CS420

Introduction to the Theory of Computation

Lecture 18: Pumping Lemma for Context-Free Languages

Tiago Cogumbreiro

Today we will learn...

- The Pumping Lemma for Context-Free Languages
- Using the Pumping Lemma to identify non-context-free languages

> Section 2.3 Non-Context-Free Languages
> Supplementary material:
>
> - Professor Harry Porter's video

Exercise 1

Exercise 1

$$L_1 = \{w \mid w \in \{a, b\}^\star \land |w| \text{ is divisible by } 3\}$$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

Exercise 1

$L_1 = \{w \mid w \in \{a,b\}^\star \wedge |w| \text{ is divisible by } 3\}$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

(i) Regular: $\big((a+b)(a+b)(a+b)\big)^\star$

Exercise 2

Exercise 2

$L_2 = \{z \mid z \text{ has the same number of a's and b's}\}$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

Exercise 2

$L_2 = \{z \mid z \text{ has the same number of a's and b's}\}$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

(ii) Context-free:

$S \rightarrow aSb \mid bSa \mid \epsilon$

Exercise 3

Exercise 3

$$L_3 = \{a^n b^n c^n \mid n \geq 0\}$$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

Exercise 3

$L_3 = \{a^n b^n c^n \mid n \geq 0\}$

- (i) Regular? Give a REGEX/NFA/DFA
- (ii) Context-free (and not regular)? Give a CFG/PDA. Prove using the pumping lemma.
- (ii) Not context-free

Not context-free

How do we prove that a language is **not** context free?

The Pumping Lemma for CFL

Intuition

If we have a string that is long enough, then we will need to repeat a non variable, say R, in the parse tree.

Example

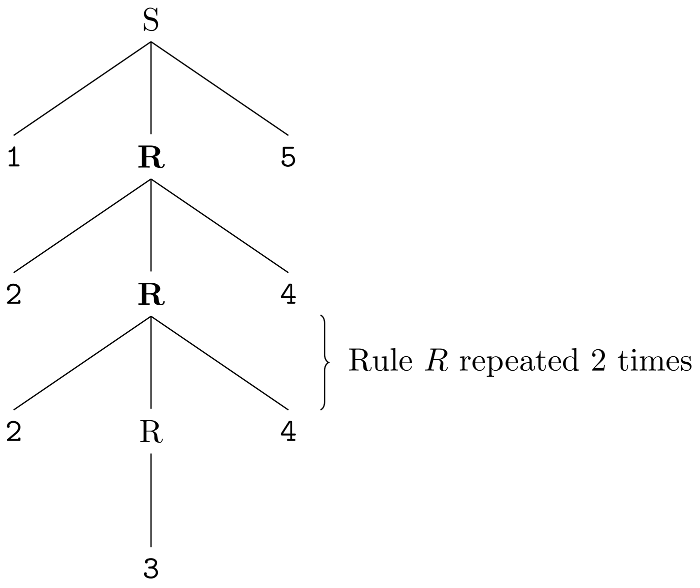
$S \to 1R5$

$R \to 2R4 \mid 3$

If we vary the number of times $R \to 2R4$ appears we note that:

- 1223445 is accepted (repeat 2 ×)
- 135 is accepted (repeat 0 ×)
- 12345 is accepted (repeat 1 ×)
- 122234445 is accepted (repeat 3 ×)

Parse tree for 1223445



Example

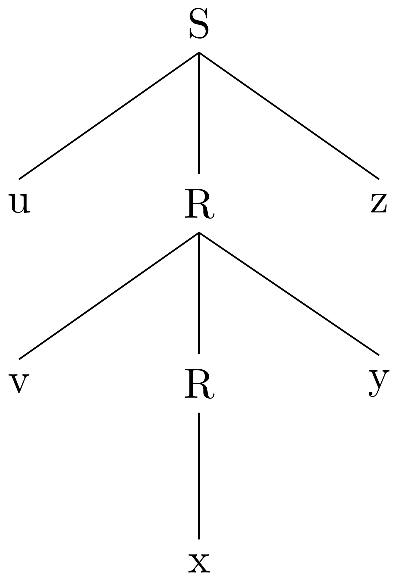
$$S \rightarrow 1R5$$
$$R \rightarrow 2R4 \mid 3$$

- $\underbrace{1}_{u}\underbrace{22}_{v^2}\underbrace{3}_{x}\underbrace{44}_{y^2}\underbrace{5}_{z}$, where $i = 2$
- $\underbrace{1}_{u}\underbrace{3}_{x}\underbrace{5}_{z}$, where $i = 0$
- $\underbrace{1}_{u}\underbrace{2}_{v^1}\underbrace{3}_{x}\underbrace{4}_{y^1}\underbrace{5}_{z}$, where $i = 2$
- $\underbrace{1}_{u}\underbrace{222}_{v^3}\underbrace{3}_{x}\underbrace{444}_{y^3}\underbrace{5}_{z}$, where $i = 3$

Thus, uv^ixy^iz is also in the language

Generalizing

For a long enough string, say $uvxyz$ in the language, then uv^ixy^iz is also in the language.



Pumping Lemma for context-free languages

> The pumping lemma tells us that all context-free languages (that have a loop) can be partitioned:

Every word in a context-free language, $w \in L$, can be partitioned into 5 parts $w = uvxyz$:

- an outer portion u and z
- a repeating portion v and y
- a non-repeating center portion x

Additionally, since v and y are a repeating portion, then v and y may be omitted or replicated as many times as we want and that word will also be in the given language, that is $uv^i xy^i z \in L$.

Example

$L_2 = \{z \mid z \text{ same number of a's and b's}\}$

You: Give me a string of size 4.

Example

$L_2 = \{z \mid z \text{ same number of a's and b's}\}$

You: Give me a string of size 4.

