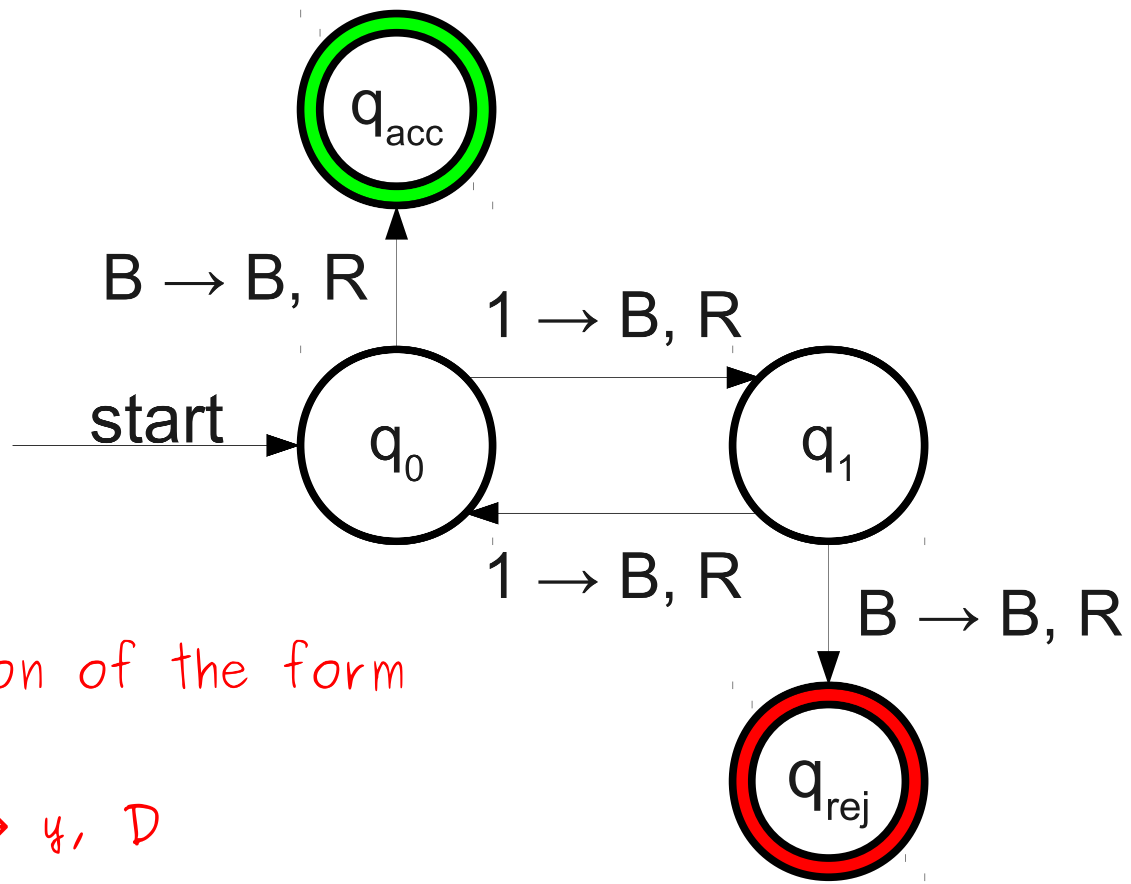
- A Turing machine consists of three parts:
  - A **finite-state control** used to determine which actions to take,
  - an **infinite tape** serving as both input and scratch space, and
  - a **tape head** that can read and write the tape and move left or right.
- At each step, the Turing machine
  - 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.
- The Turing machine **accepts** if it enters a special **accept state**. It **rejects** if it enters a special **reject state**.

A Simple Turing Machine

A Simple Turing Machine



A Simple Turing Machine



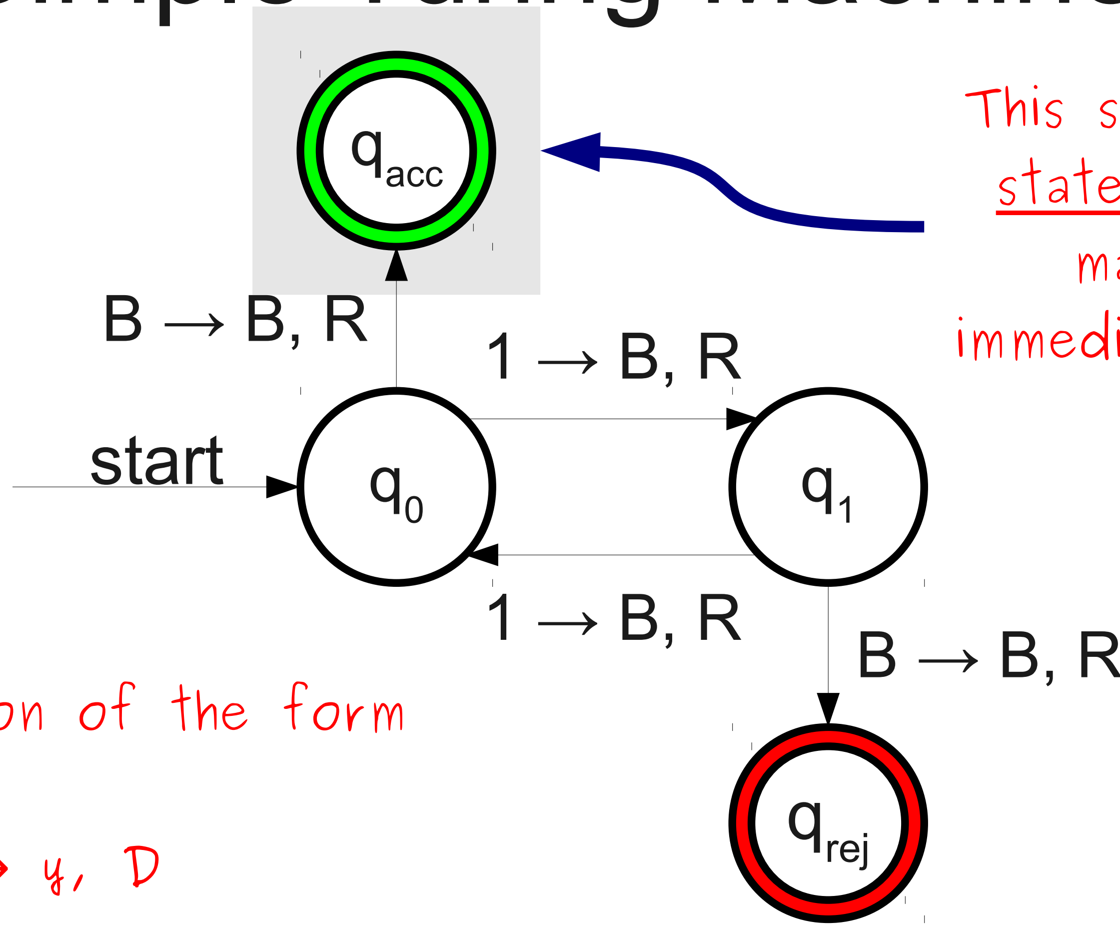
Each transition of the form

x → 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 a blank.

A Simple Turing Machine



This special accept state causes the machine to immediately accept.

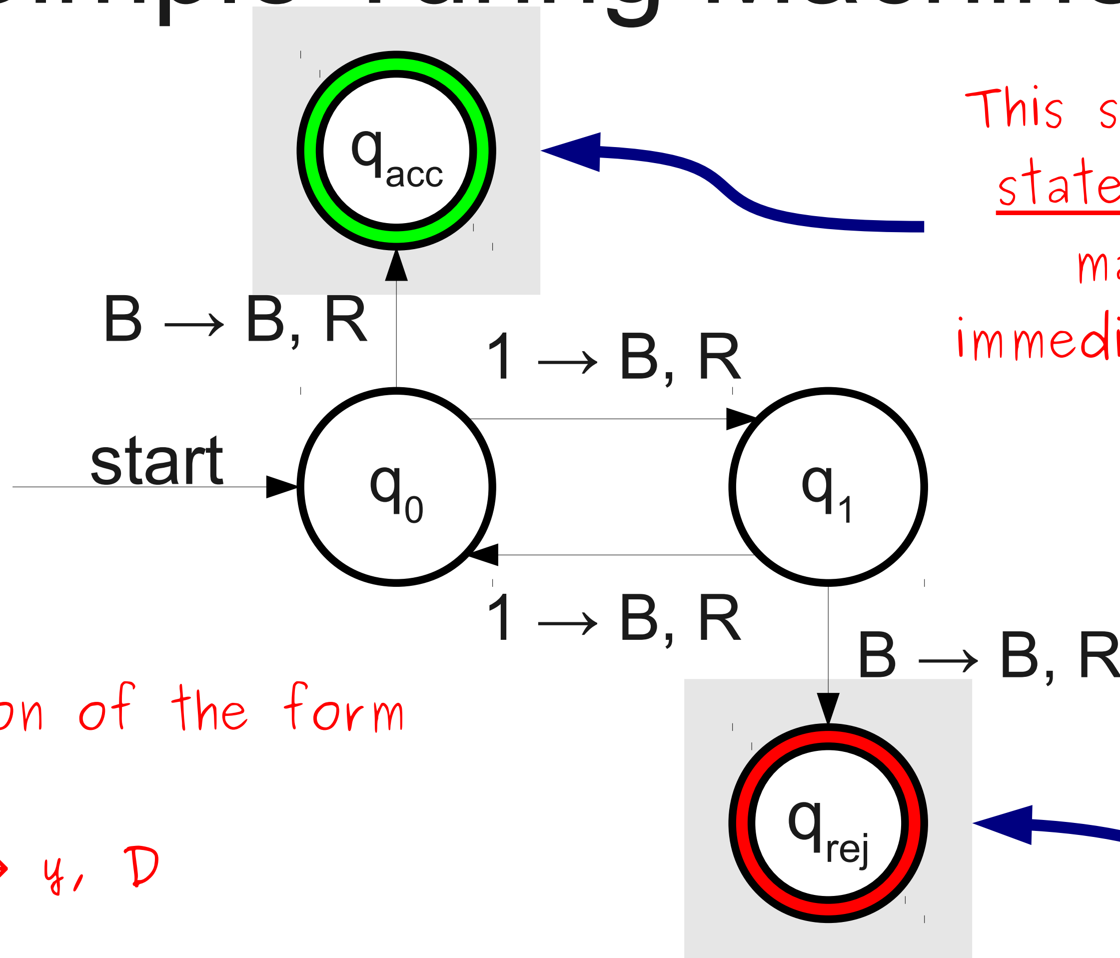
Each transition of the form

x → 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 a blank.

A Simple Turing Machine



This special <u>accept state</u> causes the machine to immediately accept.

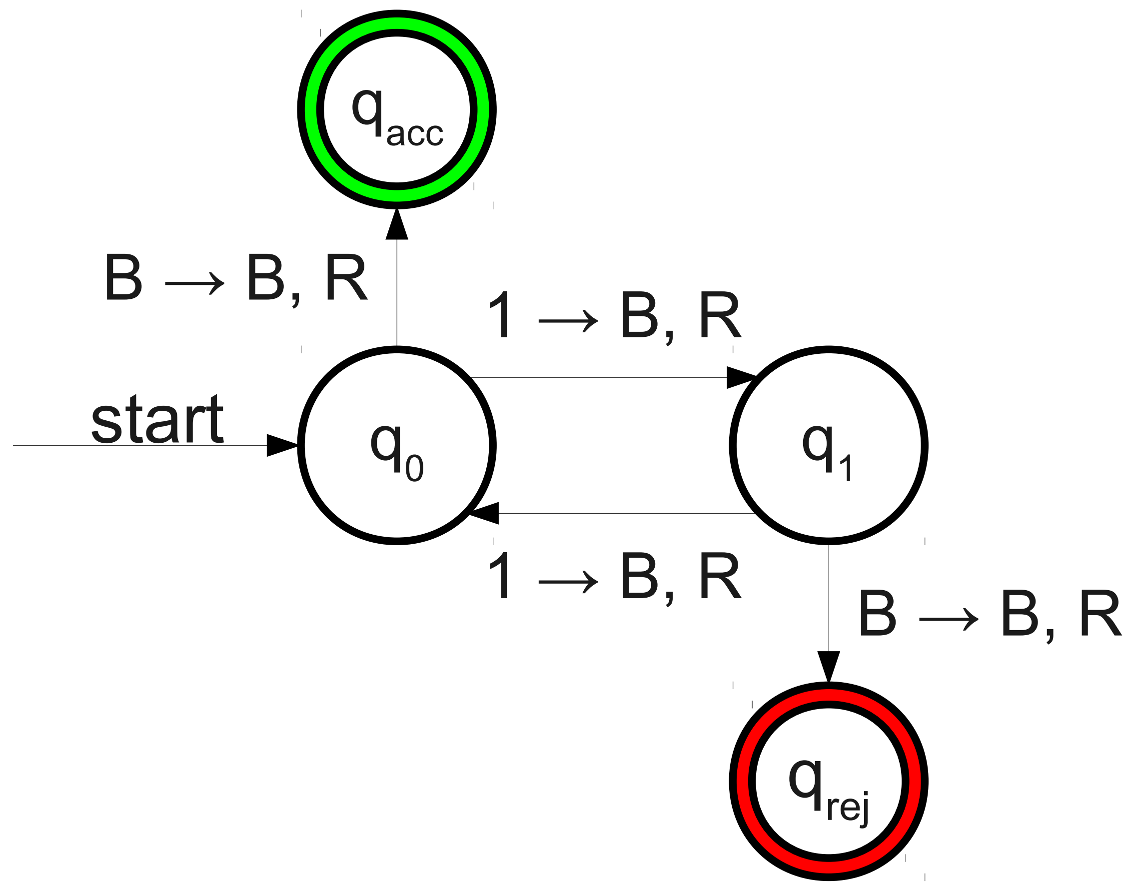
Each transition of the form

x → y, D

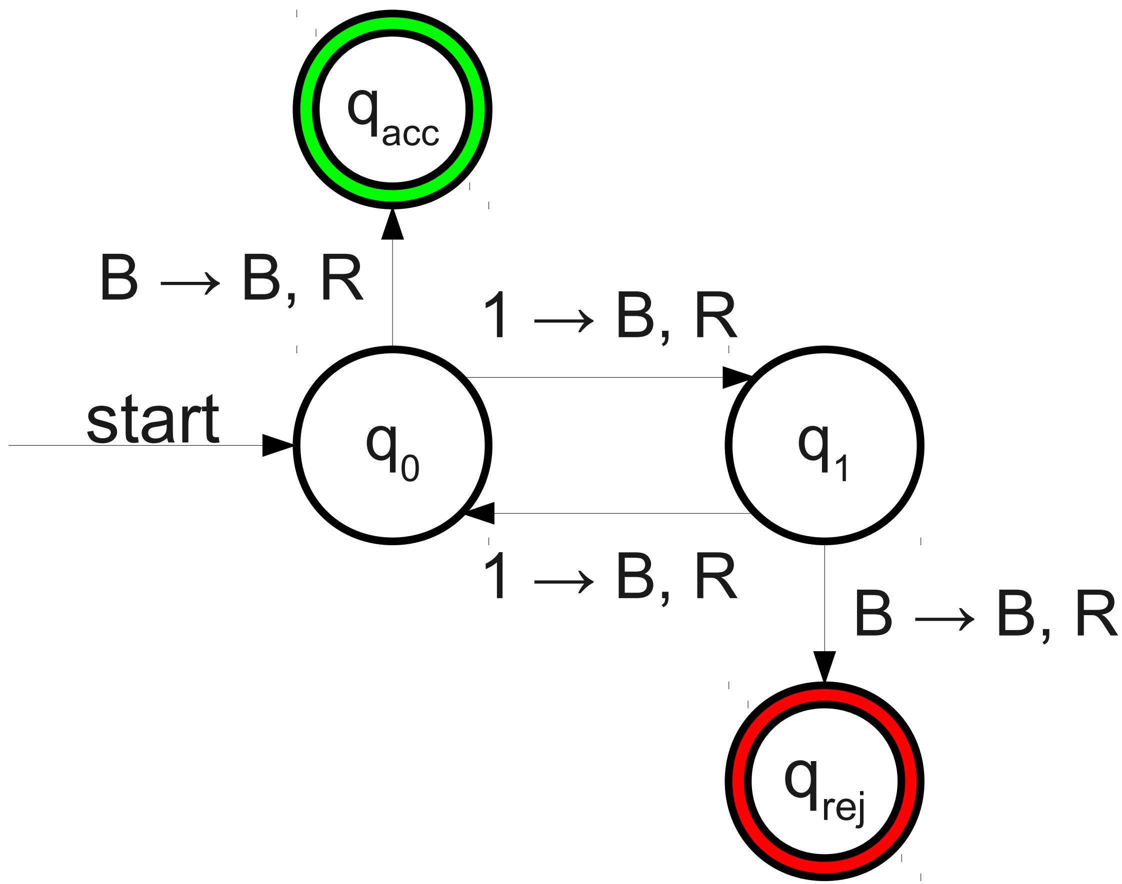
means "upon reading x, replace it with symbol y and move the tape head in direction D (which is either L or R). The letter B represents a blank.

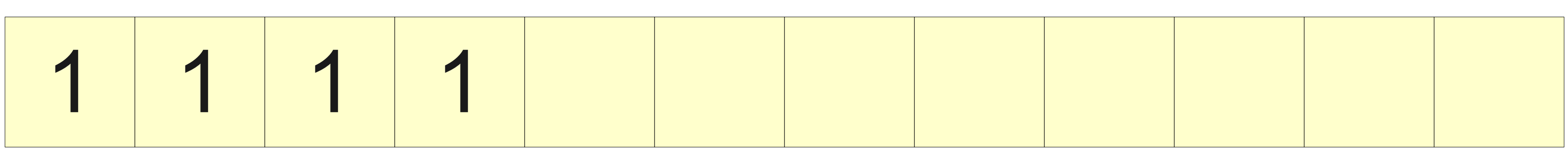
This special <u>reject state</u> causes the machine to immediately reject.

A Simple Turing Machine

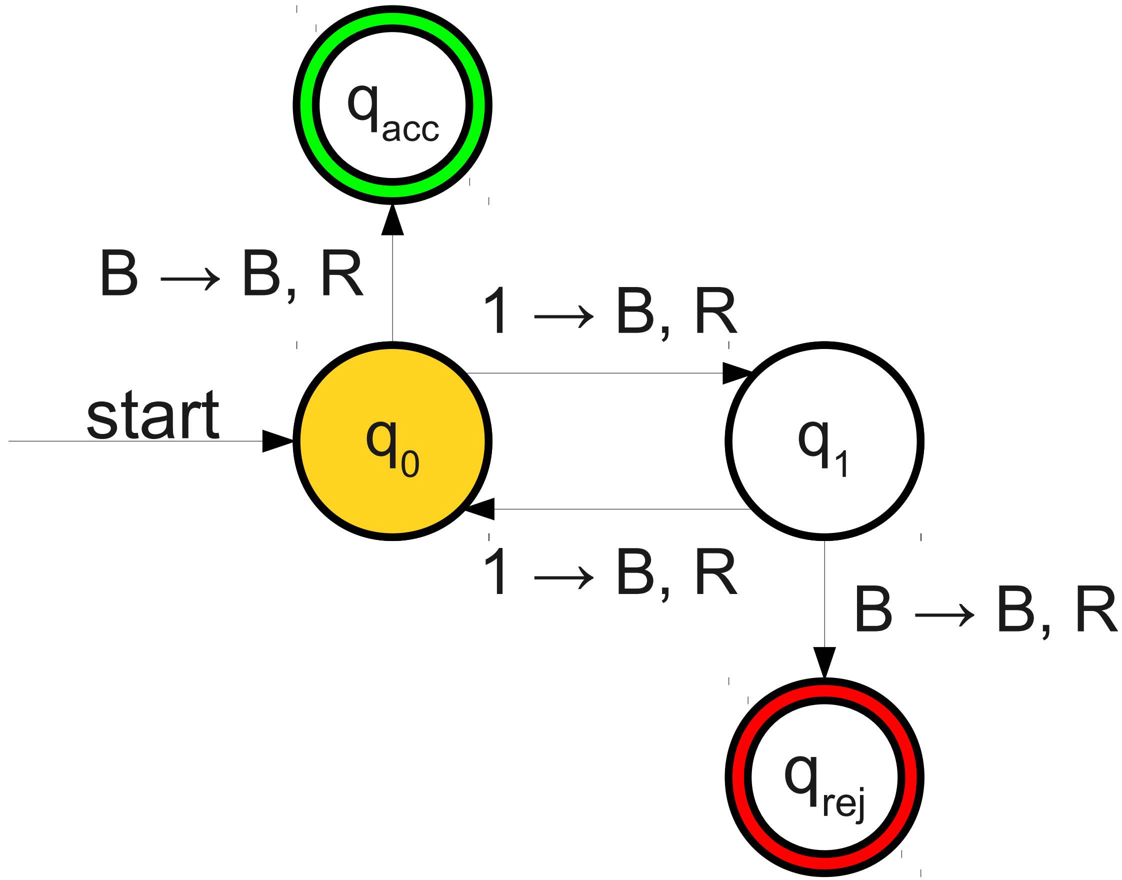


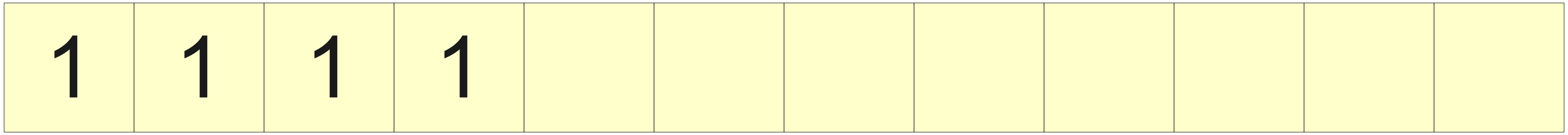
A Simple Turing Machine



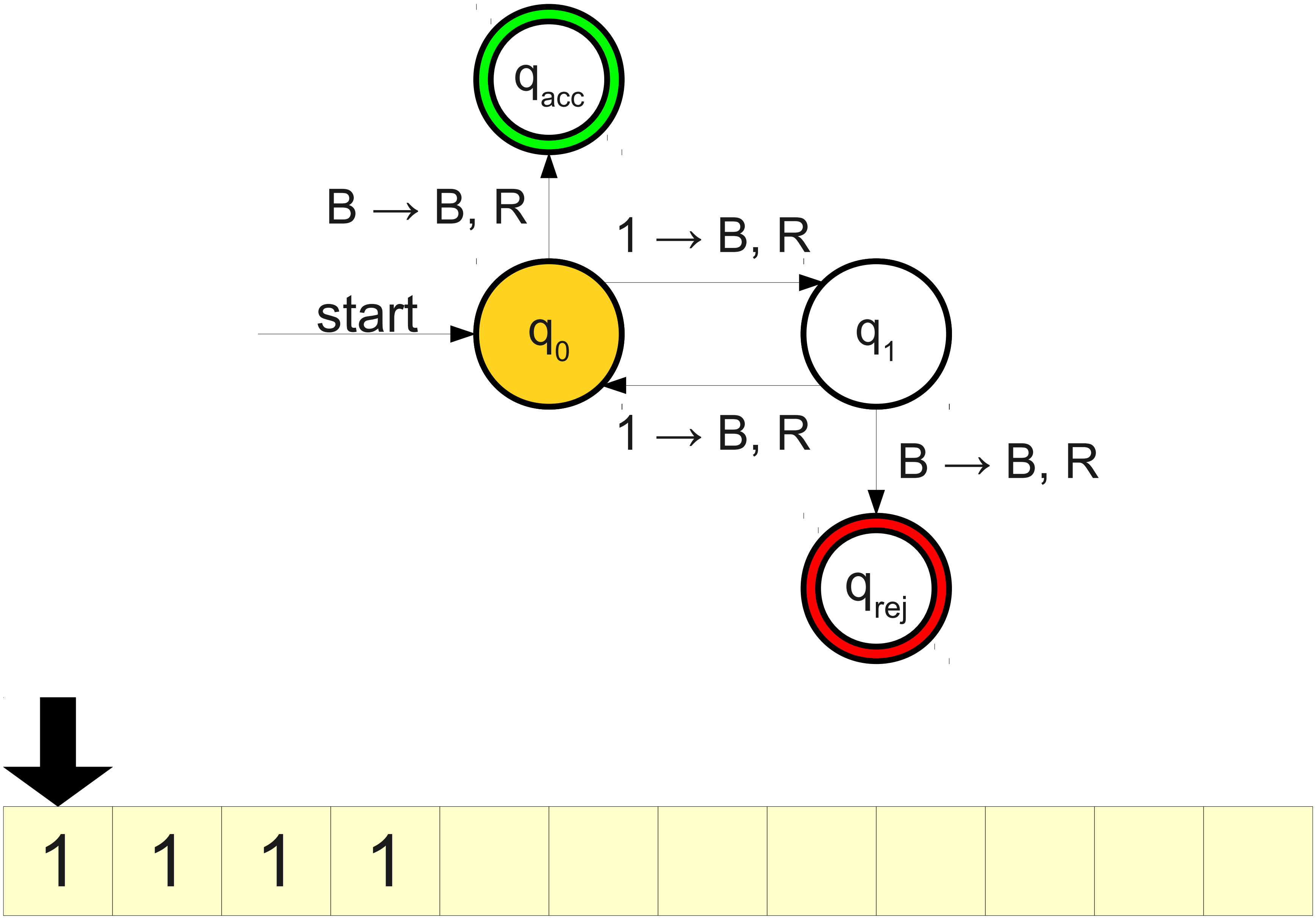


A Simple Turing Machine

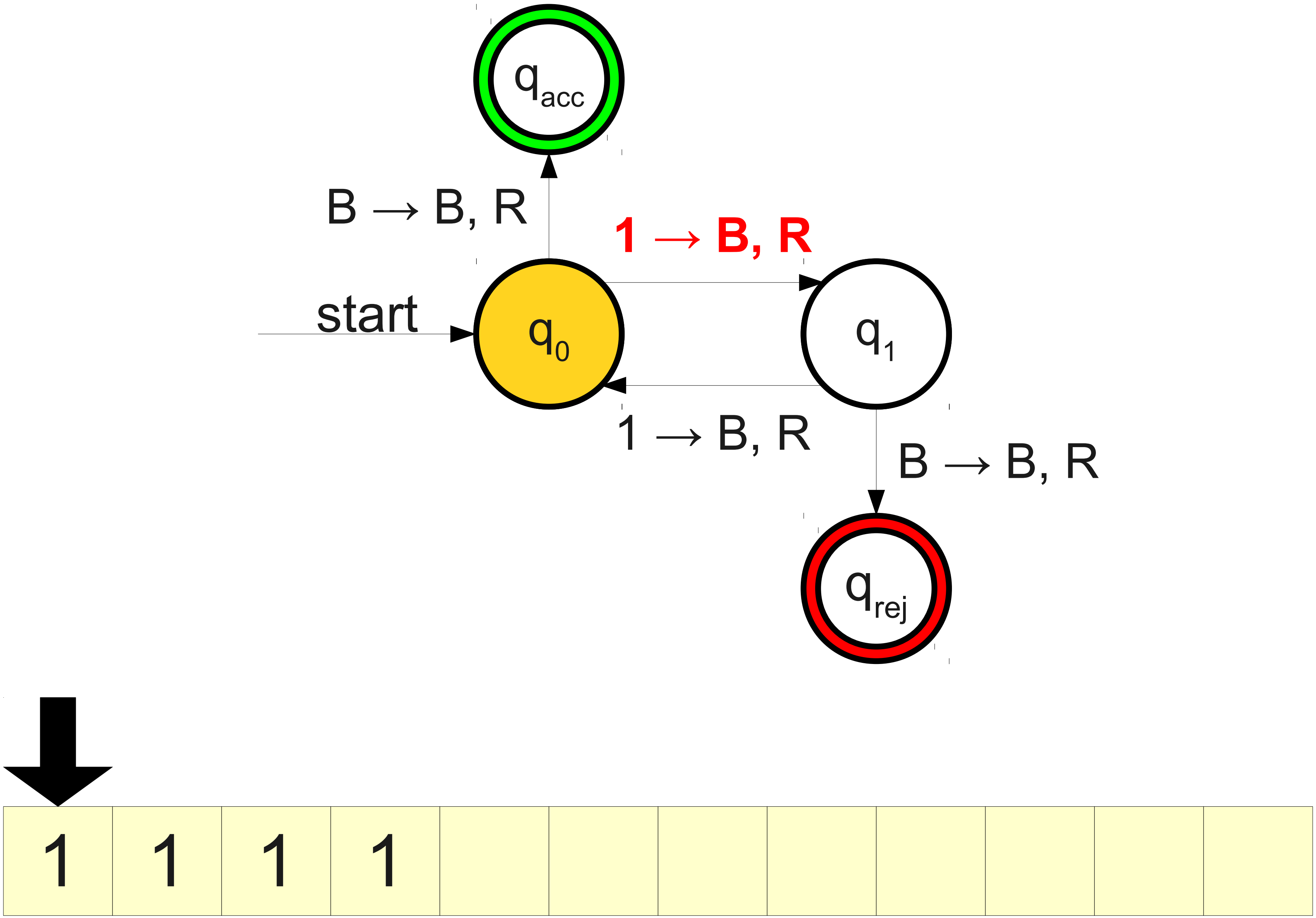




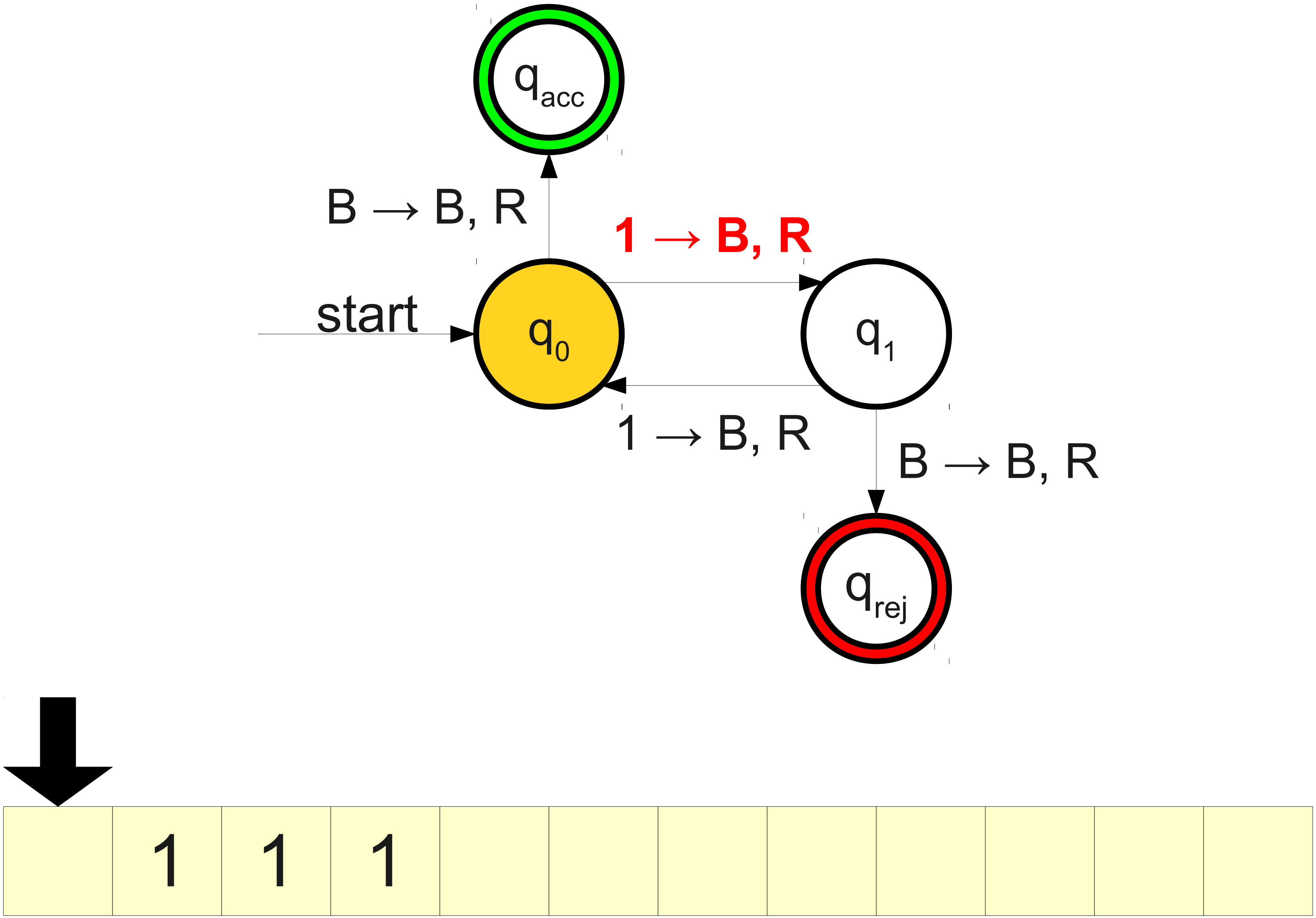
A Simple Turing Machine



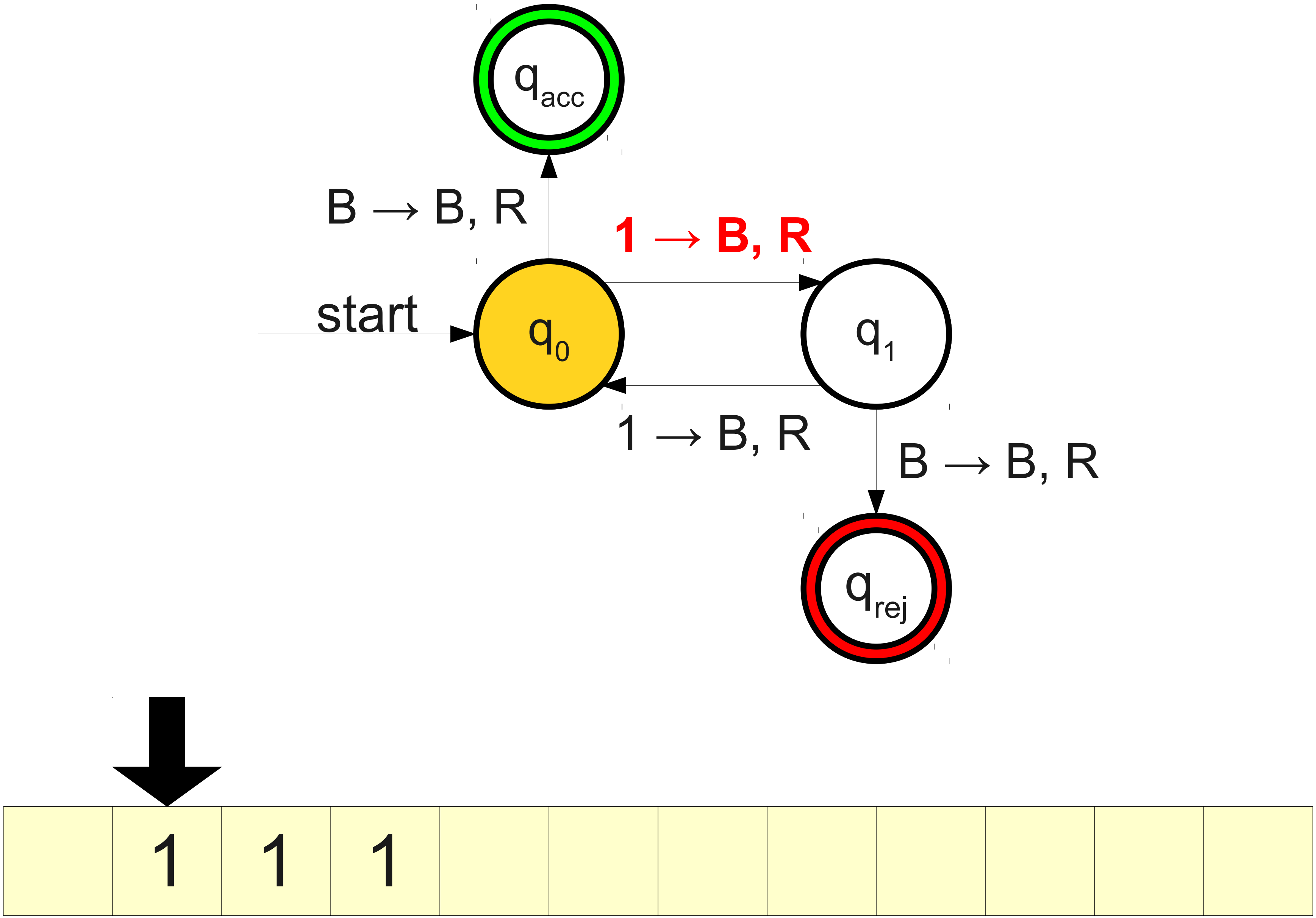
A Simple Turing Machine