Example: abab

Example

$L_2 = \{z \mid z \text{ same number of a's and b's}\}$

You: Give me a string of size 4.

Example: abab

Me: I will partition abab into 5 parts $abab = uvxyz$ such that $uv^i xy^i z$ is accepted for any i:

$$\underbrace{a}_{u}\underbrace{a}_{v}\underbrace{\epsilon}_{x}\underbrace{b}_{y}\underbrace{b}_{z}$$

- $|vy| > 0$, since $|ab| = 2$
- $|vxy| \leq 4$, since $|a\epsilon b| = 2$

- $uxz = ab$ is accepted
- $u\underline{v}x\underline{y}z = a\underline{a}\epsilon\underline{b}b$ is accepted

- $u\underline{vv}x\underline{yy}z = a\underline{aa}\epsilon\underline{bb}b$ is accepted
- $u\underline{vvv}x\underline{yyy}z = a\underline{aaa}\epsilon\underline{bbb}b$ is accepted

The Pumping Lemma (Theorem 2.34)

For context-free languages

If L is **context free**, then there is a ***pumping length*** p where, if $w \in L$ and $|s| \geq p$, then there exists u, v, x, y, z such that:

1. $w = uvxyz$
2. $|vy| \geq 1$
3. $|vxy| \leq p$
4. $uv^i xy^i z \in L$ for any $i \geq 0$

```
Theorem pumping_cfl:
  forall L,
  ContextFree L →
  exists p, p ≥ 1 /\
  forall w, L w → (* w ∈ L *)
  length w ≥ p → (* |w| ≥ p *)
  exists u v x y z, (
    w = u ++ v ++ x ++ y ++ z /\ (* w = uvxyz *)
    length (v ++ y) ≥ 1 /\      (* |vy| ≥ 1 *)
    length (v ++ x ++ y) ≤ p /\ (* |vxy| ≤ p *)
    forall i,
    L (u ++ (pow v i) ++ x ++ (pow y i) ++ z)
    (* u v^i x y^i z ∈ L *)
  ).
```

Non-context-free languages

Theorem: non-context-free languages

Informally

If there exist a word $w \in L$ such that for any pumping length $p \geq 1$,

- $w \in L$
- $|w| \geq p$
- $w = uvxyz, |vy| \geq 1, |vxy| \leq p$ implies $\exists i, uv^i xy^i z \notin L$

then, L is not context-free.

Formally

```
Lemma not_cfl:
  forall (L:lang),
  (* Assume 0 *) (forall p, p ≥ 1 →
  (exists w,
  (* Goal 1 *) L w /\
  (* Goal 2 *) length w ≥ p /\
  forall u v x y z, (
    (* Assume 1 *) w = u ++ v ++ x ++ y ++ z →
    (* Assume 2 *) length (v ++ y) ≥ 1 →
    (* Assume 3 *) length (v ++ x ++ y) ≤ p →
    (* Goal 3 *) exists i,
    ~ L (u ++ (pow v i) ++ x ++ (pow y i) ++ z)
  ))) →
  ~ ContextFree L.
```

Theorem: non-context-free languages

Part 1

There exist a word w such that for any pumping length $p \geq 1$

Goal 1: $w \in L$

Goal 2: $|w| \geq p$

Part 2

Assumptions:

- H_1: $w = uvxyz$
- H_2: $|vy| \geq 1$
- H_3: $|vxy| \leq p$

Goal 3: $\exists i, uv^i xy^i z$

Exercise 3

Show that $L_3 = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

Proof.

We use the theorem of non-CFL.

For any pumping length $p > 0$ we pick $w = a^p b^p c^p$.

Goal 1: $w \in L_3$. ***Proof.*** which holds since $w = a^p b^p c^p$ and $p \geq 0$ (by hypothesis).

Goal 2: $|w| \geq p$. ***Proof.*** $|w| = 3p$, thus $|w| \geq p$.

Exercise 3

Assumptions

- H_1: $w = uvxyz$
- H_2: $|vy| \geq 1$
- H_3: $|vxy| \leq p$

Goal 3: $\exists i, uv^i xy^i z \notin L_3$

Proof. We pick $i = 2$. Let

$$w = a^p b^p c^p$$

Exercise 3

Assumptions

- H_1: $w = uvxyz$
- H_2: $|vy| \geq 1$
- H_3: $|vxy| \leq p$

Goal 3: $\exists i, uv^i xy^i z \notin L_3$

Proof. We pick $i = 2$. Let

$$w = a^p b^p c^p$$

Let $N = |vxy|$. From (H_1) $a^p b^p c^p = u\underline{vxy}z$ and (H_2) $|vxy| \leq p$ we can conclude that vxy can match one of two cases:

1. vxy has only a's (or only b's) (or only c's)
2. vxy has only a's and b's (or only b's and c's)

Proof. (Continuation...)

Case: only contains one type of letter

1. Without loss of generality, let us consider that there are only a's.
2. We must show that $a^{p+N} b^p c^p \notin L_3$.
3. It is enough to show that there are more a's than b's, thus $p + N \neq p$. This holds because $N > 0$ (from H_2).

Proof. (Continuation...)

Case: contains two types of letters.

Without loss of generality, let us consider that v contains a's and y contains b's. Let $N = n + m$, where n is the number of a's and m is the number of b's.

$$\underbrace{a^p b^p c^p}_{uvxyz} = \underbrace{a^{p-n}}_{u}\underbrace{a^n b^m}_{vxy}\underbrace{b^{p-m} c^p}_{z}$$

Next, we recall that vx may still contain only a's, or it may contain a's and b's (because of H_2 and H_3). In the case of the latter, then since we picked $i = 2$ the string is trivially not in L_3. The rest of the proof assumes that v only has a's and y only has b's.