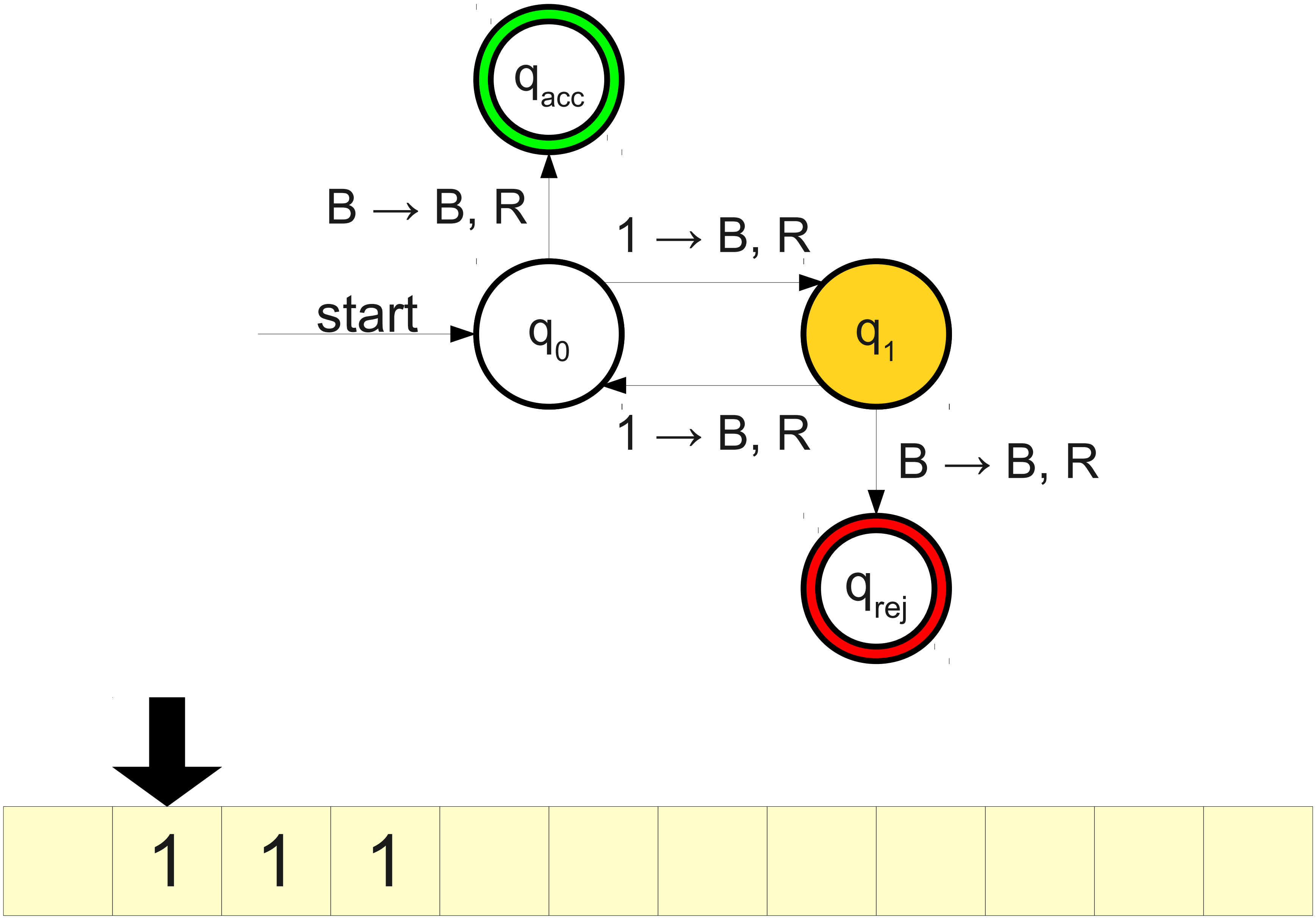
A Simple Turing Machine



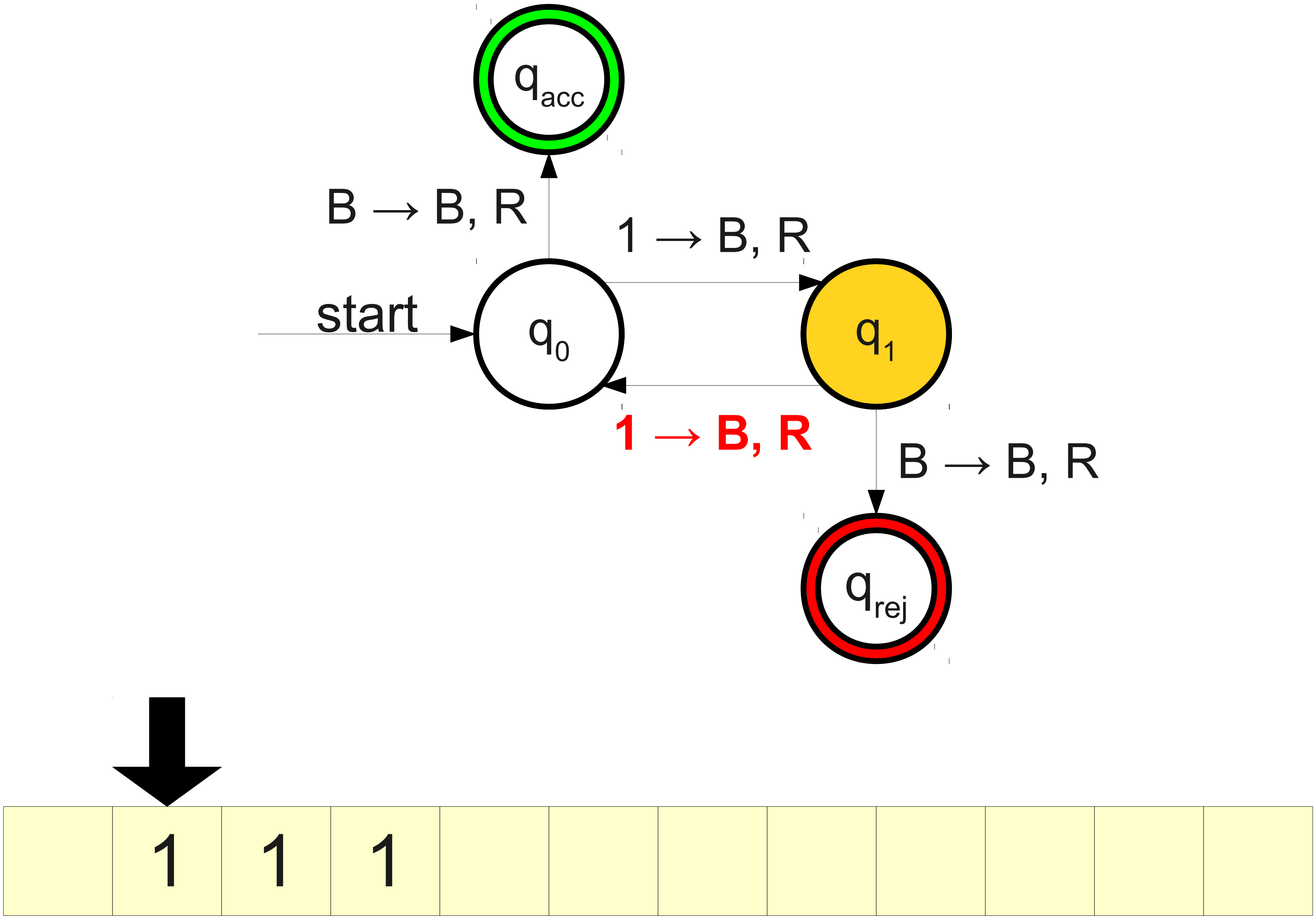
A Simple Turing Machine



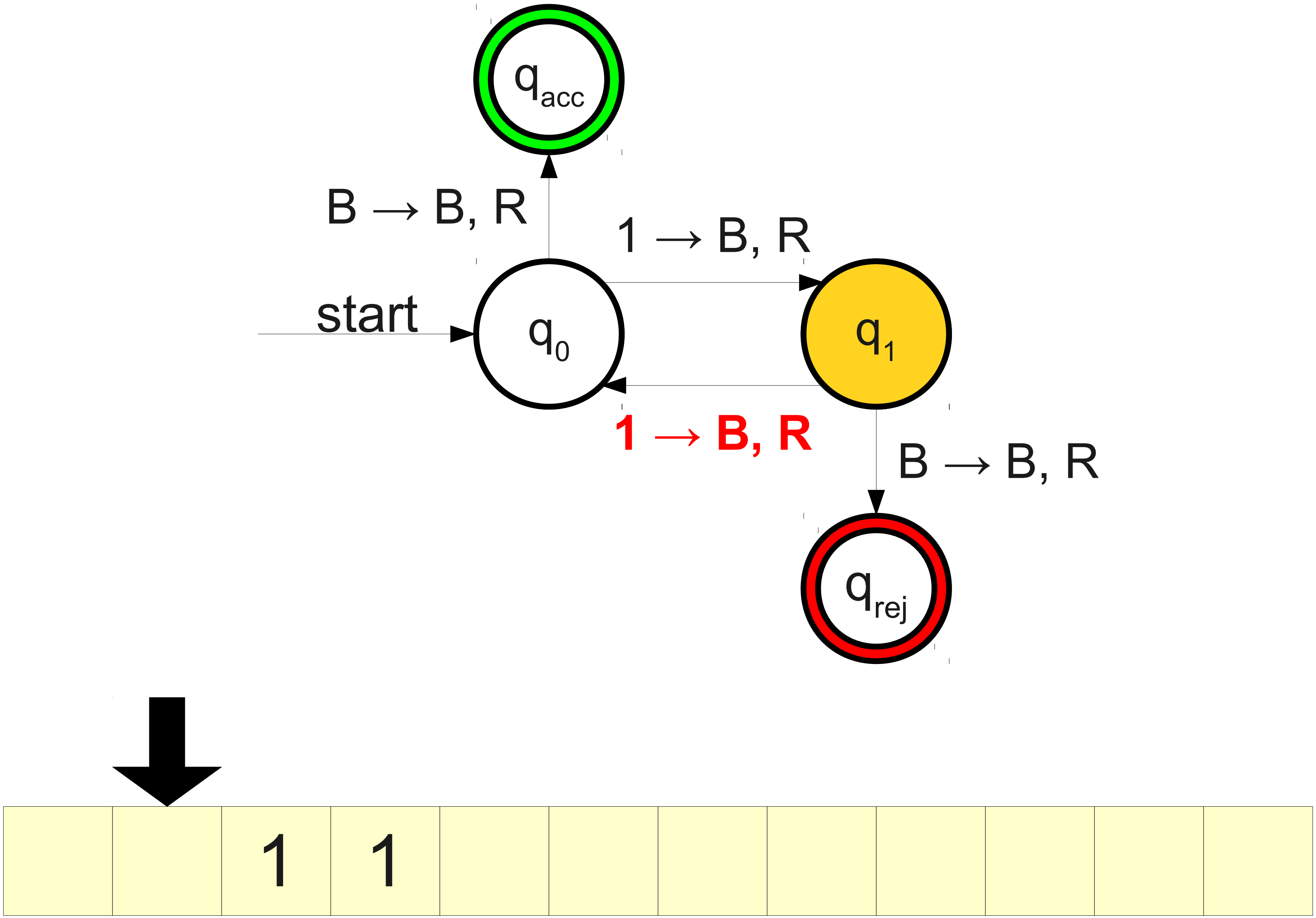
A Simple Turing Machine



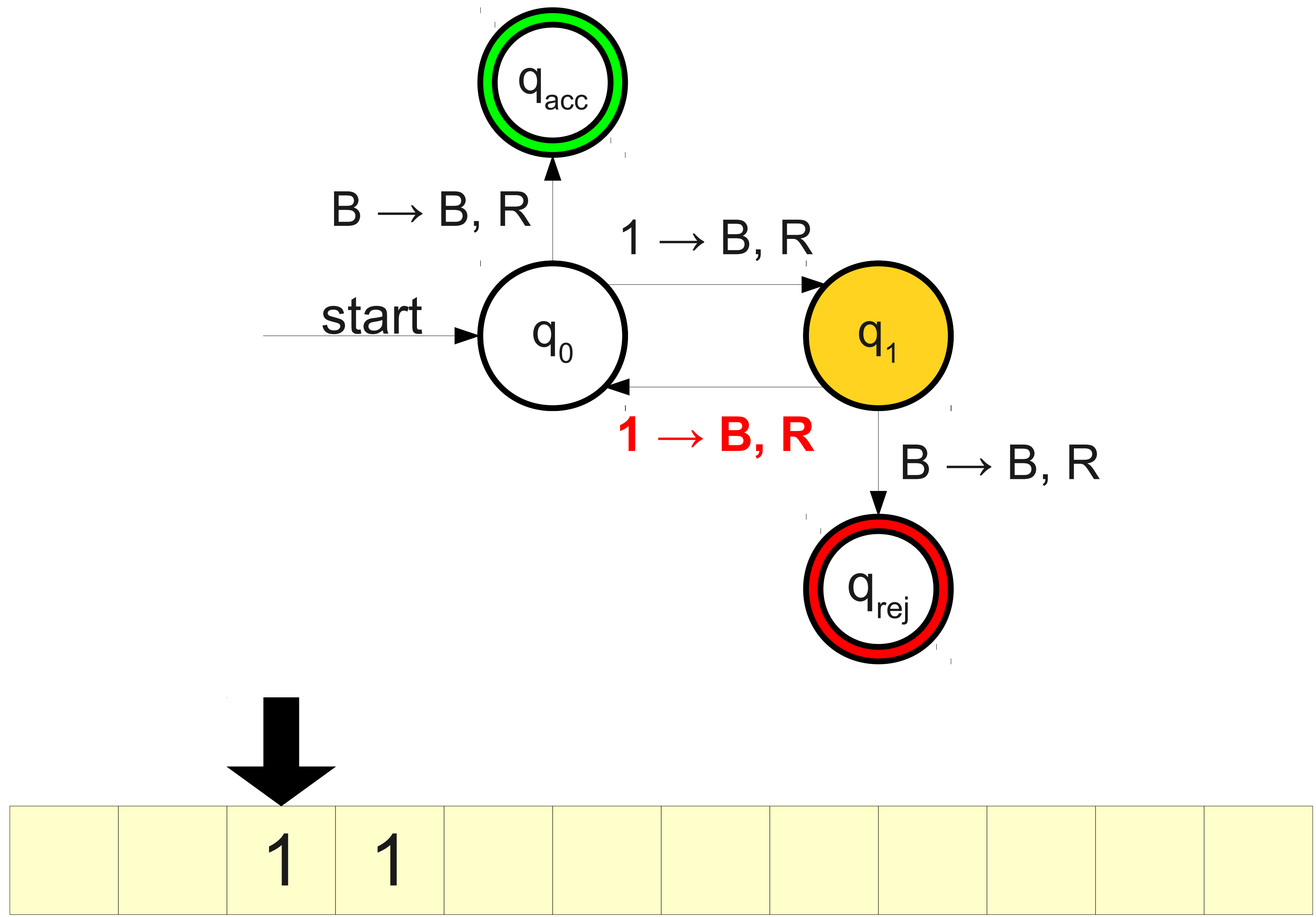
A Simple Turing Machine



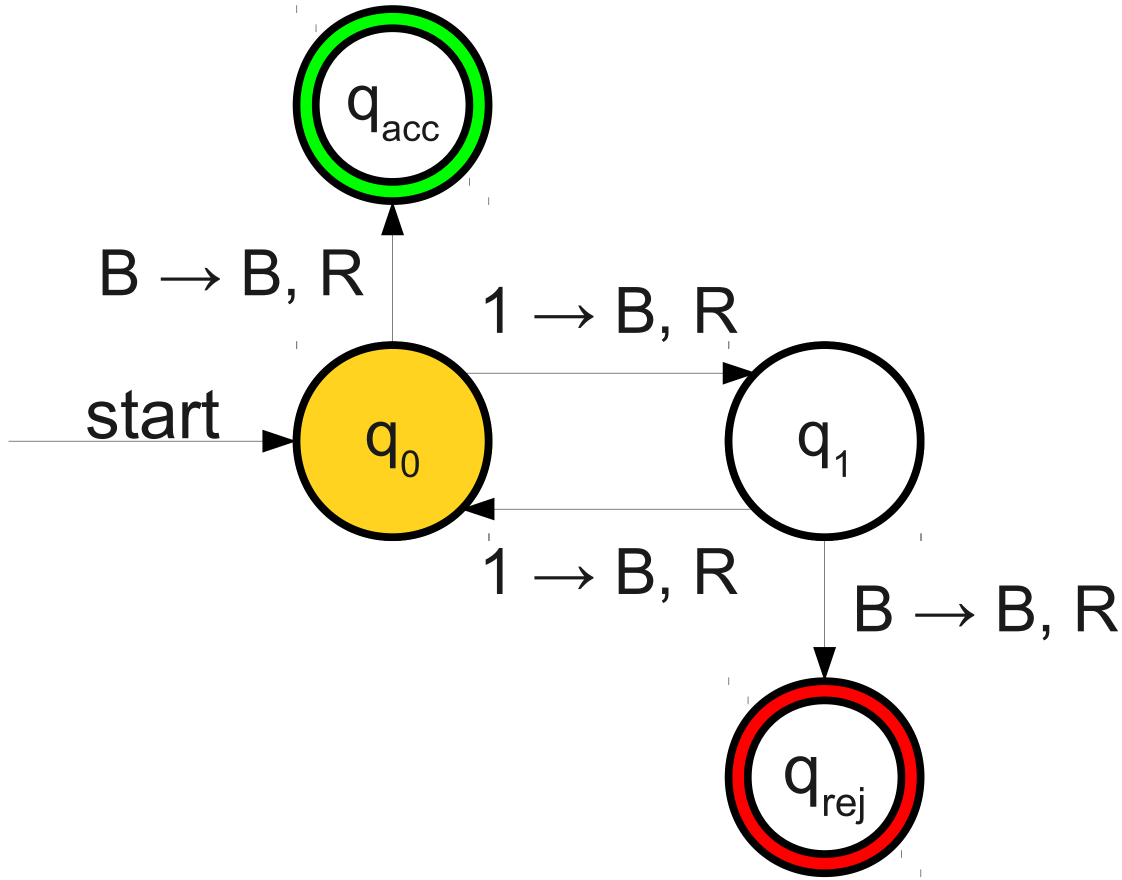
A Simple Turing Machine

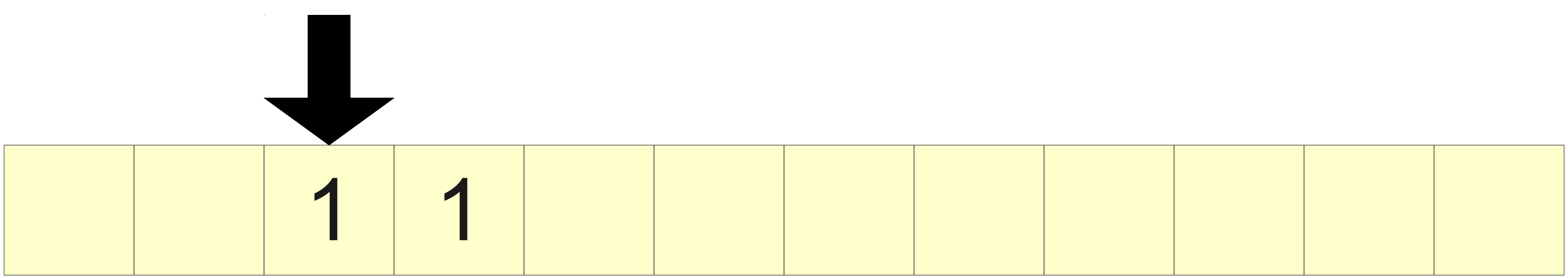


A Simple Turing Machine

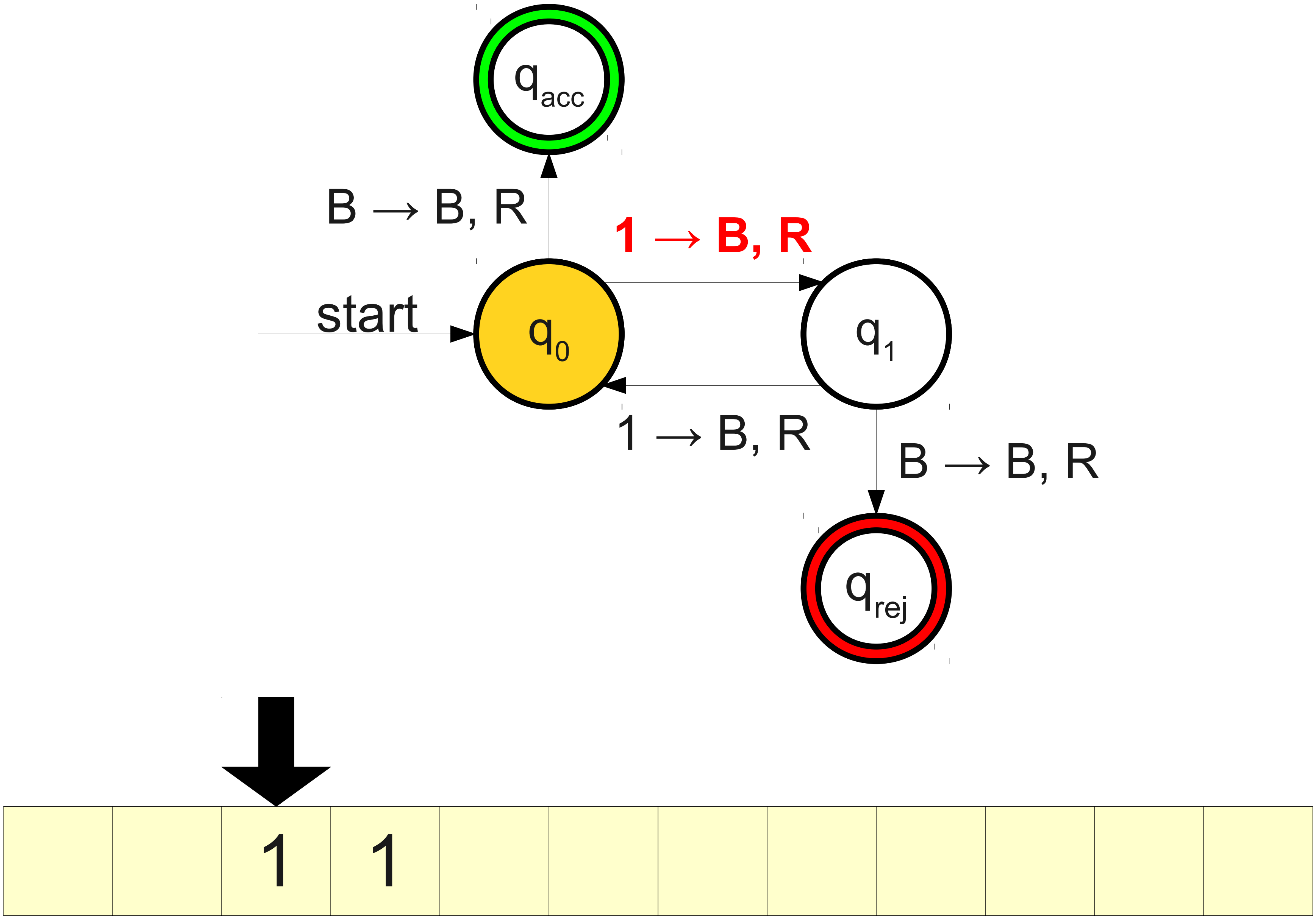


A Simple Turing Machine

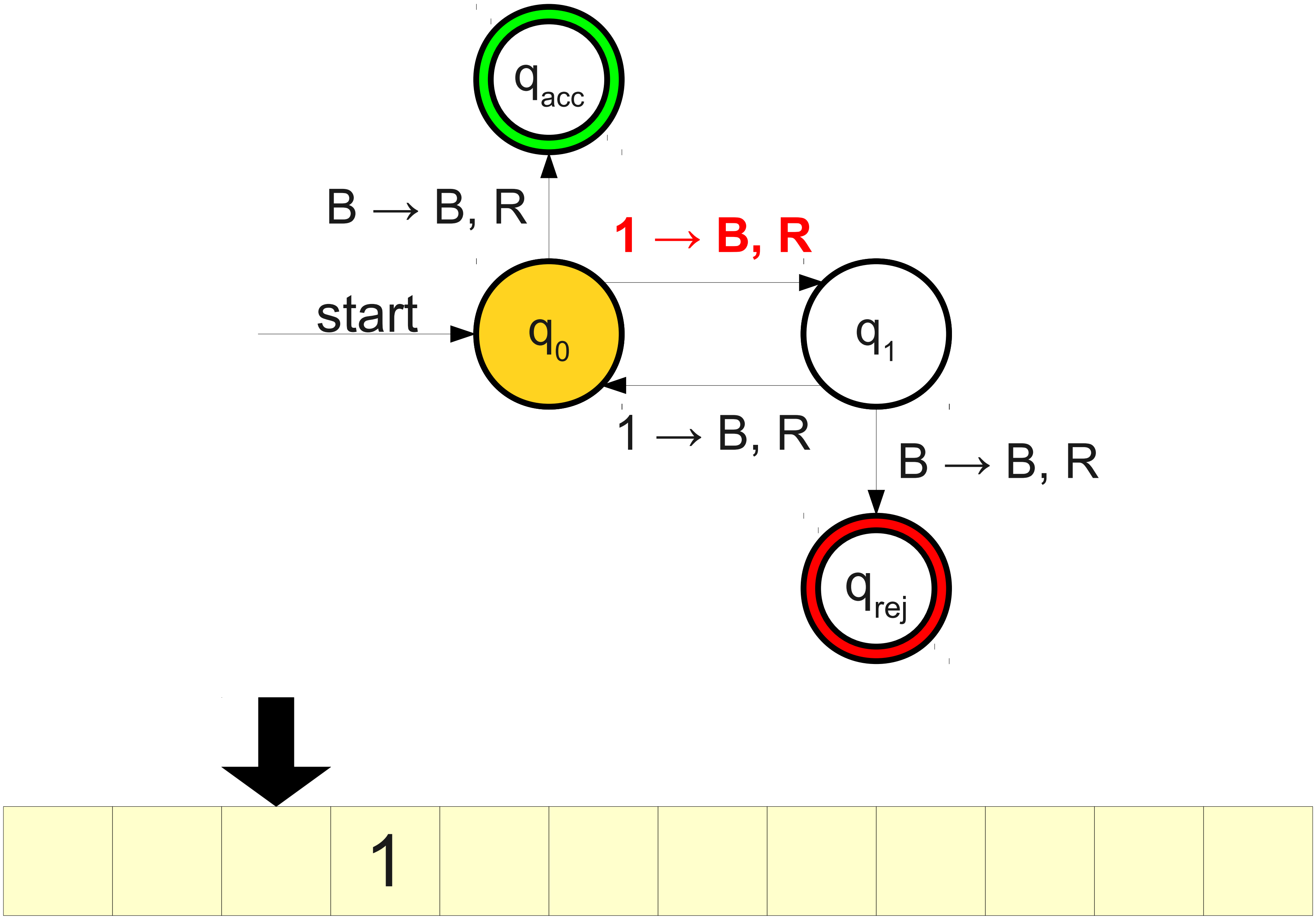




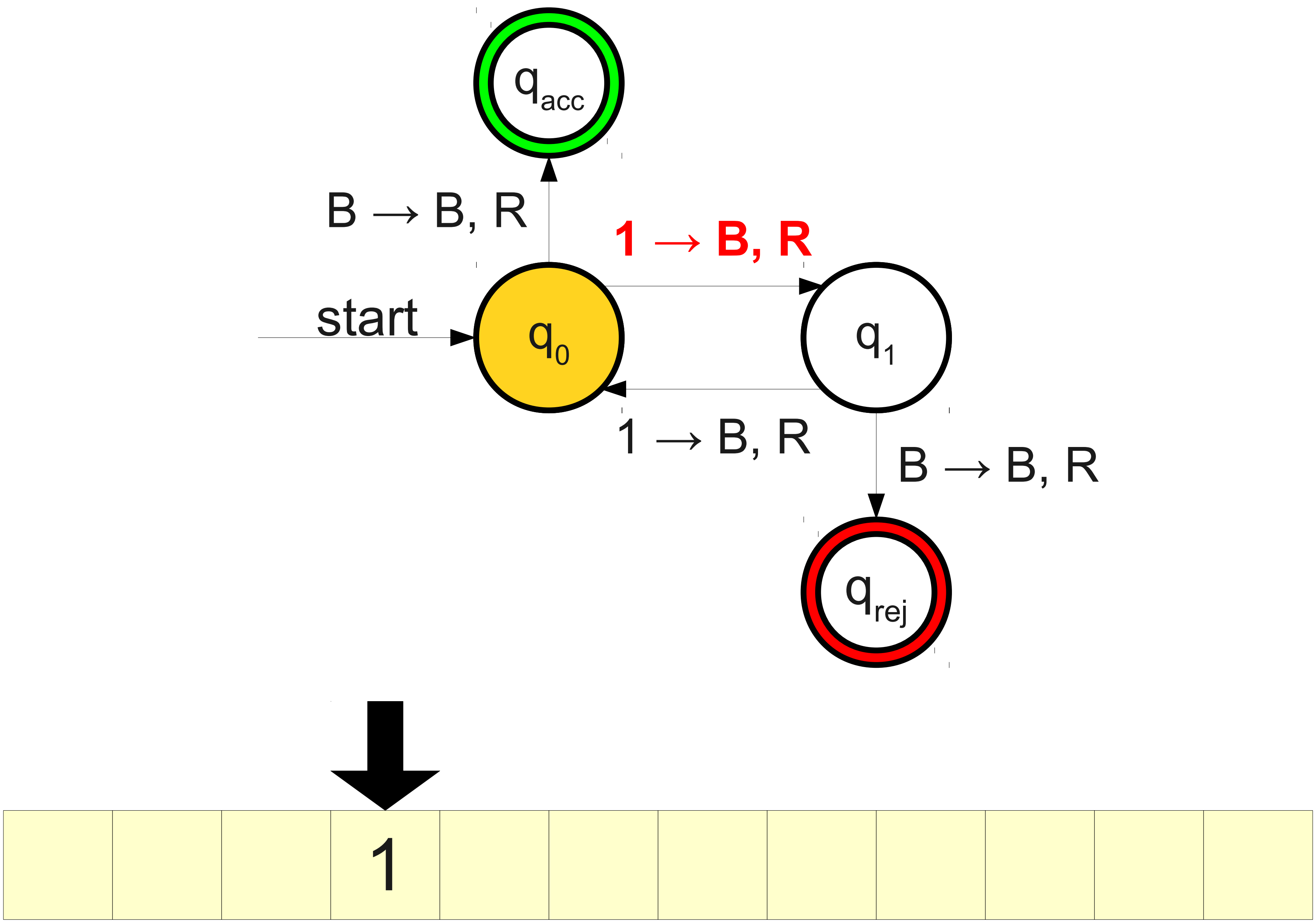
A Simple Turing Machine



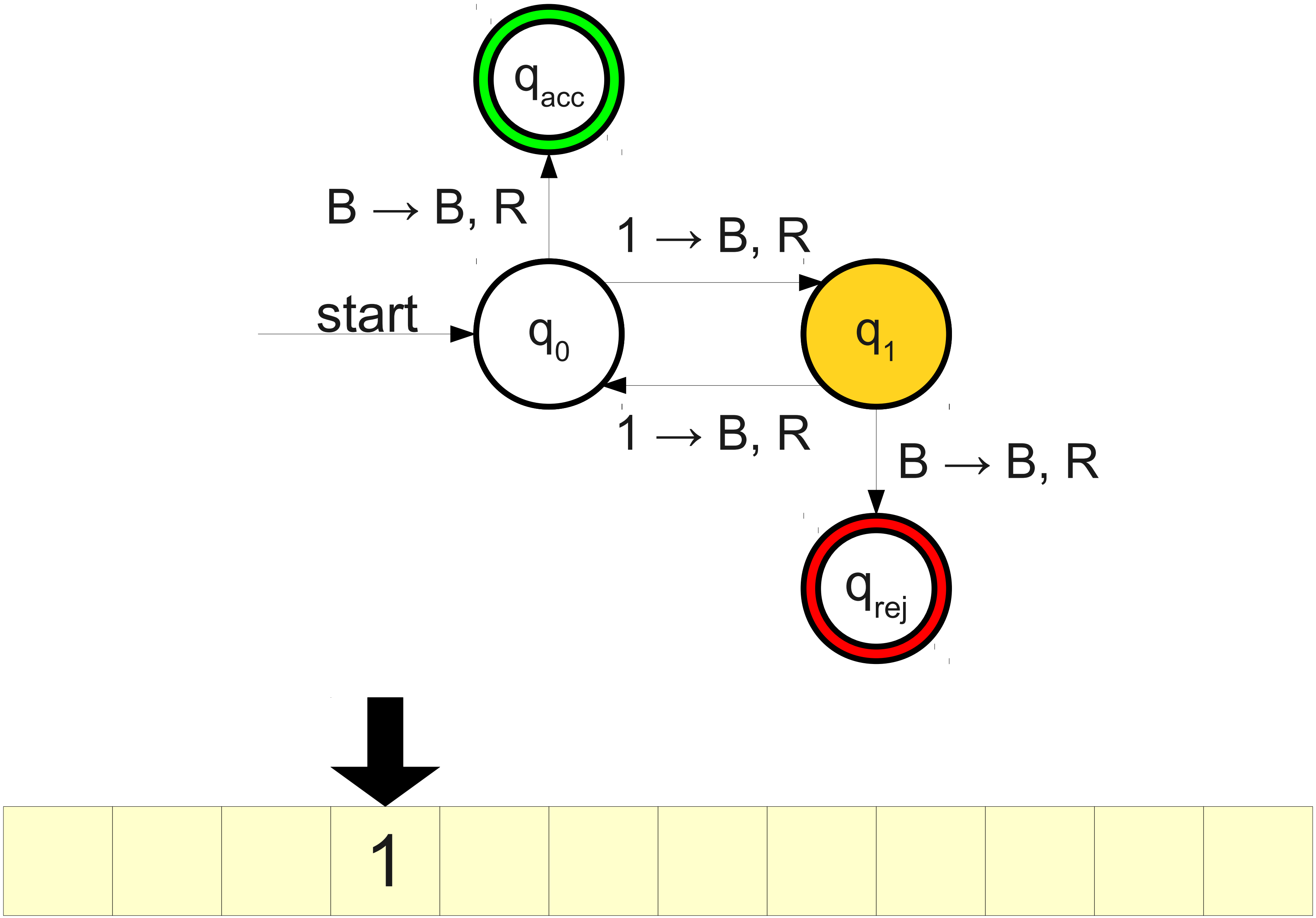
A Simple Turing Machine



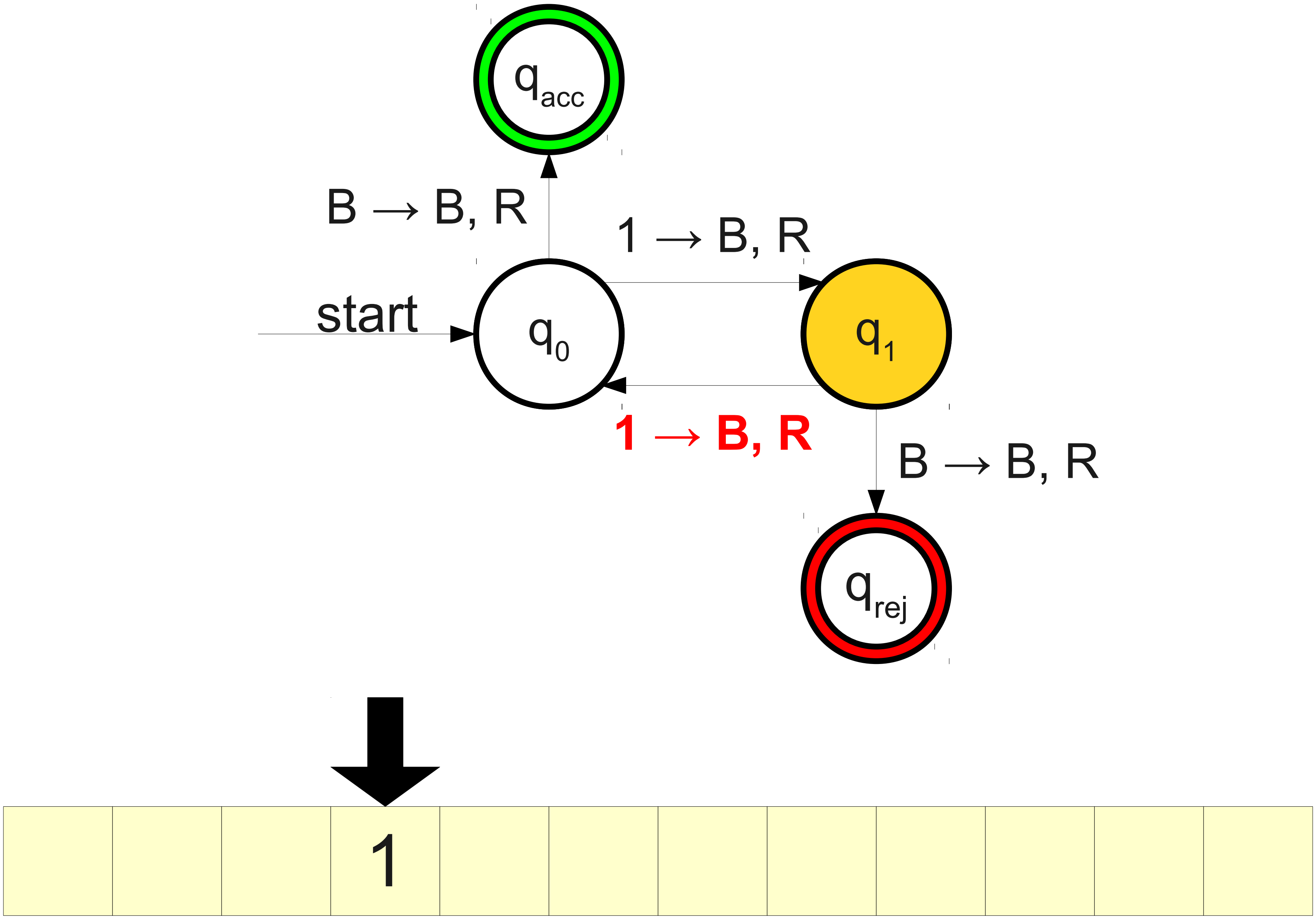
A Simple Turing Machine



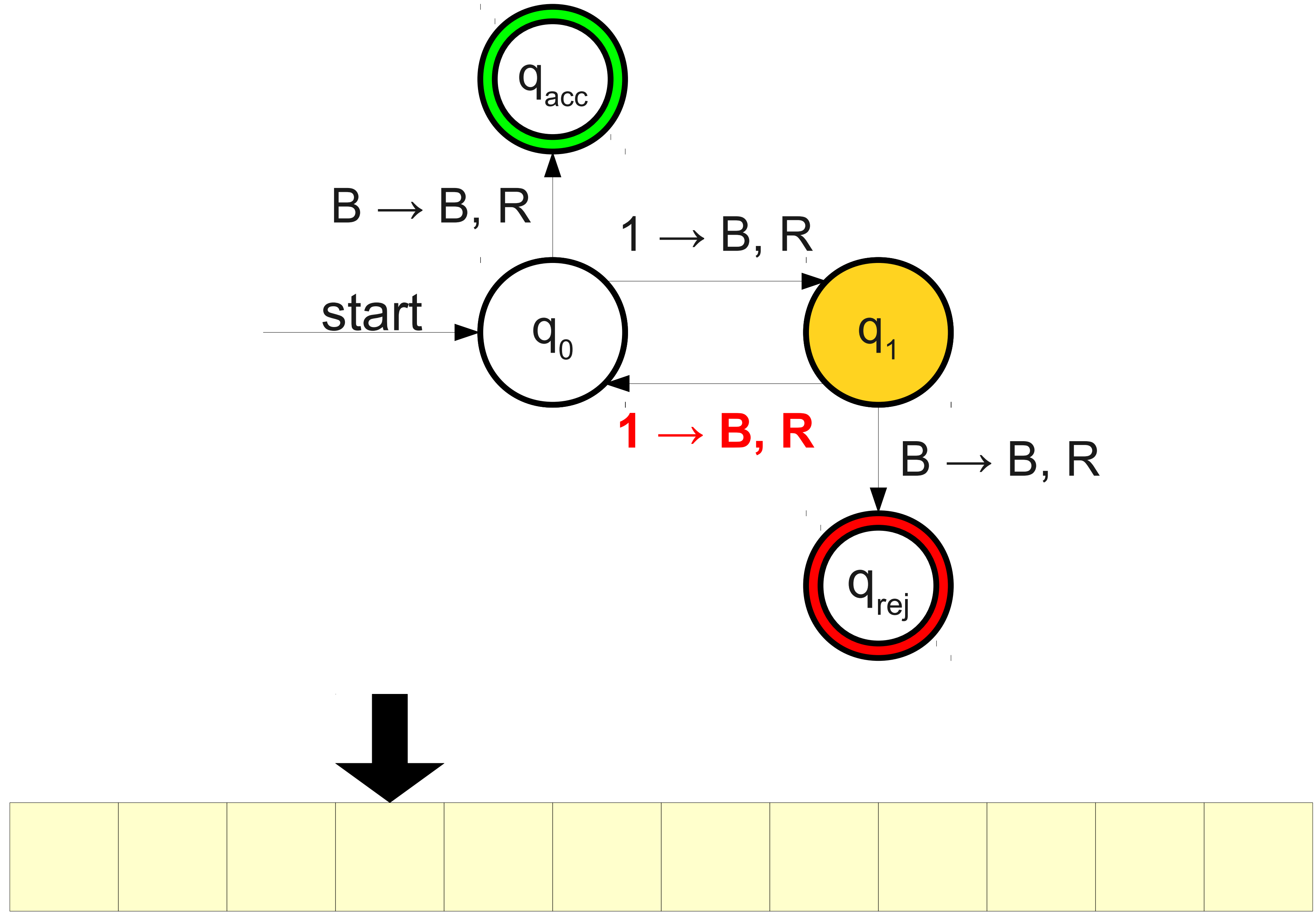
A Simple Turing Machine