Our goal is to show that

$$\underbrace{a^{p-n}}_{u}\underbrace{a^{n+|v|} b^{m+|y|}}_{v^2 x y^2}\underbrace{b^{p-m} c^p}_{z} \notin L_3$$

Proof. (Continuation...)

Goal

$$\underbrace{a^{p-n}}_{u}\underbrace{a^{n+|v|}b^{m+|y|}}_{v^2xy^2}\underbrace{b^{p-m}c^{p}}_{z} \notin L_3$$

Since (H_2) $|vy| \geq 1$, then either $|v| \geq 1$ or $|y| \geq 1$.

- If $|v| \geq 1$, it is enough to show that the number of a's differs from the number of c's, thus $p - n + n + |v| \neq p$, which holds because $|v| \geq 1$.
- If $|y| \geq 1$, then we must show that the number of b's differs from the number of a's. Hence, $m + |y| + p - n \neq p$, which holds because $|y| \geq 1$.

Exercise 4

$L_4 = \{ww \mid w \in \{a, b\}^\star\}$

The language is **not** context free.

We pick $w = a^p b^p a^p b^p$

Goal 1: $w \in L_4$, because $a^p b^p \in \{a, b\}^\star$

Goal 2: $|w| \geq p$, because $|w| = 4p$.

Goal 3: $\exists i, uv^i xy^i z \notin L_4$.

Assumptions

- $H_1 : w = uvxyz$
- $H_2 : |vy| \geq 1$
- $H_3 : |vxy| \leq p$

(Proof...) Let $|vxy| = V$. If $a^p b^p a^p b^p = uvxyz$, then because $H_3 : |vxy| \leq p$, we have that w can be divided into two cases:

(Proof...) Let $|vxy| = V$. If $a^p b^p a^p b^p = uvxyz$, then because $H_3 : |vxy| \leq p$, we have that w can be divided into two cases:

Case 1: only a's/only b's.

Without loss of generality we handle the case for only a's and any portion of the string will work.

Thus, $w = \underbrace{a^{|u|}}_{u}\underbrace{a^V}_{xyz}\underbrace{b^p a^p b^p}_{z}$ and $|u| + V = p$.

(Proof...) Let $|vxy| = V$. If $a^p b^p a^p b^p = uvxyz$, then because $H_3 : |vxy| \leq p$, we have that w can be divided into two cases:

Case 1: only a's/only b's.

Without loss of generality we handle the case for only a's and any portion of the string will work.

Thus, $w = \underbrace{a^{|u|}}_{u} \underbrace{a^V}_{xyz} \underbrace{b^p a^p b^p}_{z}$ and $|u| + V = p$.

Case 2: some a's and some b's. Let A be the number of a's and B be the number of b's, where $V = A + B$. Without loss of generality we handle the case where the string has some a's and some b's. Thus, $w = \underbrace{a^{p-A}}_{u} \underbrace{a^A b^B}_{xyz} \underbrace{b^{p-B} a^p b^p}_{z}$

Why do we need only this 2 cases?

- Whatever a's and b's you pick (even in the middle), you must always show that that either you add/subtract $|x|$ non-empty and then you add/subtract $|y|$ non empty.

Turing machines

1. Recap

- Deterministic Finite Automaton that recognize Regular Languages
- Pushdown Automaton that recognize Context-Free Languages

2. Turing Machines

- Introduced to research into the foundations of mathematics
- characterizes **computation**
- can represent any computable machine unbounded by time and space

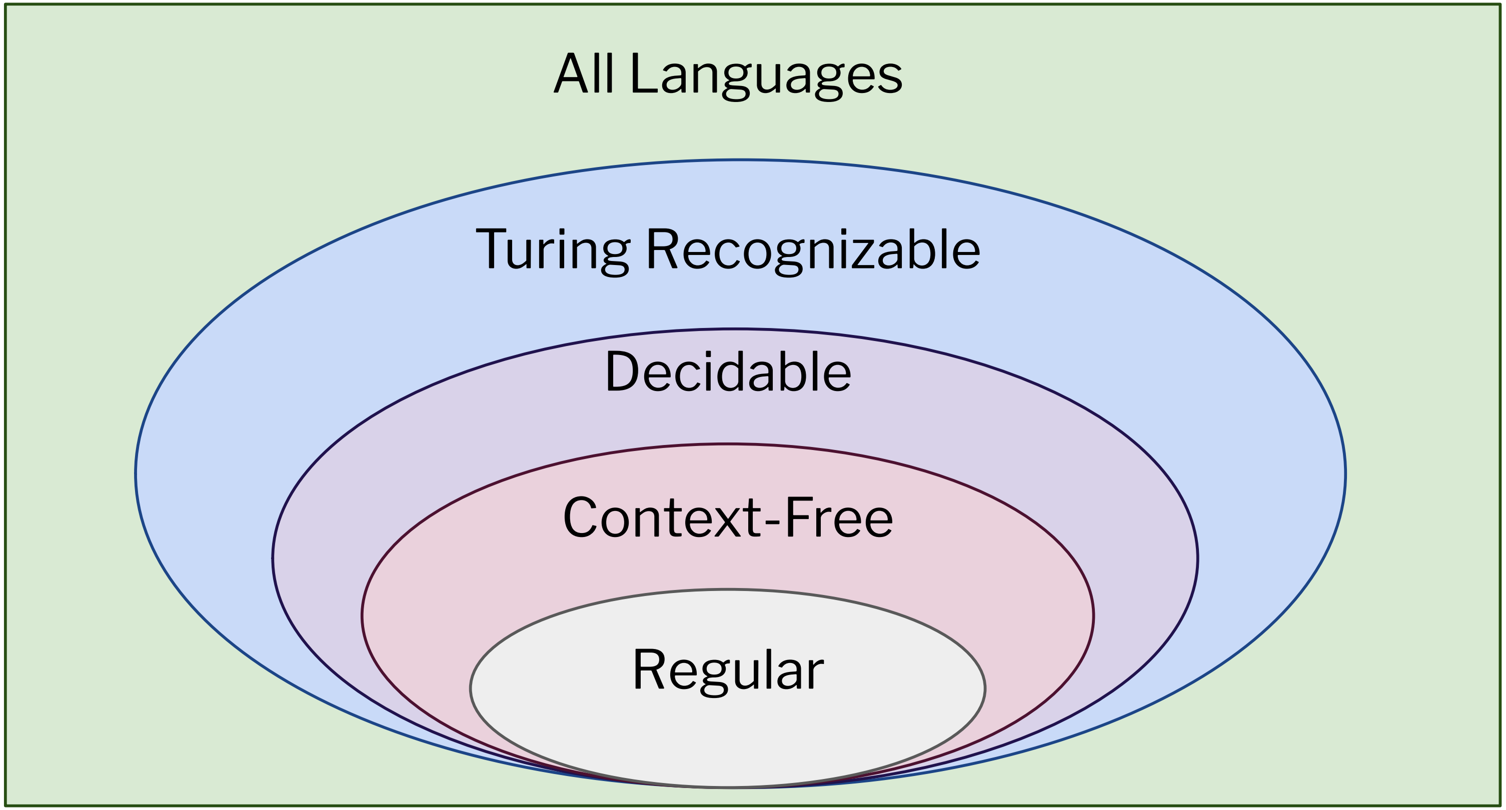
In general, describes problems of the form:

> Decide for any given x whether or not x has property P

Next lecture

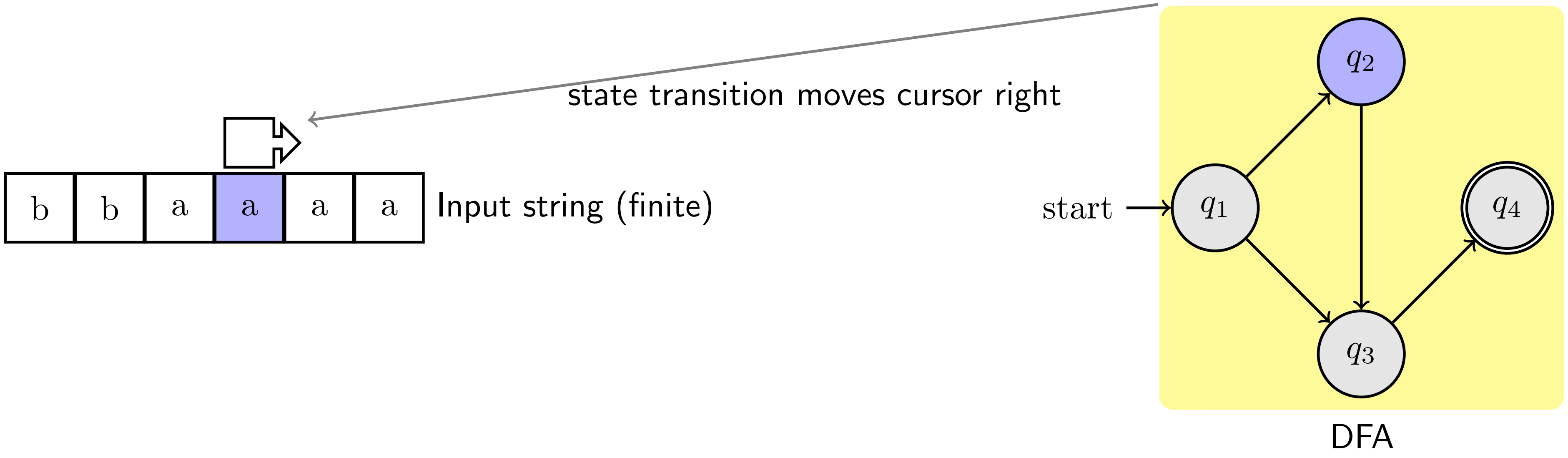
- Historical background on Turing machines

The big picture



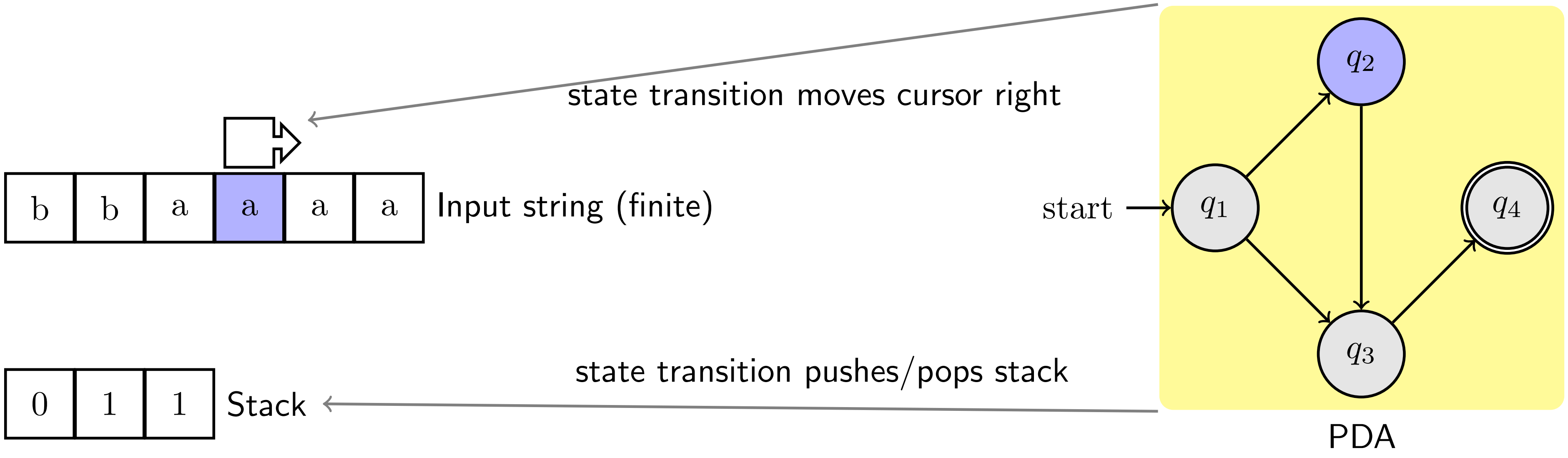
Recall DFA operation

- Automaton processes a finite input string (acceptance)
- Transition moves the cursor forward
- Final state accepts the string if the cursor is at the end