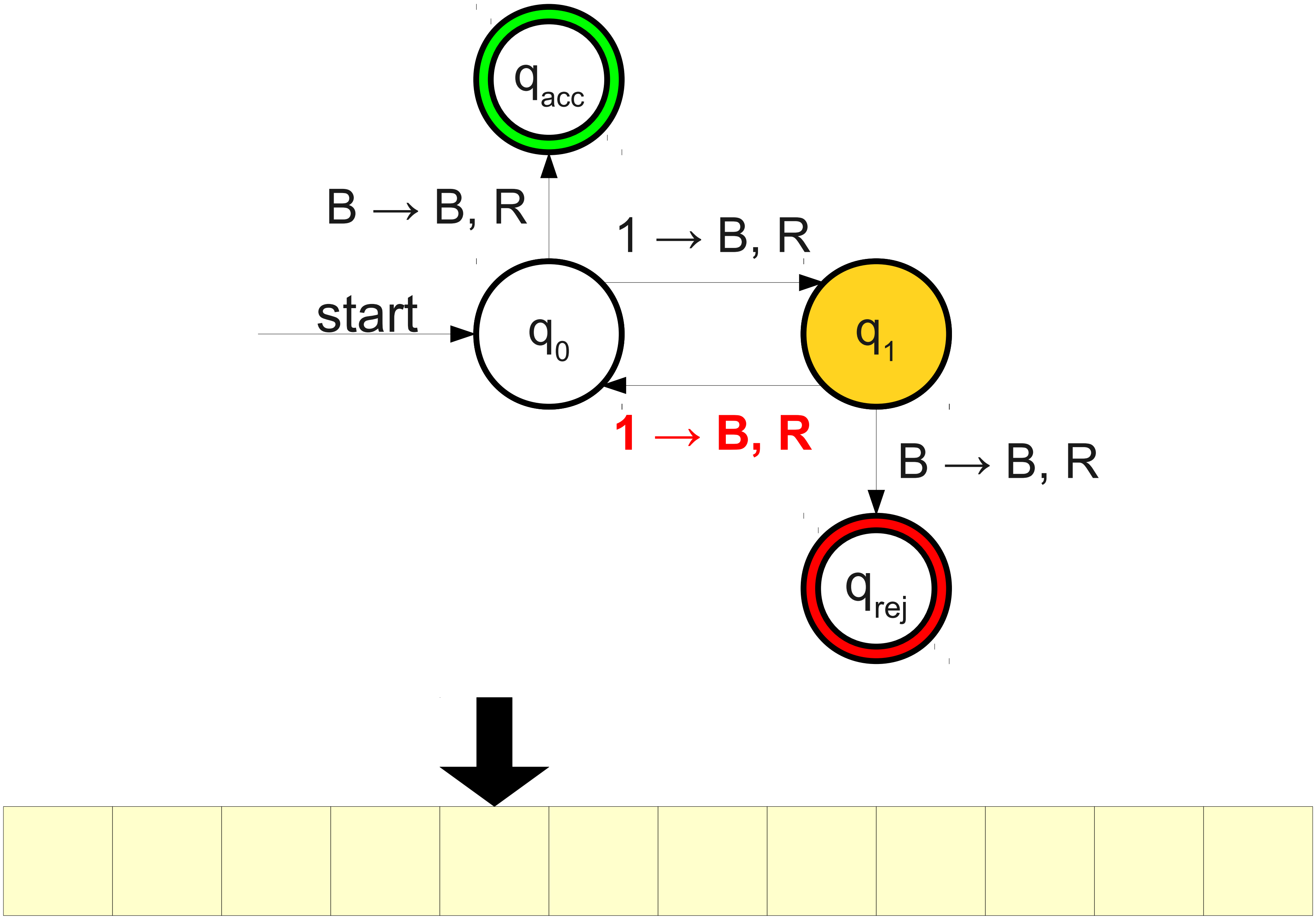
A Simple Turing Machine



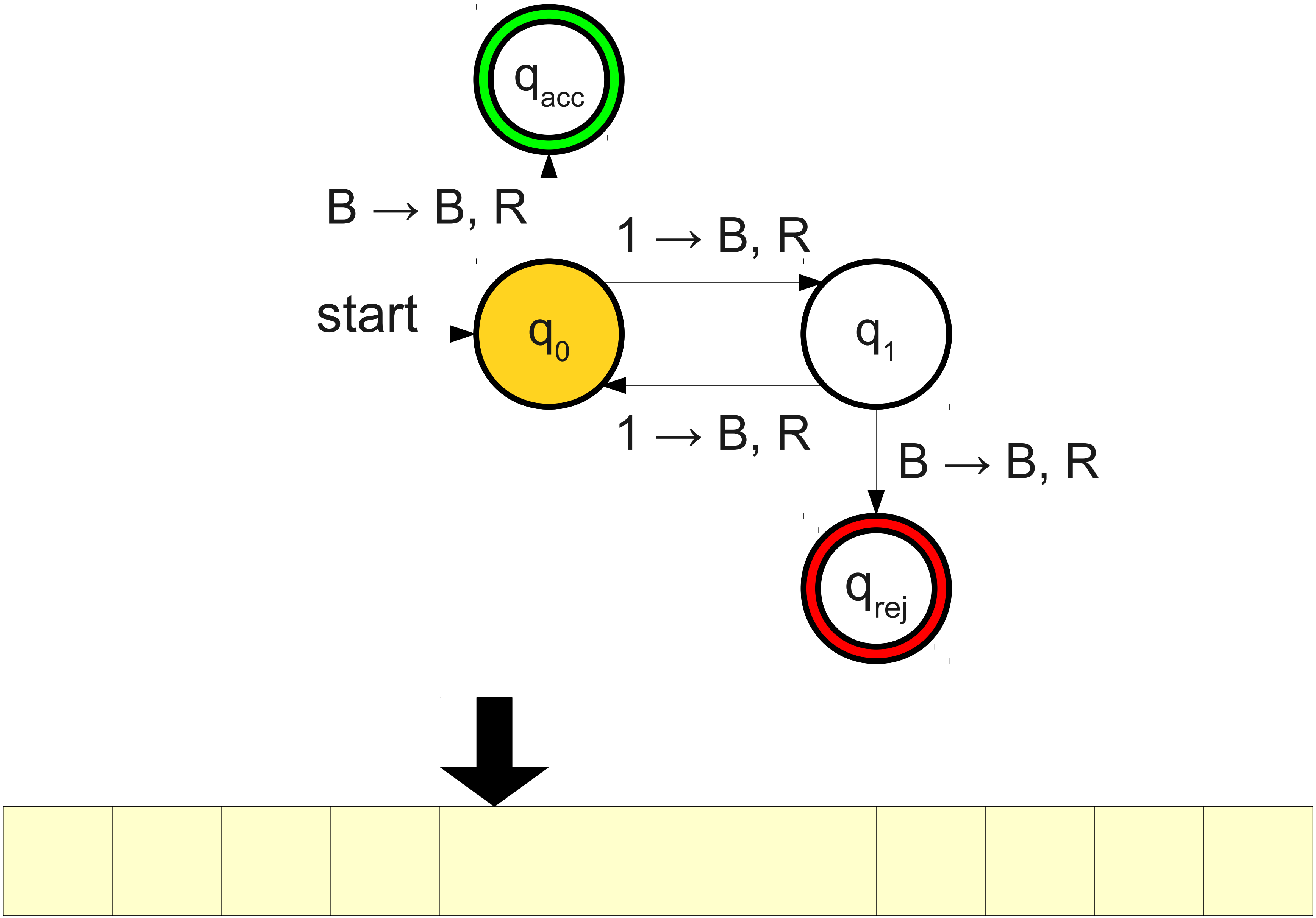
A Simple Turing Machine



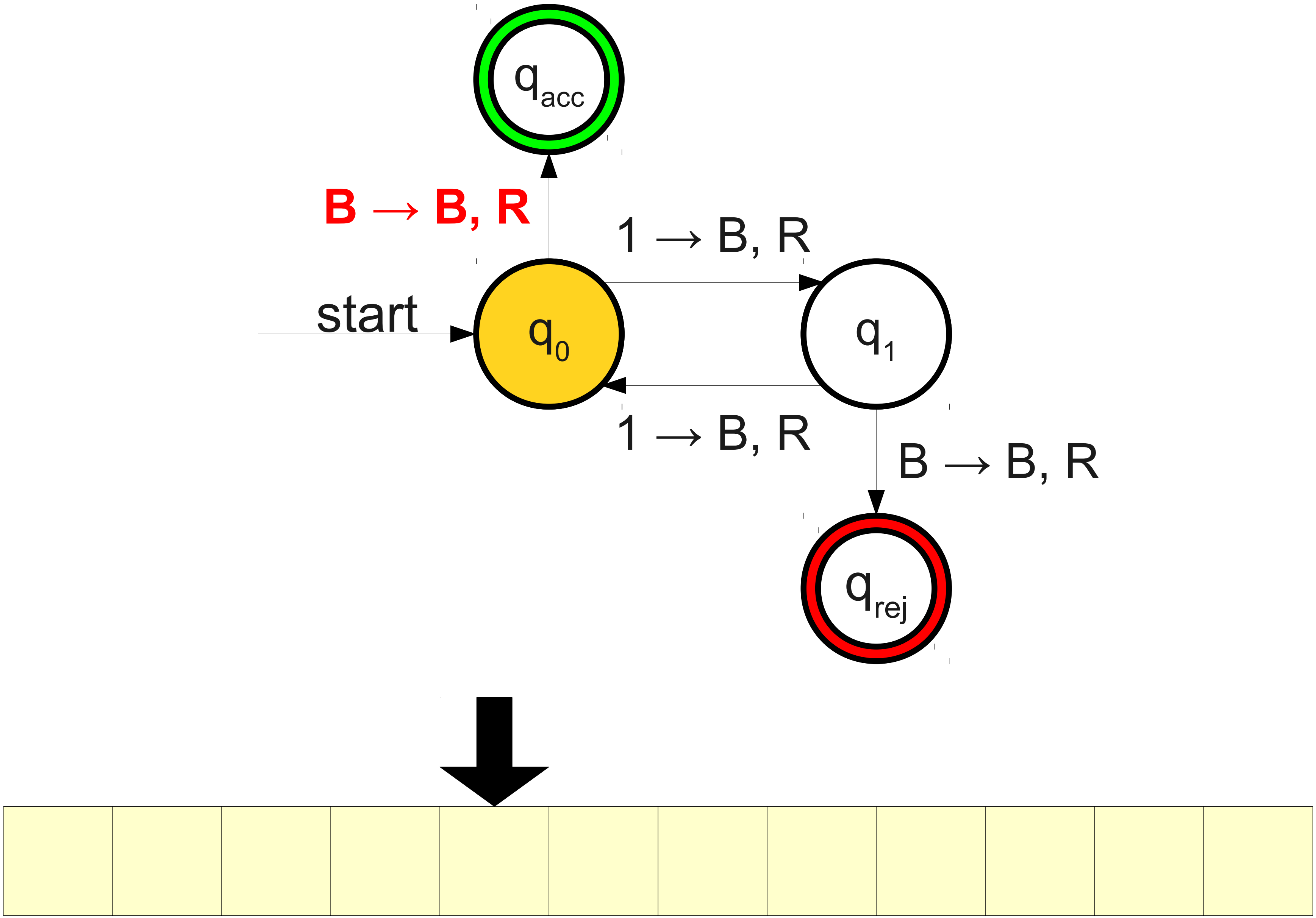
A Simple Turing Machine



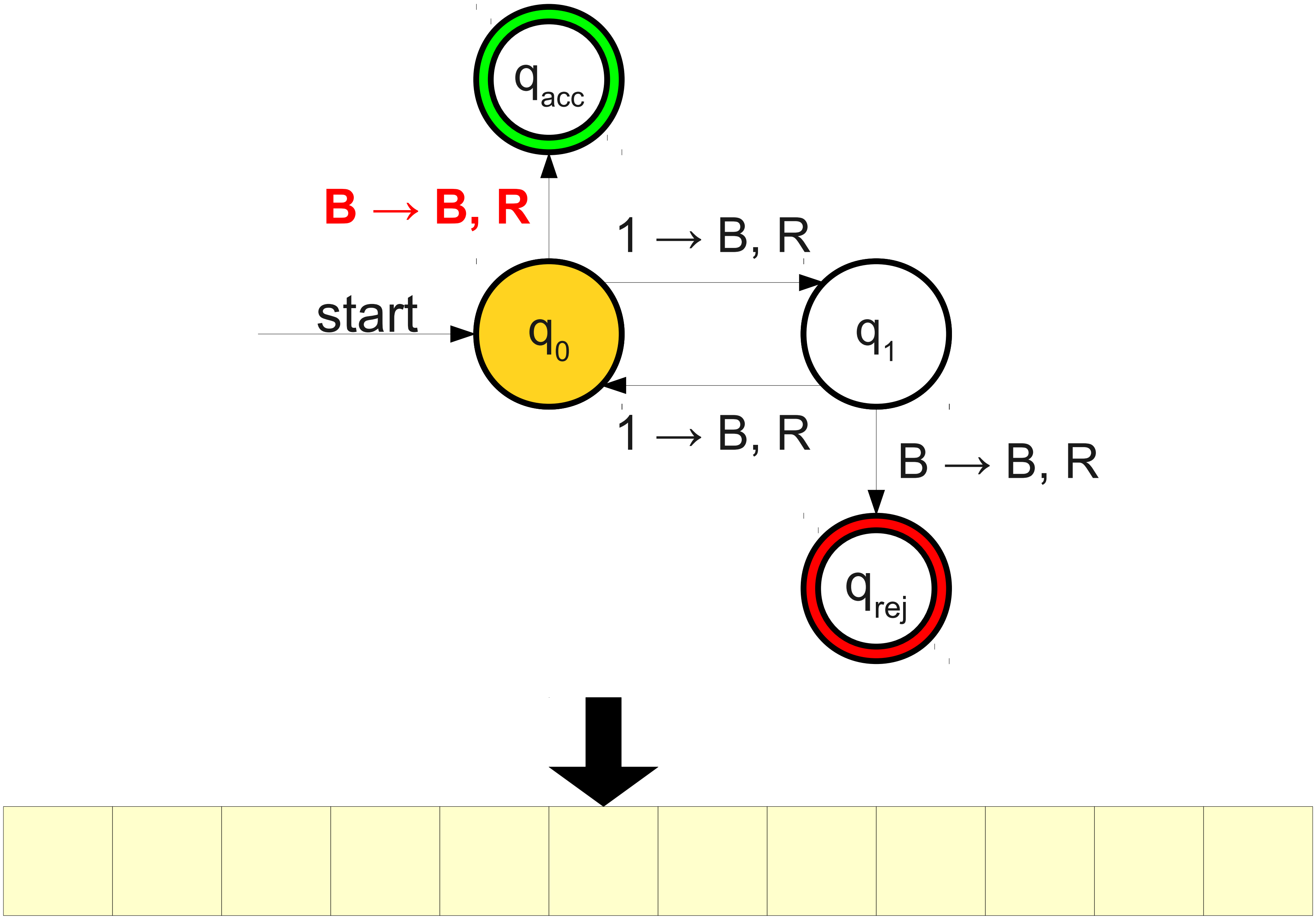
A Simple Turing Machine



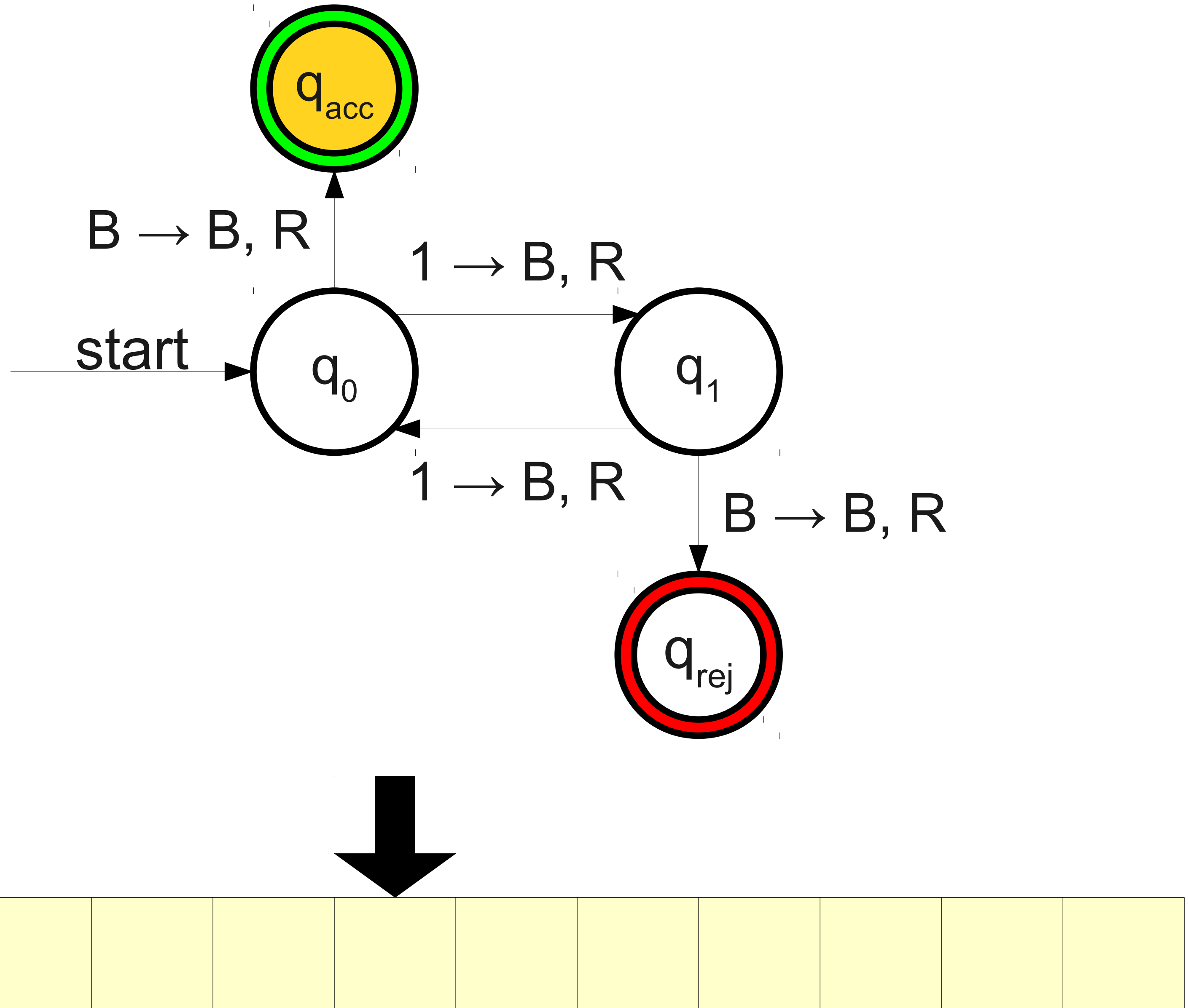
A Simple Turing Machine



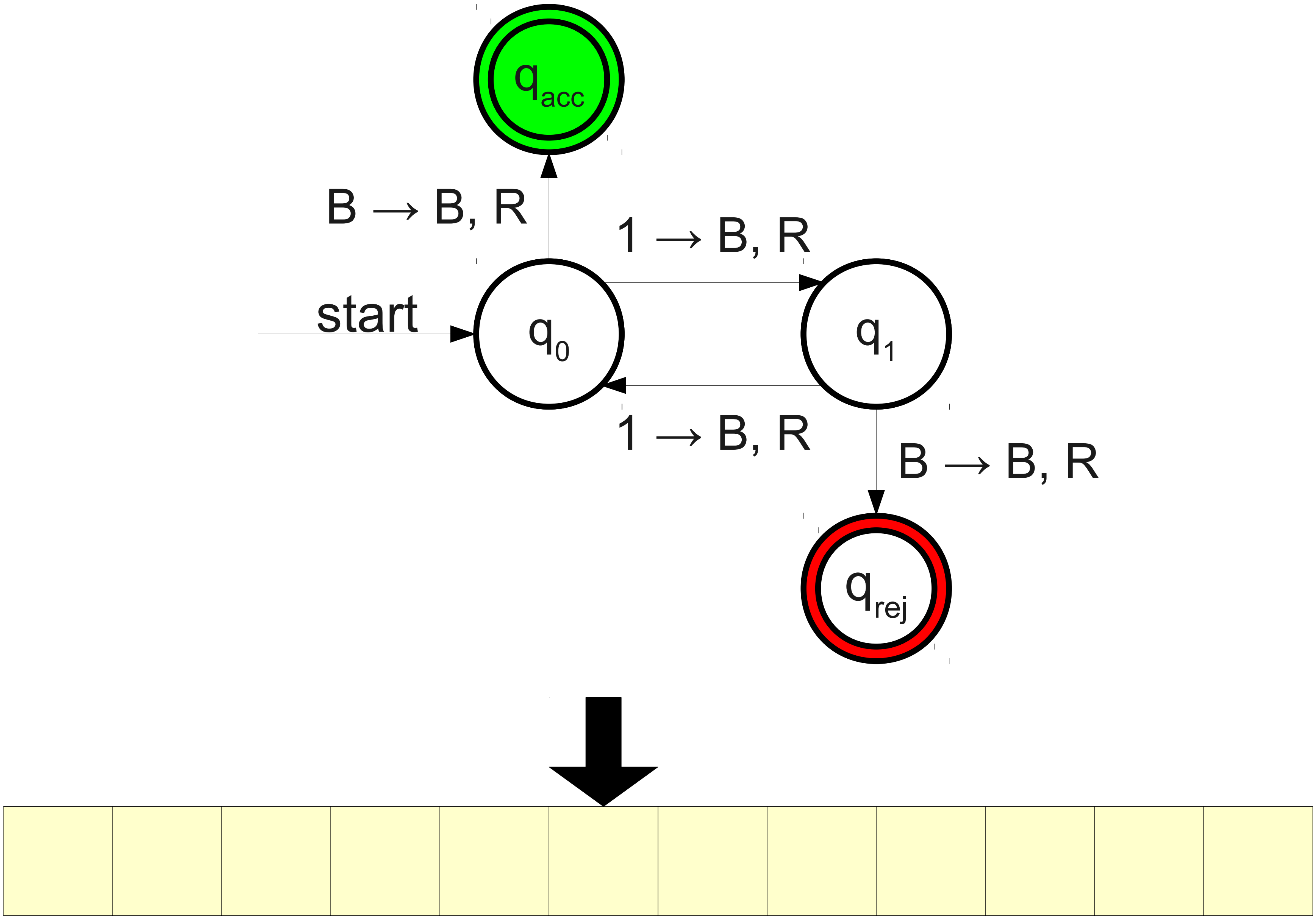
A Simple Turing Machine



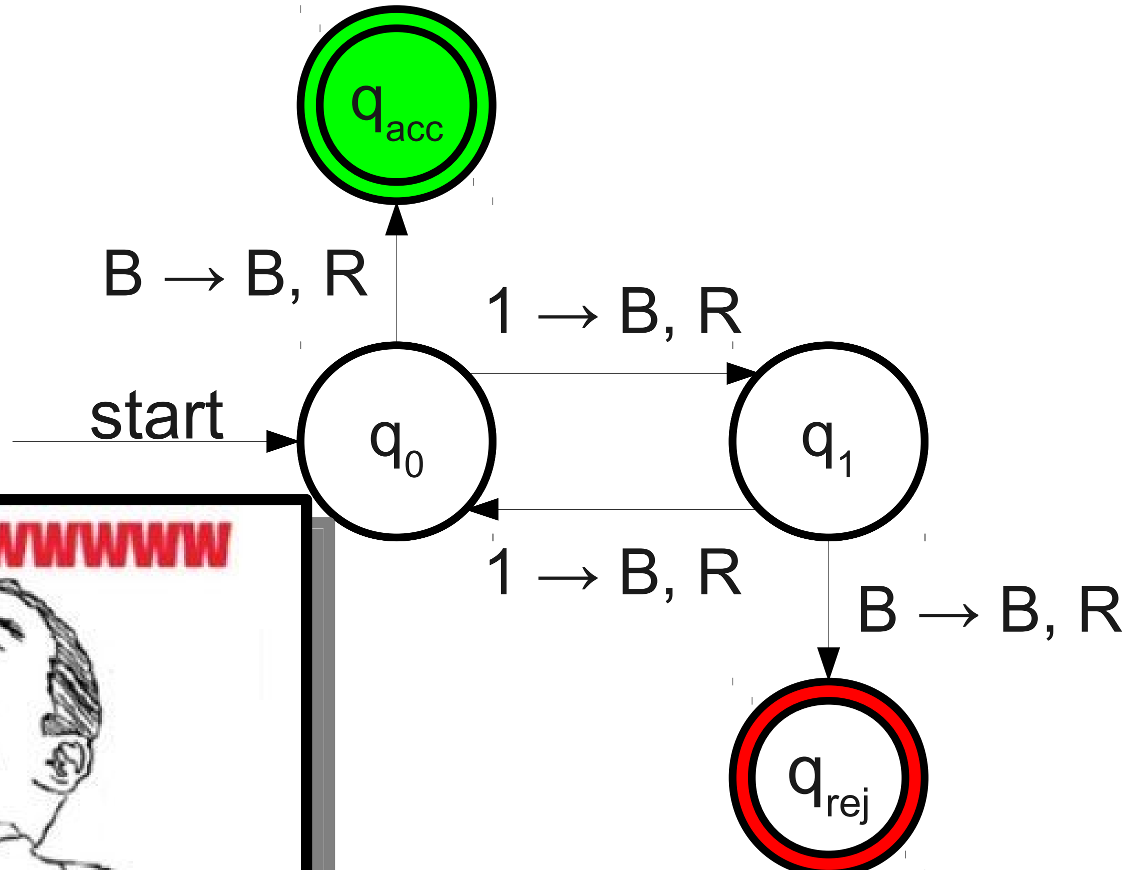
A Simple Turing Machine

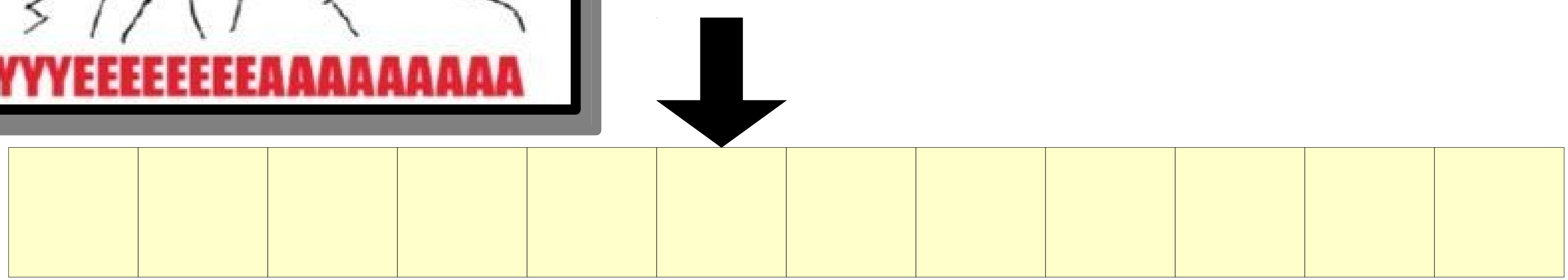


A Simple Turing Machine

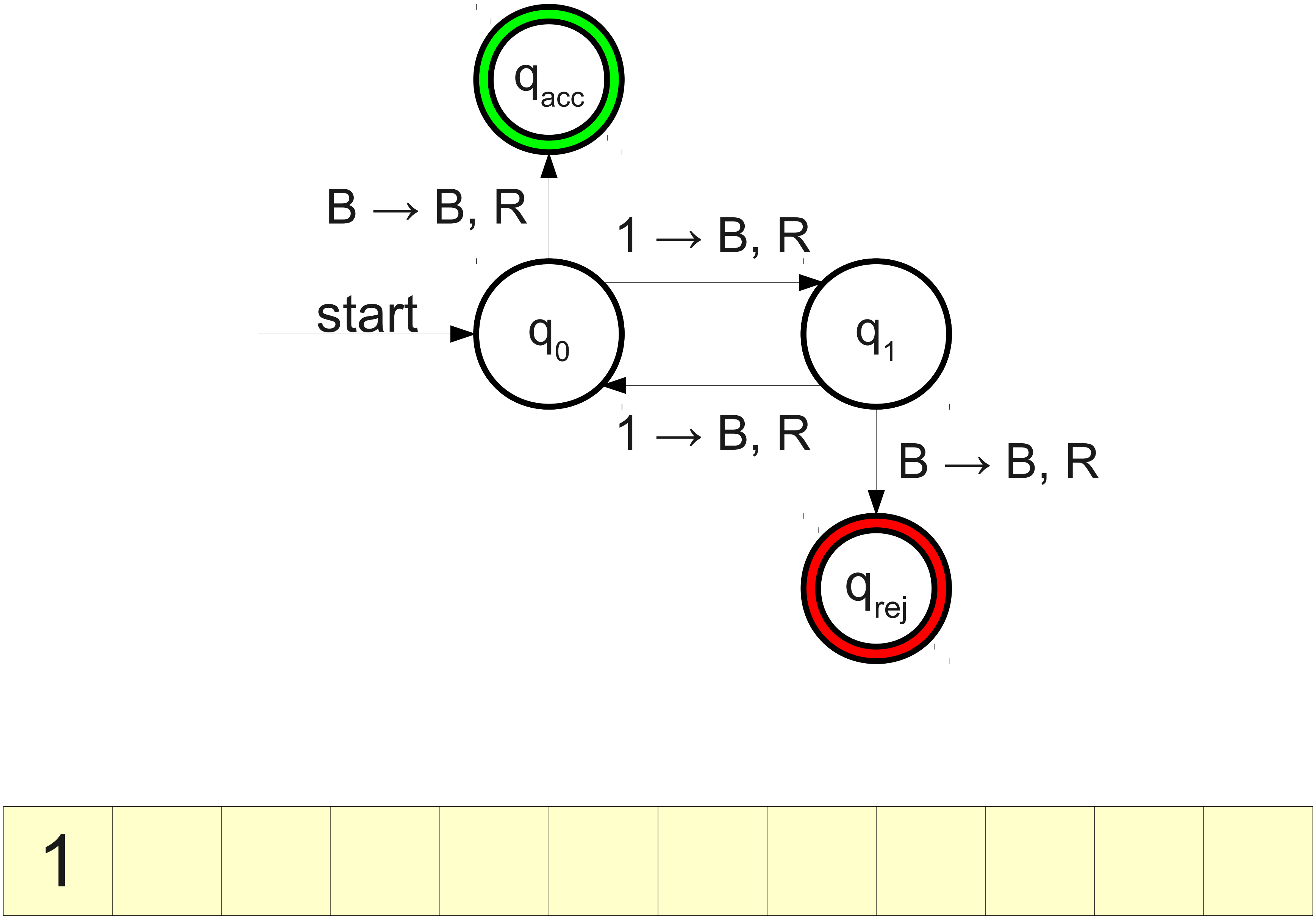


A Simple Turing Machine

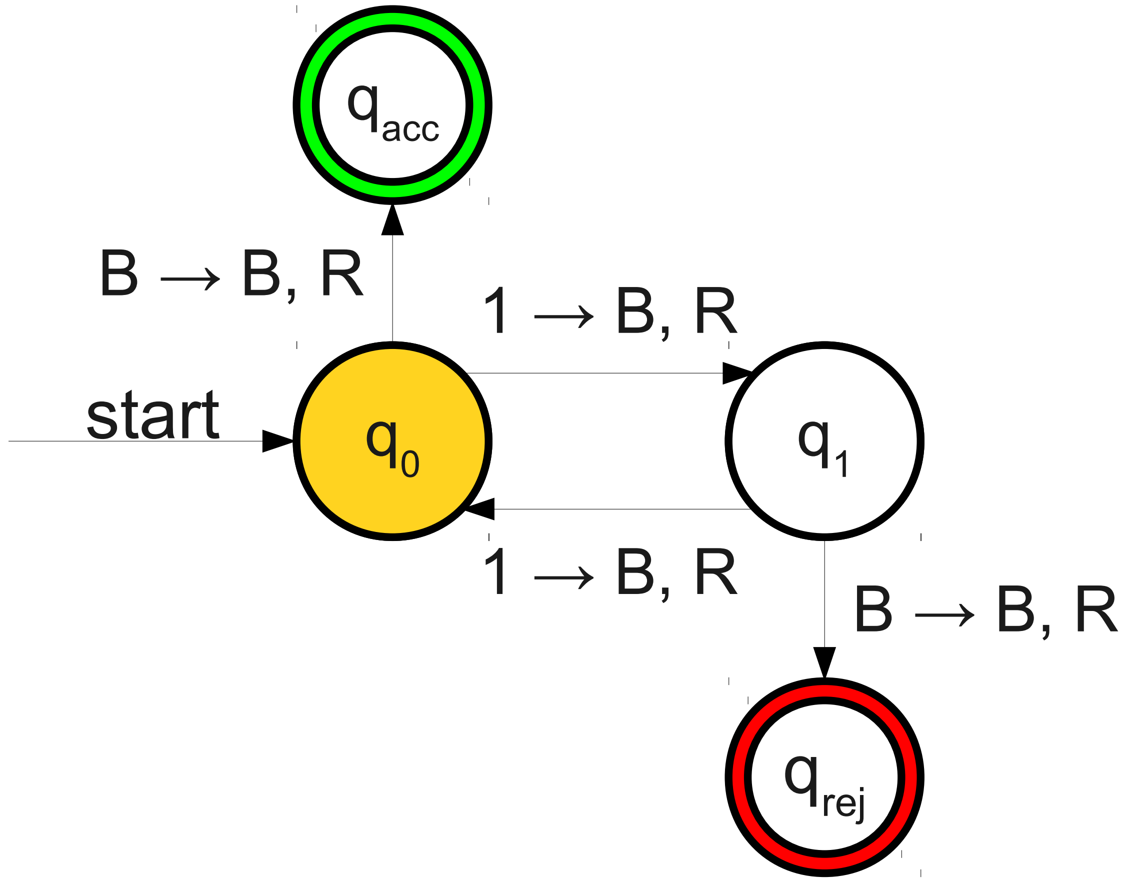


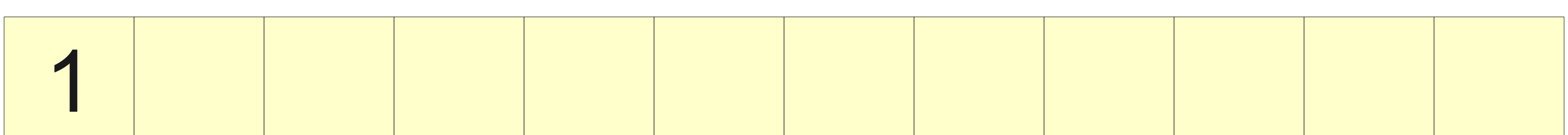


A Simple Turing Machine

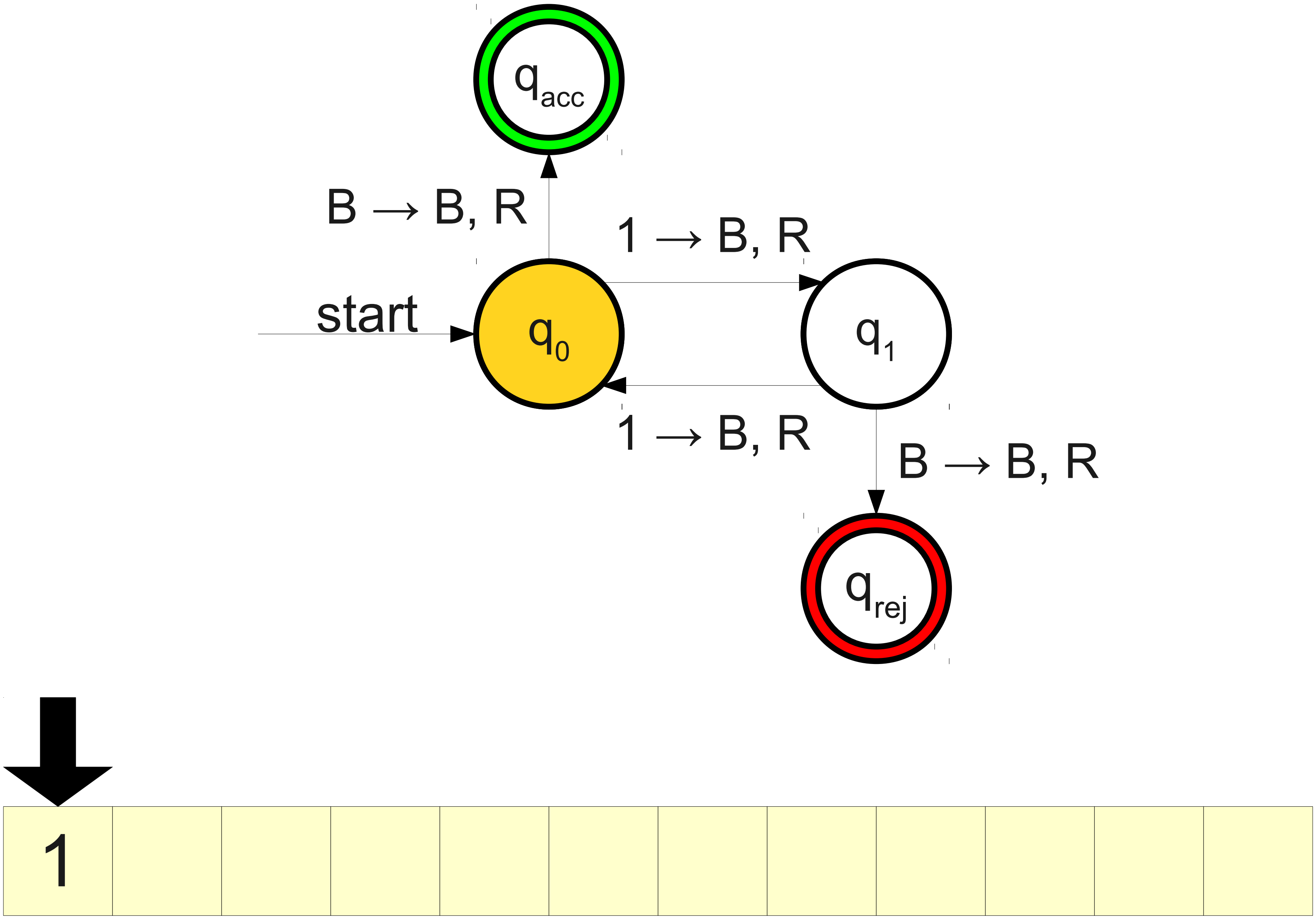