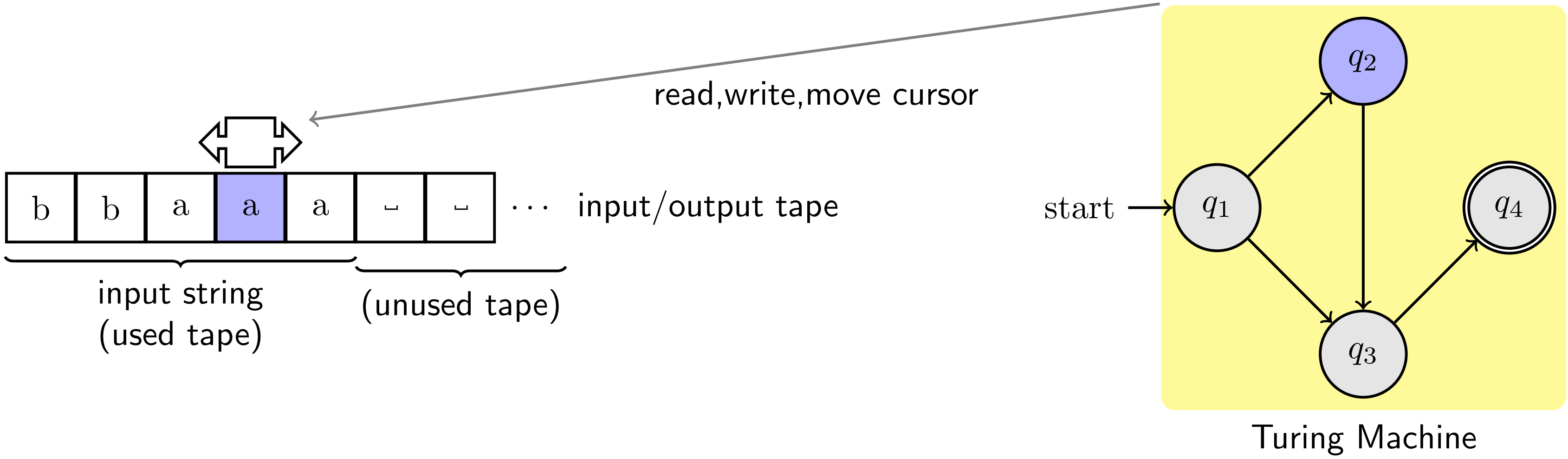
Recall PDA operation

- Automaton processes a finite input string (acceptance) and a stack
- Transition may move the cursor forward and may push/pop the stack
- Final state accepts the string if the cursor is at the end



Turing Machine operation

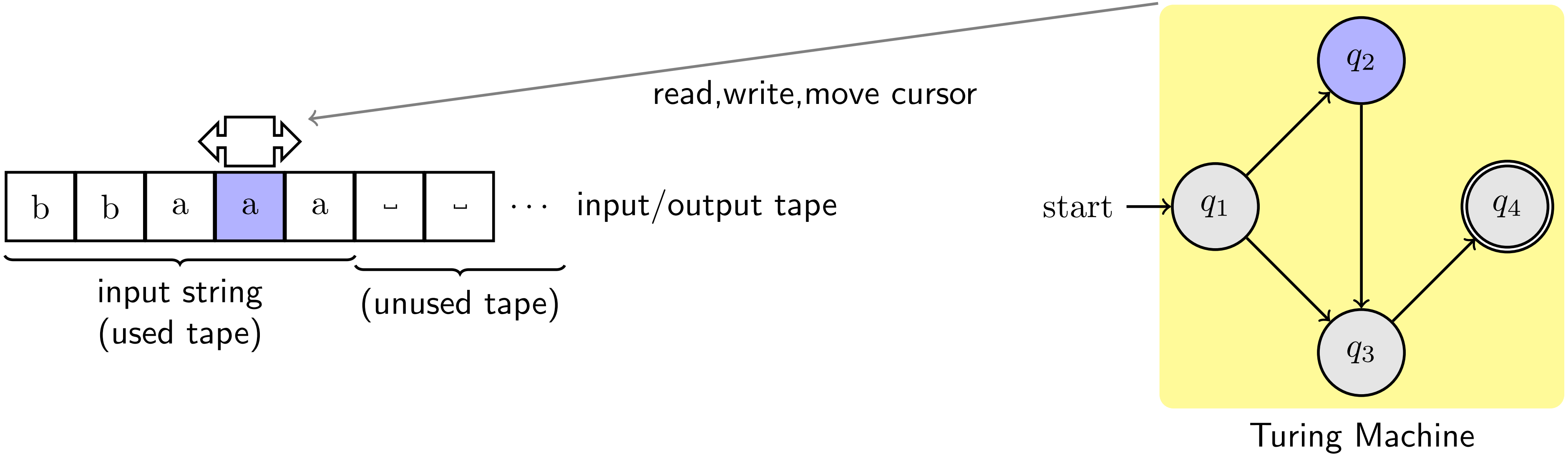
- Automaton processes an **infinite tape**
- Transition may move the cursor forward **or backward**
- Elements of the tape may be written or read
  (tape combines the input string and the stack)
- Tapes may contain a special character called black, notation ␣ (akin to NULL)



Turing Machine operation

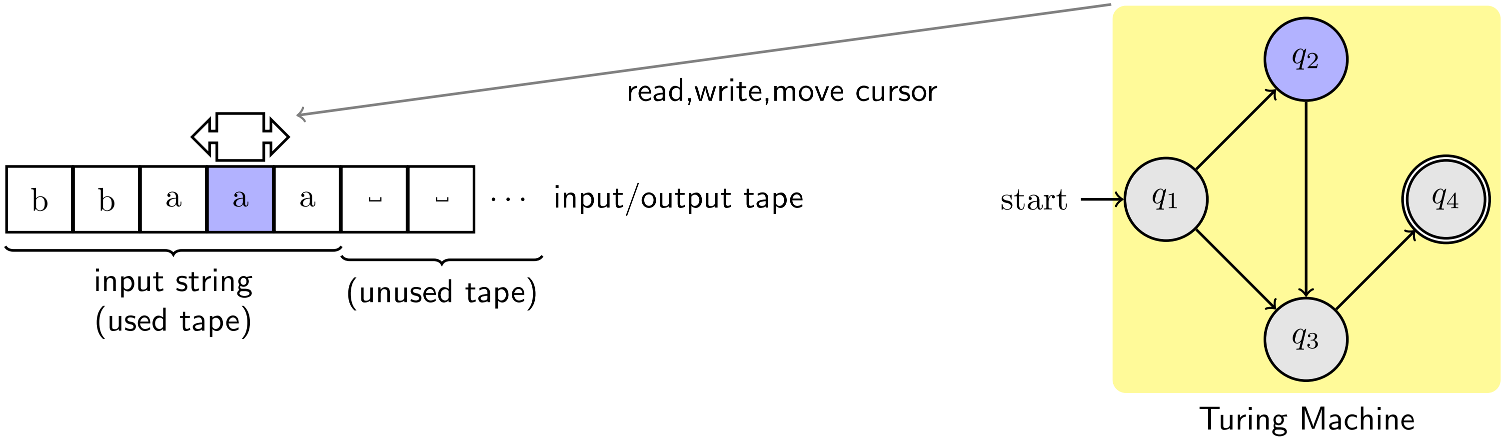
- The **tape head** (or cursor) points to a position in the tape (akin the instruction pointer in a processor)
- Transition: read $\to$ write, move direction

$$q \xrightarrow{a \to b, \mathsf{R}} q'$$



Turing Machine control

- The automaton (the turing machine) is known as the **control** or the **program**
- The automaton is deterministic (nondeterminism has same expressiveness!)
- A single initial state
- A single accept state
- A reject state



Turing Machines acceptance

Given a tape (with an ***input string***) and a Turing machine, there are three kinds of answers:

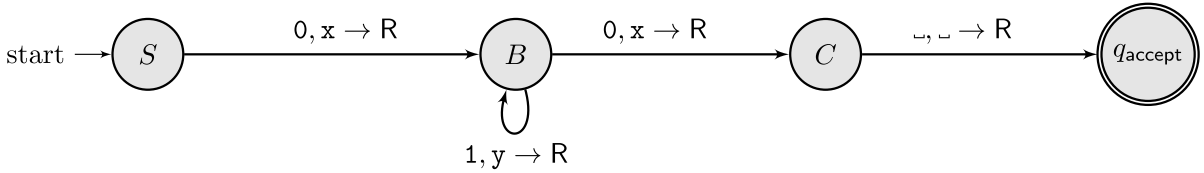
- **Accept**
  Whenever the machine reaches the accept state, the automaton halts and the input string is accepted.
- **Reject**
  Whenever the machine reaches the reject state, the automaton halts and the input string is rejected.
- **Loop forever**
  The machine keeps doing transitions in a loop, never accepting nor rejecting the input string.

> While a PDA and a DFA can either accept or reject a string, a Turing machines can also loop forever!

Examples

Example 1

$L = 01^{\star}0$



- Deterministic (only one outgoing edge **per input**)
- Convention: missing transitions go to reject state (hidden).

Example

State	Tape
S	01110
B	x1110
B	xy110
B	xyy10
B	xyyy0
C	xyyyx_
q_{accept}	xyyyx _

Simulate

Example 2

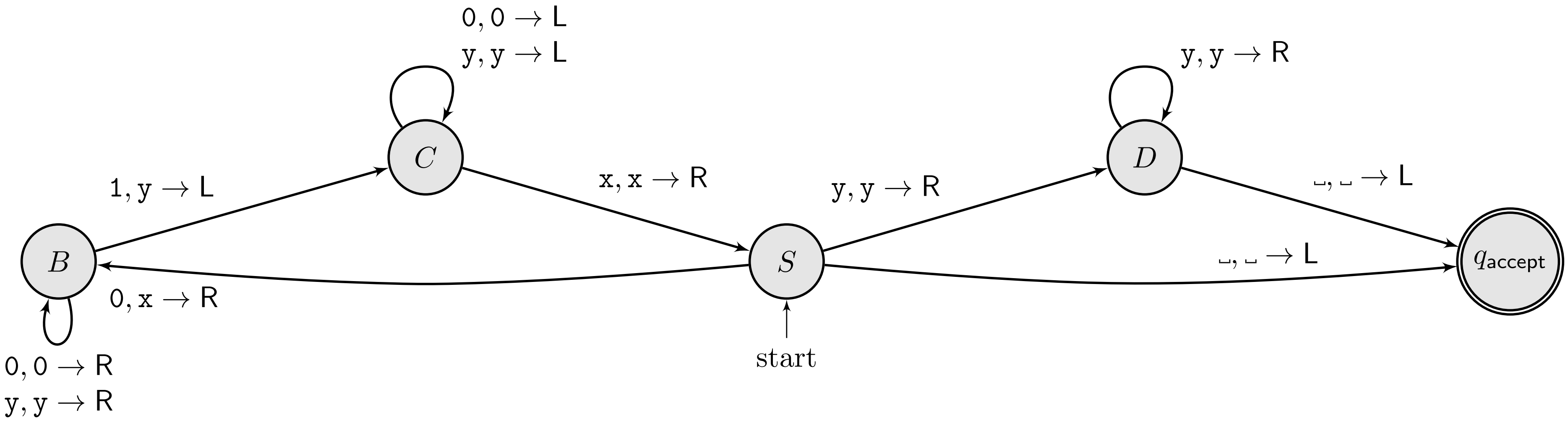
$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

> Mark 0 seek and mark 1 and cycle back.