A Simple Turing Machine

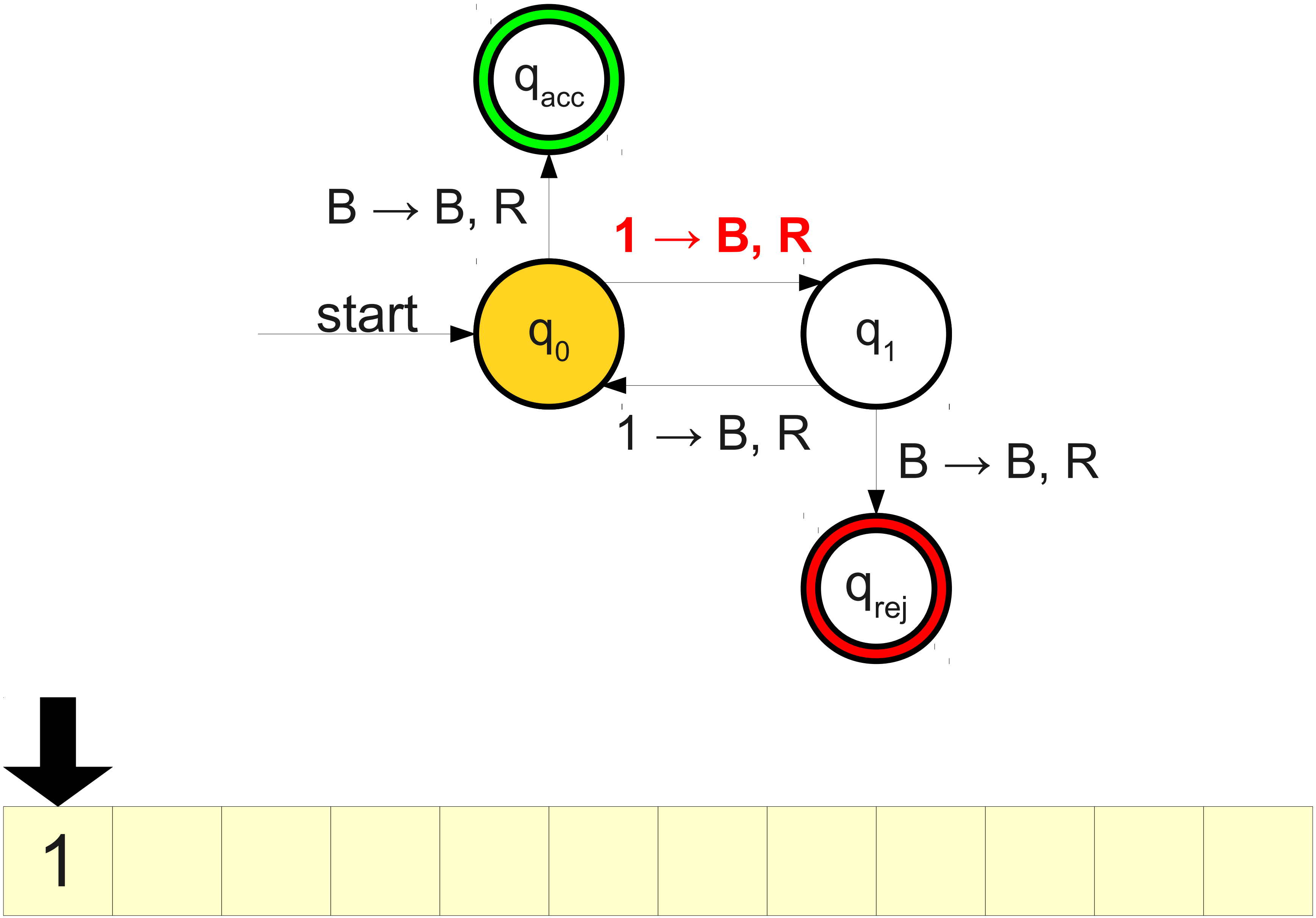




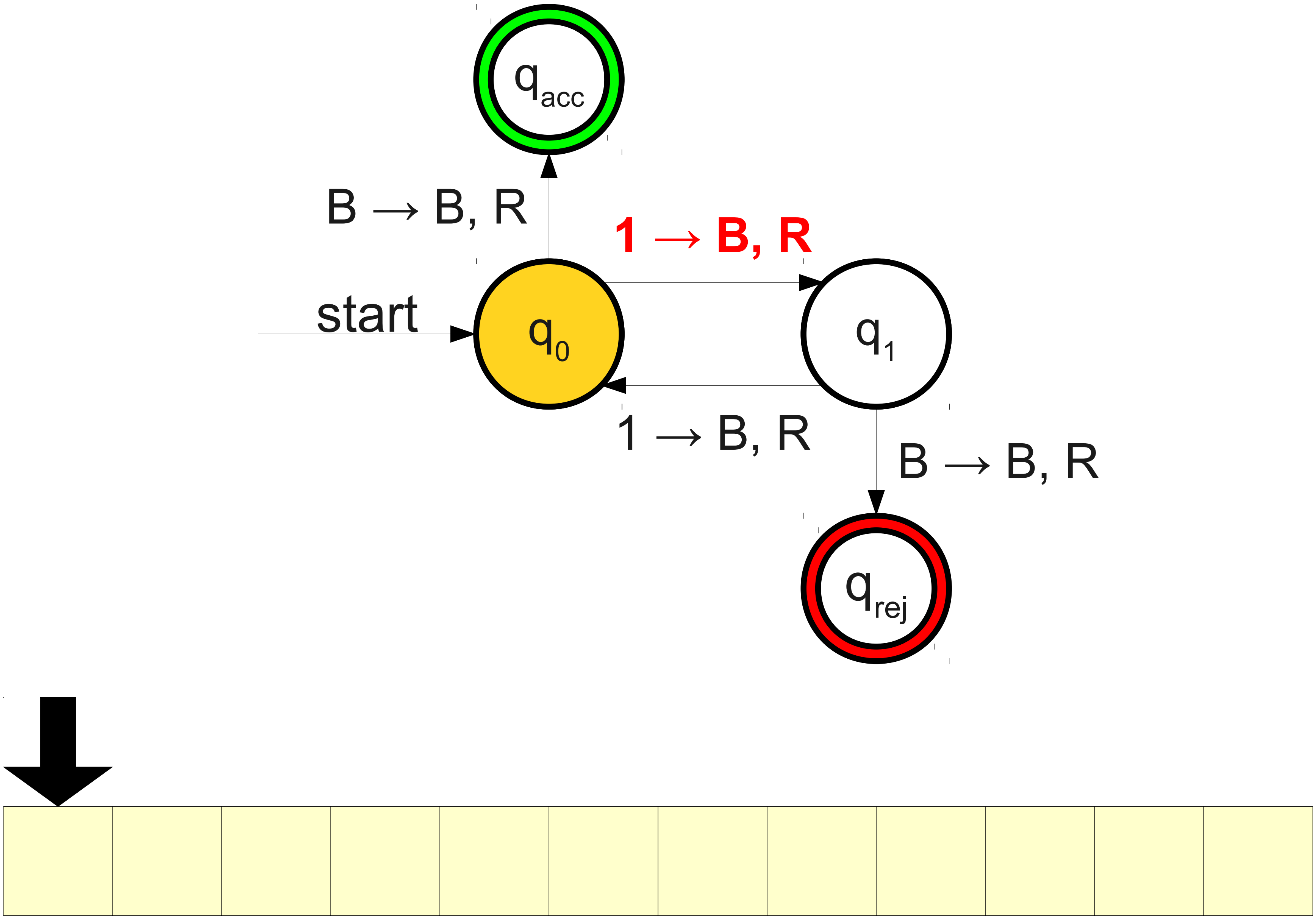
A Simple Turing Machine



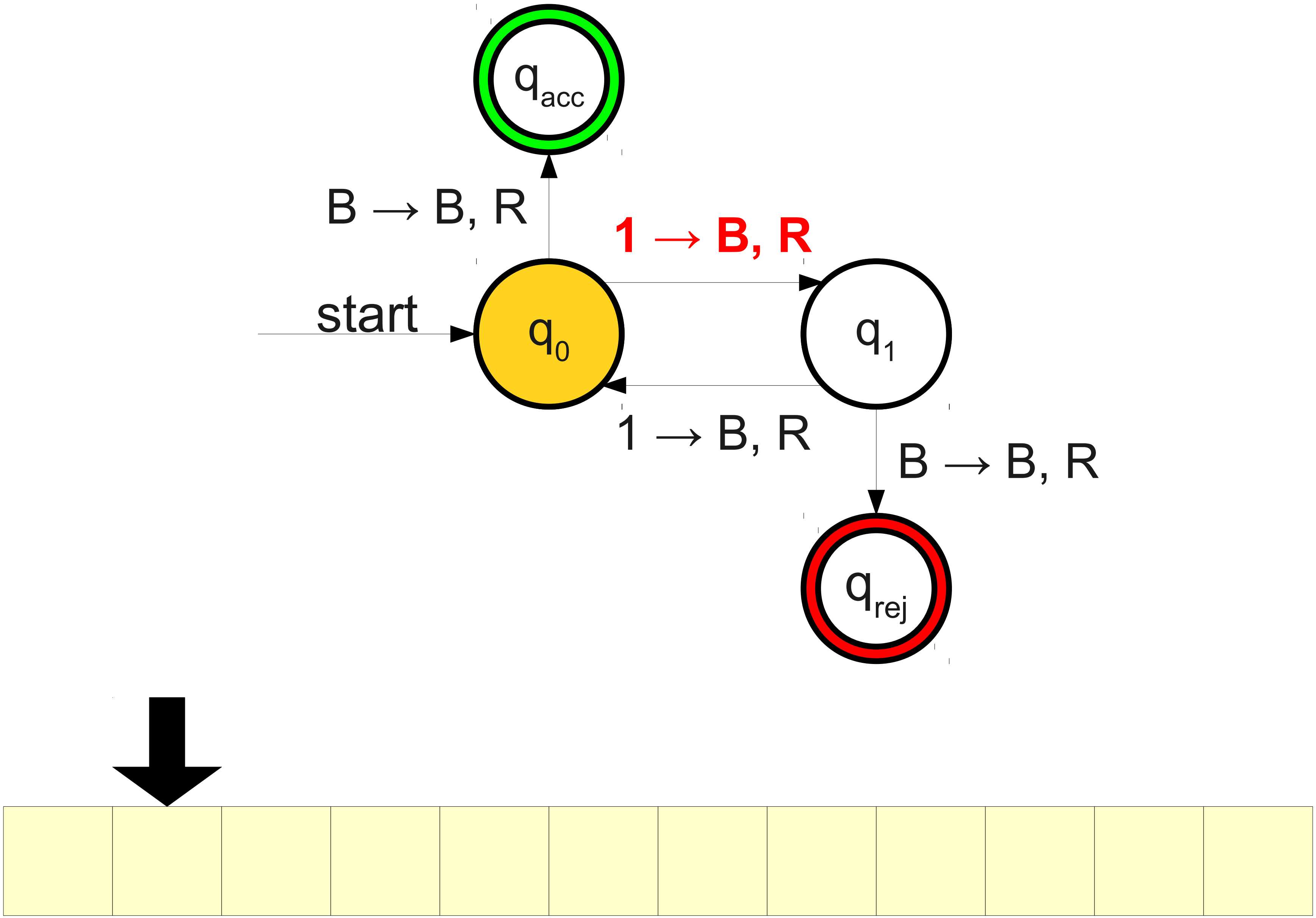
A Simple Turing Machine



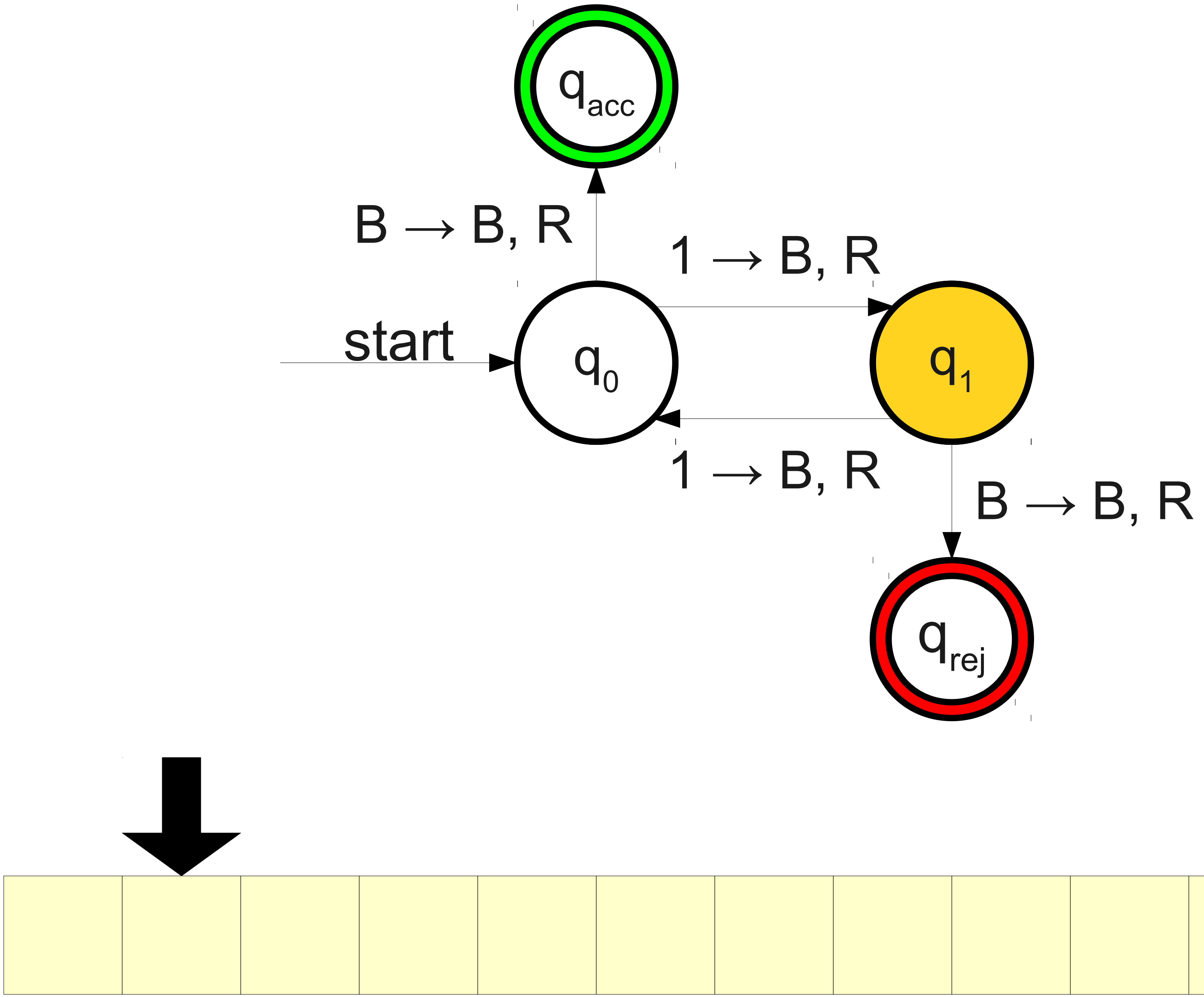
A Simple Turing Machine



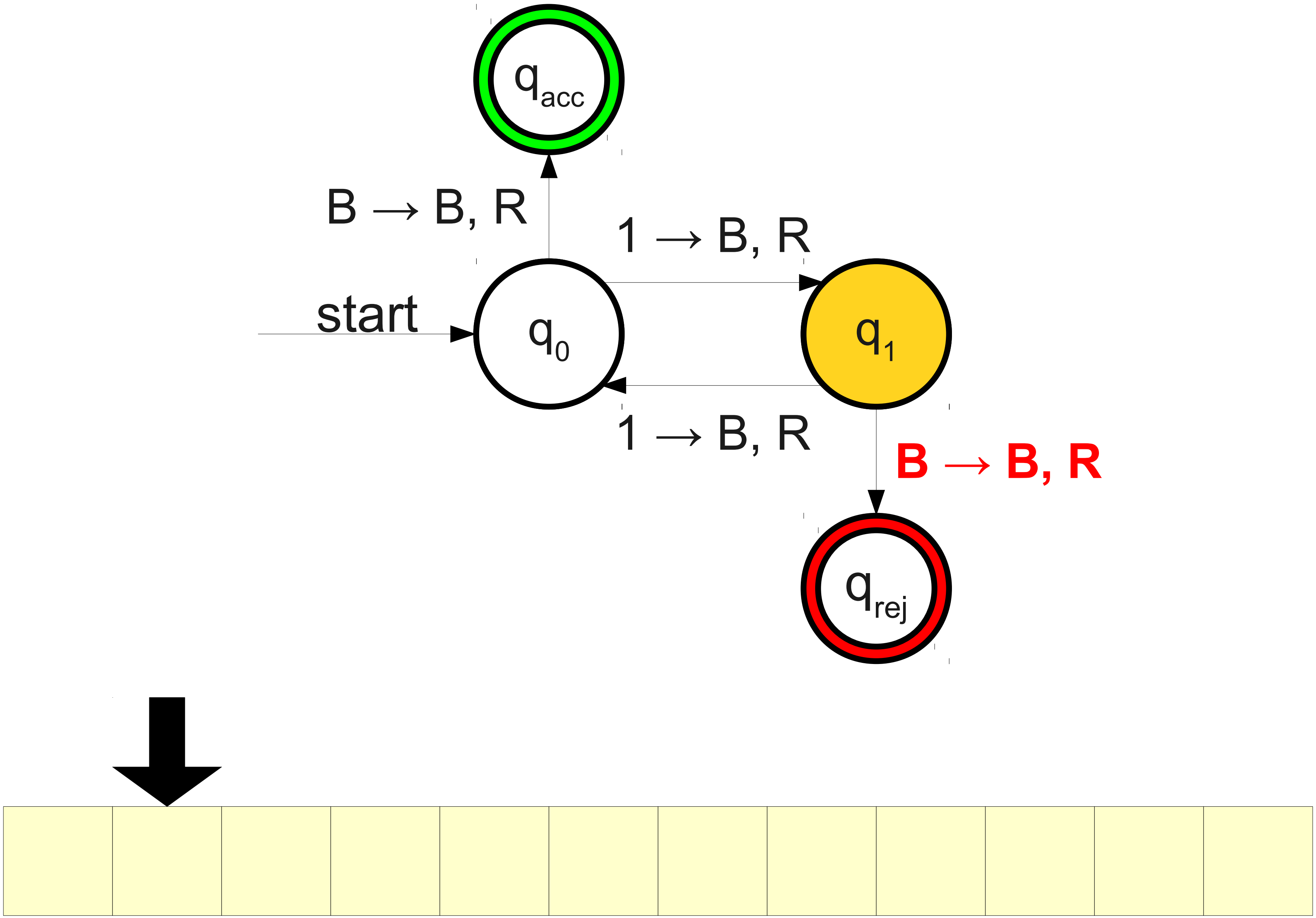
A Simple Turing Machine



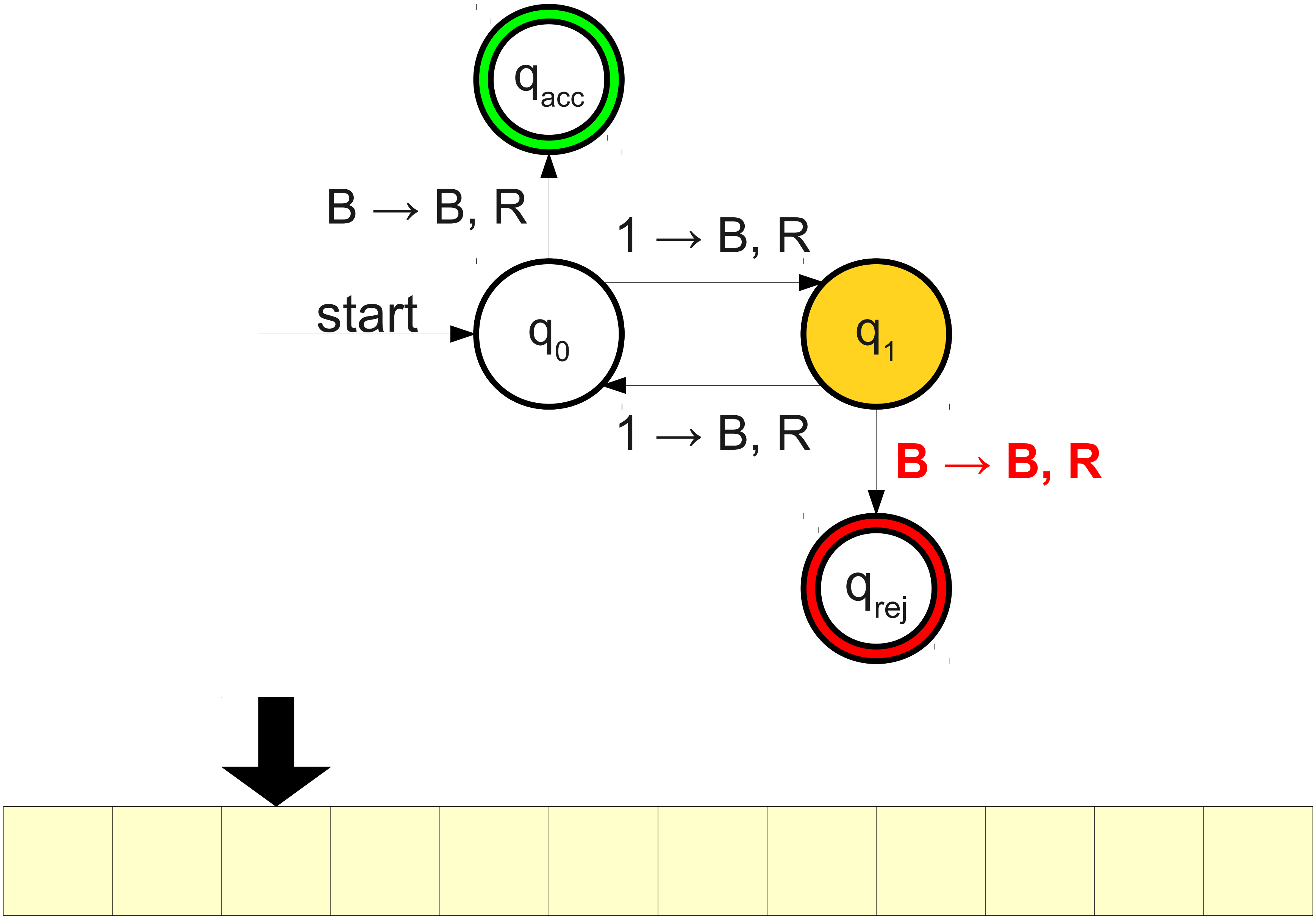