- Start (S): `if 0 {write X; move right; goto B}; if Y {skip right; goto D}`
- Seek 0 (B): `while 1 or X {skip right};if 1 {write Y; move right; goto C}`
- Seek 1 (C): `while 0 or Y {skip left};if X {skip; move right; goto S}`
- Check valid (D): `while Y {skip right};if ␣ {skip;move right; goto accept}`

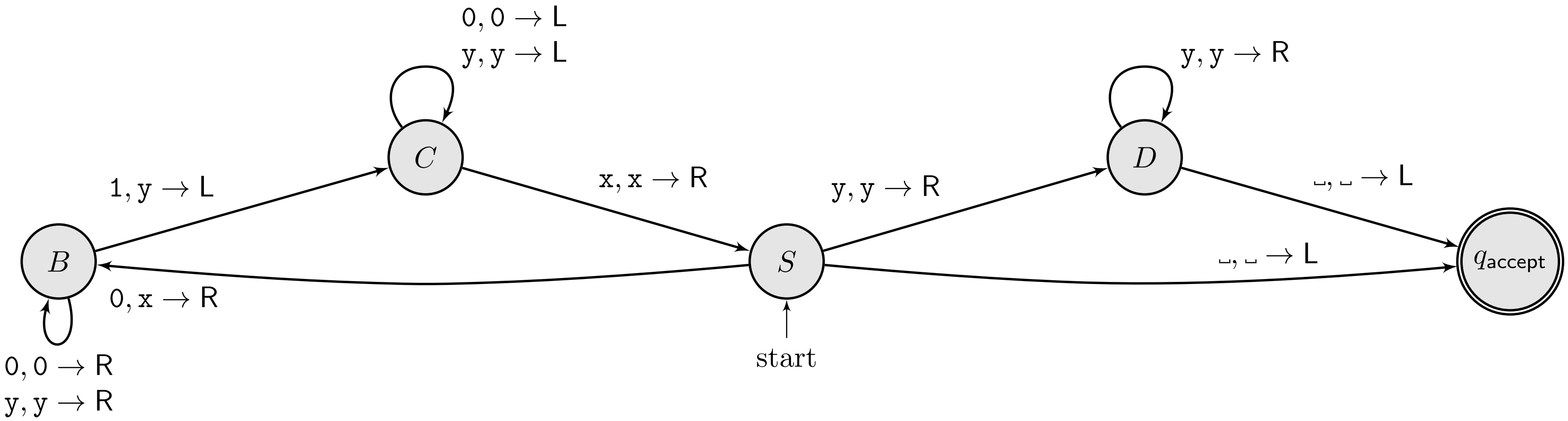
Tape	State	Rule
0011	S	read 0; write X; move right; goto B
X011	B	skip right while 1 or x; if 1 {write Y; move right; goto C}
X0Y1	C	skip left while 0 or y; if x {skip; move right; goto S}
X0Y1	S	read 0; write x; move right; goto B
XXY1	B	skip right while 1 or x; if 1 {write Y; move right; goto C}
XXYY	C	skip left while 0 or y; if x {skip; move right; goto S}
XXYY	S	read y; skip right; goto D
XXYY␣	D	read ␣, goto accept

Example 2



State	Tape
S	0011
B	X011
B	X011
C	X0Y1

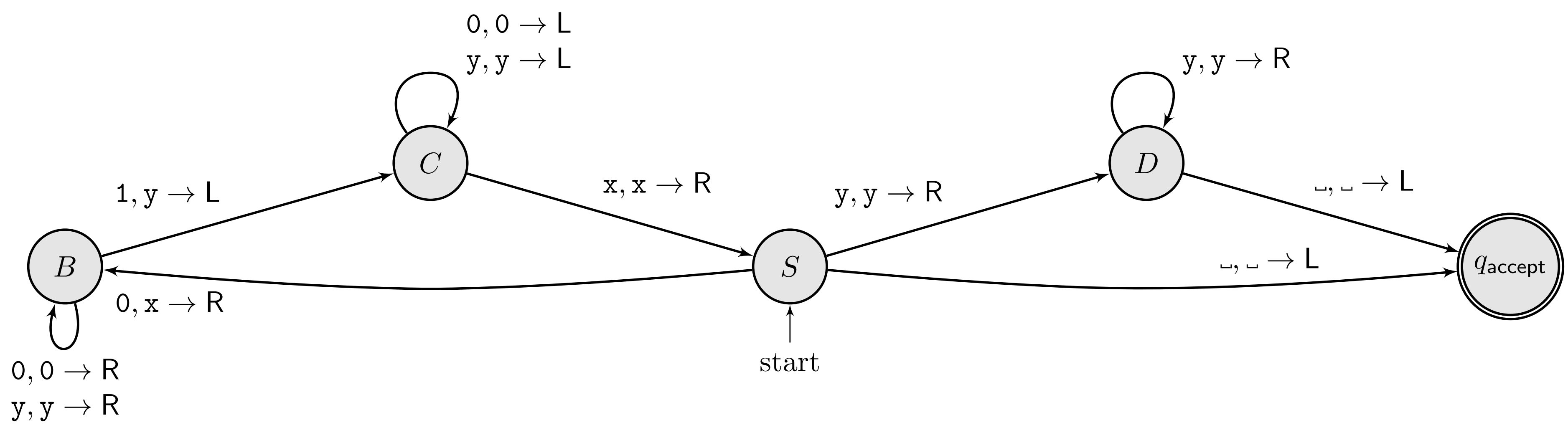
Example 2



State	Tape
S	0011
B	X011
B	X011
C	X0Y1

State	Tape
C	X0Y1
S	X0Y1
B	XXY1
B	XXY1

Example 2

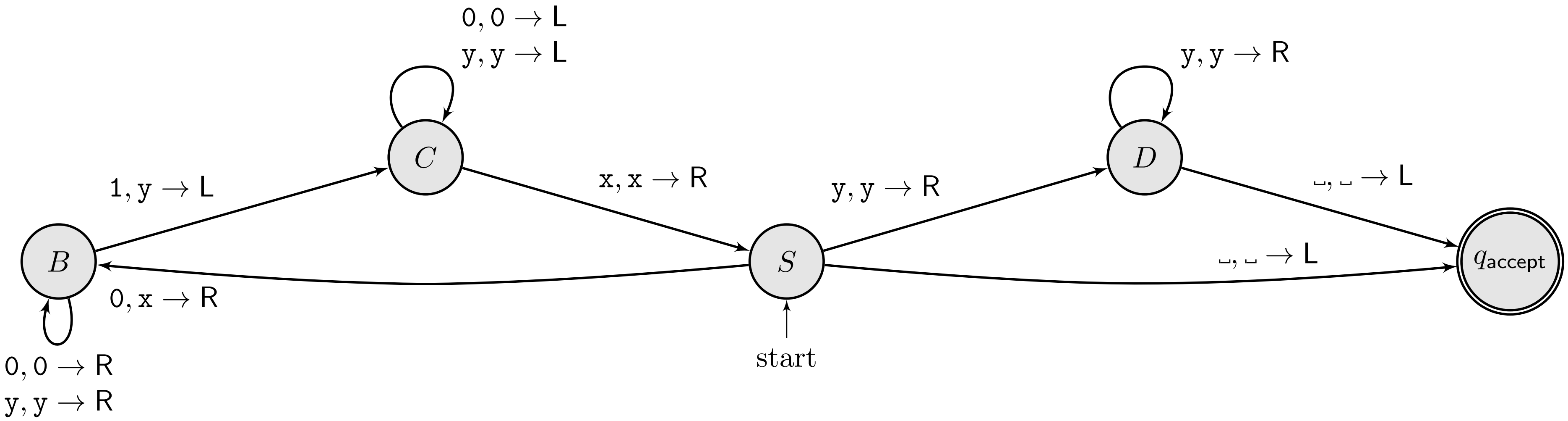


State	**Tape**
S	0011
B	X011
B	X011
C	X0Y1

State	**Tape**
C	X0Y1
S	X0Y1
B	XXY1
B	XXY1

State	**Tape**
C	XXYY
C	XXYY
S	XXYY
D	XXYY

Example 2



State	Tape
S	0011
B	X011
B	X011
C	X0Y1

State	Tape
C	X0Y1
S	X0Y1
B	XXY1
B	XXY1

State	Tape
C	XXYY
C	XXYY
S	XXYY
D	XXYY

State	Tape
D	XXYY␣

Accept!

Simulate

Example 3

$L_3 = \{a^n b^n c^n \mid n \geq 0\}$

Example 3

$L_3 = \{a^n b^n c^n \mid n \geq 0\}$

- START: Skip marks **right** until we: i) read a; mark it; go to A; ii) read blank, accept.
- A: Skip **right** until read b; mark it; go to Bs
- B: Skip **right** until read c; mark it; go to Cs
- C: Skip **right** until read blank; move left; go to REWIND
- REWIND: Skip **left** until we reach blank, go to START

Simulate

Turing Machines

Formally

Turing Machines

Definition 3.3

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

1. Q set of states
2. Σ input alphabet not containing the blank symbol ␣
3. Γ the tape alphabet, where $␣ \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ transition function
5. $q_0 \in Q$ is the start state
6. q_{accept} is the accept state
7. q_{reject} is the reject state ($q_{reject} \neq q_{accept}$)

Configuration

> A configuration is a snapshot of a computation. That is, it contains all information necessary to resume (or replay) a computation from any point in time.

A configuration consists of

- the tape
- the head of the tape
- the current state

Configuration

Textual notation

> We write the table and place the current state **before** (left of) where the head of the tape points to:

In the following example, the head points to position no.5, the tape is `0130045`, and the current state is q_3:

Recall example 1

State	***Tape***	***Configuration***
S	`0̲1110`	`S 01110`
B	`x1̲110`	
B	`xy1̲10`	
B	`xyy1̲0`	
B	`xyyy0̲`	
B	`xyyyx_`	

Fill in the configuration...

Example 1 configuration

State	Tape	Configuration
S	`01110`	`S 01110`
B	`x1110`	`x B 1110`
B	`xy110`	`xy B 110`
B	`xyy10`	`xyy B 10`
B	`xyyy0`	`xyyy B 0`
B	`xyyyx_`	`xyyyx B`

Configuration history

> The configuration history (sequence of configurations), describes all configurations from the initial state until a current state.

Definition

We say that C_1 **yields** C_2

Example

Configuration history
`S 01110`
`x B 1110`
`xy B 110`
`xyy B 10`
`xyyy B 0`
`xyyyx B`

More examples

- $L_5 = \{w\#w \mid w \in \{a,b\}^\star\}$
- $L_6 = \{w \mid w \text{ is a palindrome}\}$
- $L_7 = \{a^n b^{2n} \mid n \geq 0\}$