A Simple Turing Machine



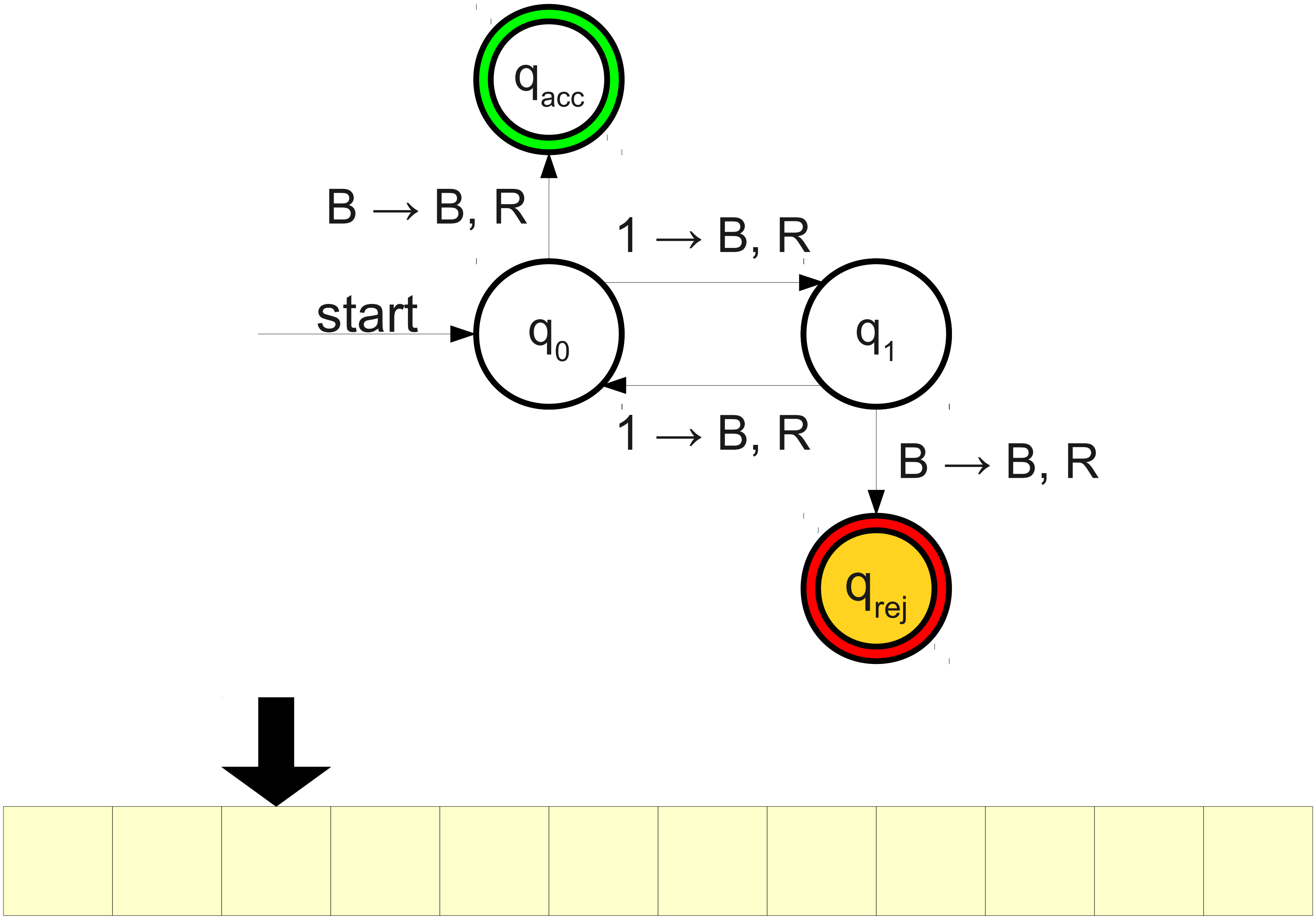
A Simple Turing Machine



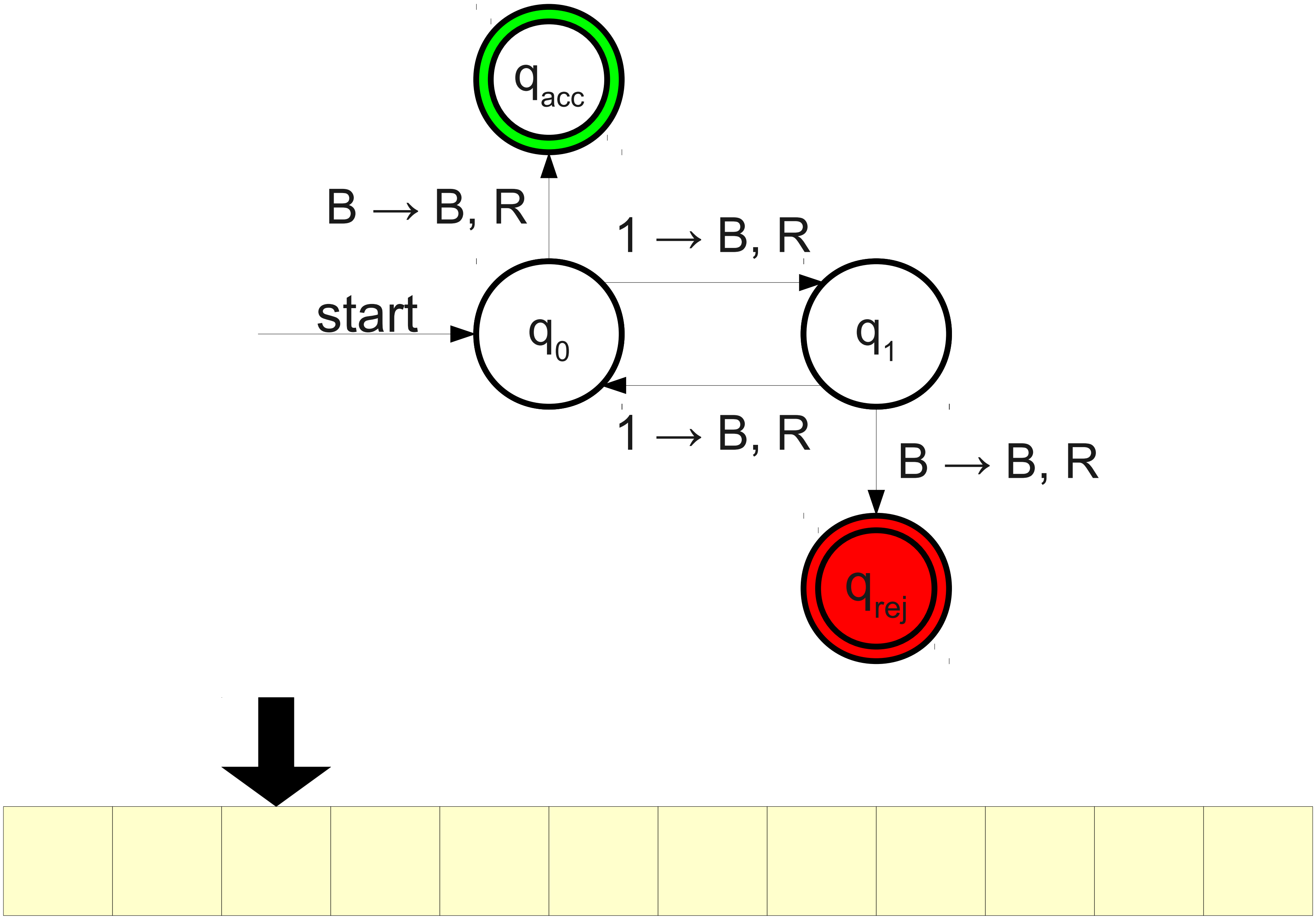
A Simple Turing Machine



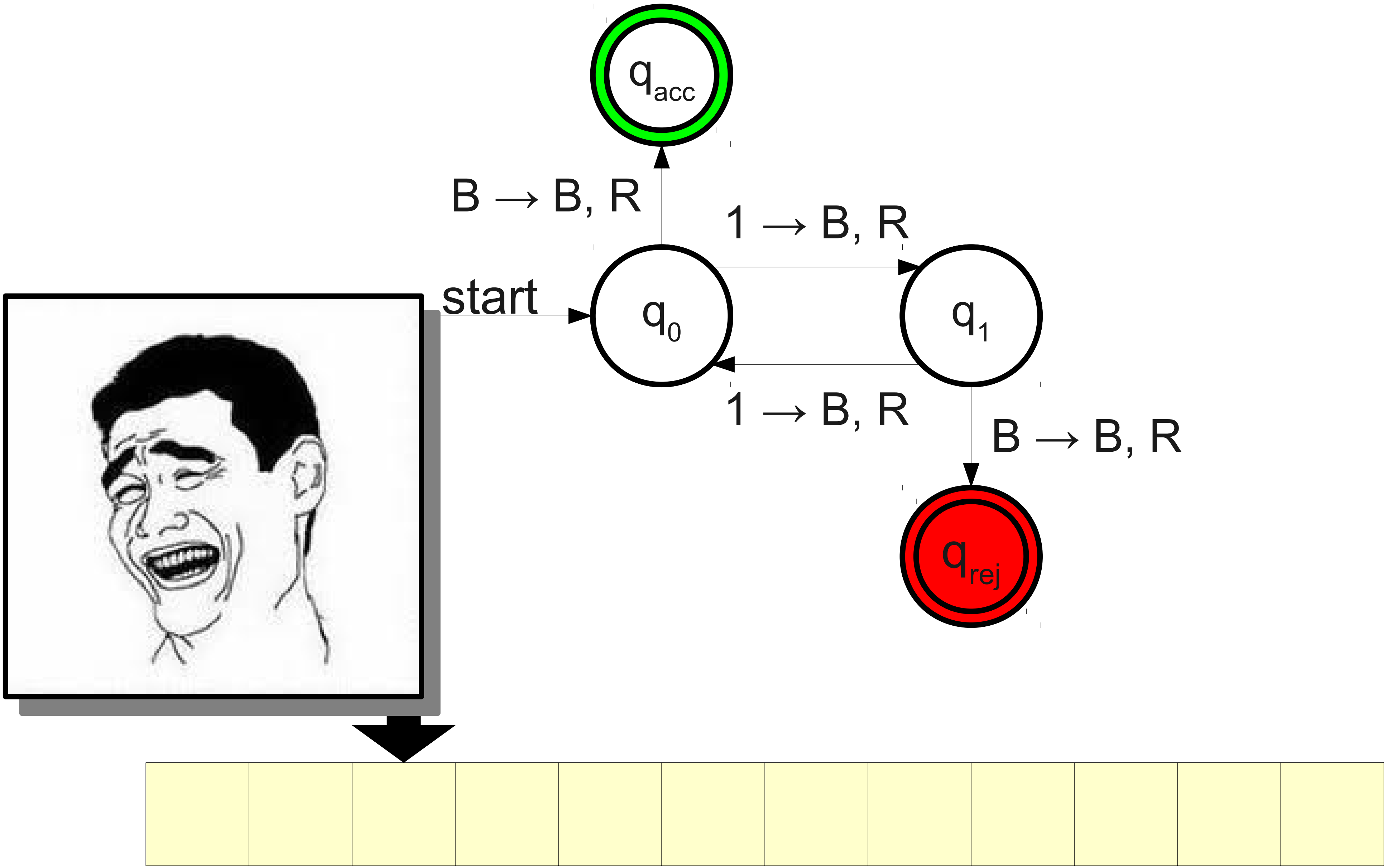
A Simple Turing Machine



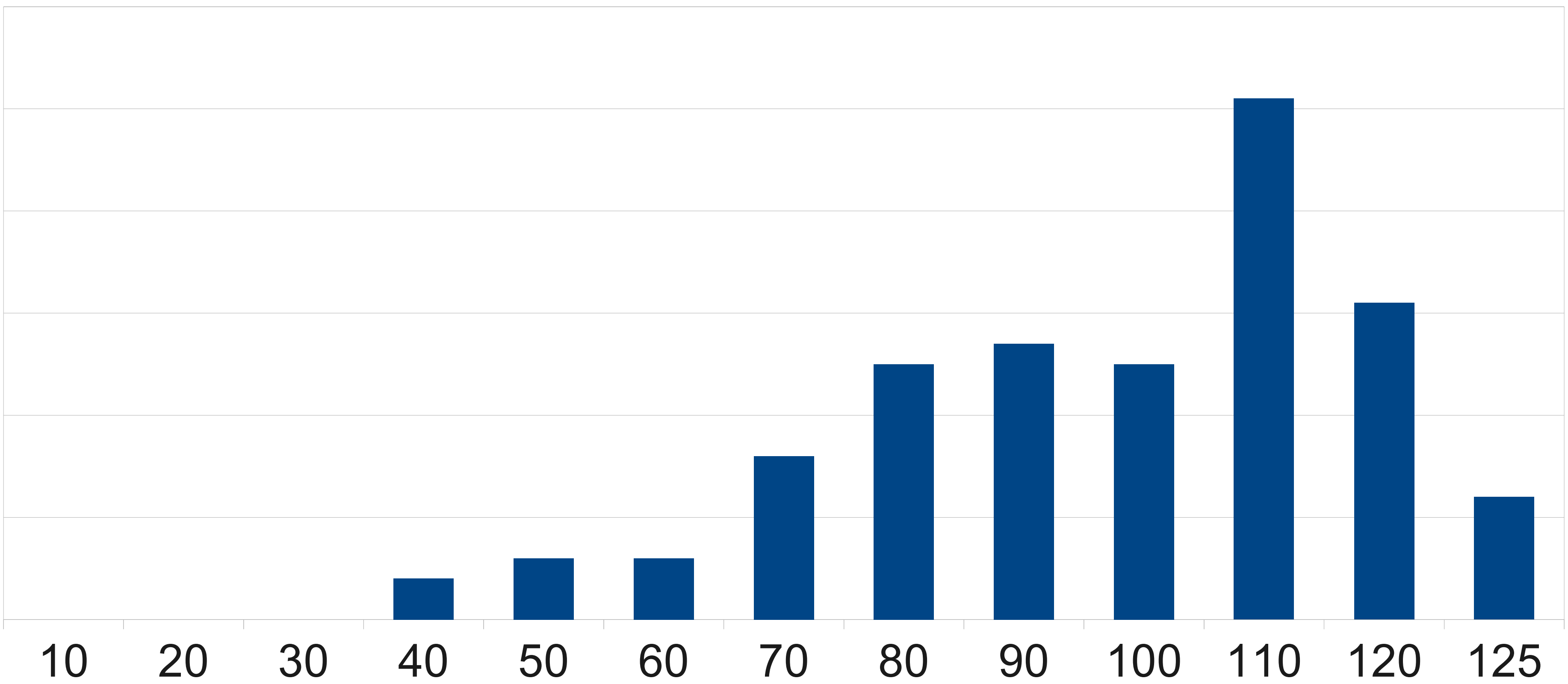
A Simple Turing Machine



A Simple Turing Machine



Problem Set 4 Grades



Mean: **92**
Median: **98**
Stdev: **21**