CSC 320 - Lecture 11

#PDA #context-free #languages #turing-machines #pumping-lemma #TM #decider #decidable-languages

---

Non Context-Free Languages

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

Idea. Stack not sufficient to keep track of more than one counter.

Pumping Lemma - Context-Free Languages

If A is a context-free language, then there is a number p (pumping length) such that: if s is any string in A of length at least p, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions...

1. for each $i \geq 0$, $uv^i xy^i z \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

Example

Use pumping lemma for context-free languages to show: $B = \{a^n b^n c^n \ n \geq 0\}$ is not context free.

- Proof by contradiction...
  - Assume B is context free
  - Let p be pumping length for B (guaranteed to exist by PL)
  - Choose a string $s \in B$, $|s| \geq p$ with the goal to show s is a counterexample to properties of PL for B being regular, i.e. we want to show...

s cannot be divided into five strings $s = uvxyz$ **satisfying**...

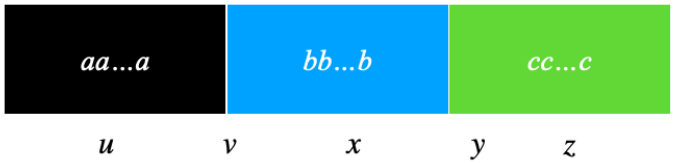
1. for each $i \geq 0$, $uv^i xy^i z \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

how

Goal. Show $B = \{a^n b^n c^n \; n \geq 0\}$ not context free.

Proof by Contradiction. Assume B context free.

- p be pumping length for B
- Let $s = a^p b^p c^p$
  - Then $s \in B$ and $|s| \geq p$
- Let's think about dividing s into u, v, w, x, y, z
- Because of 2) $vy \neq \varepsilon$, i.e.$v \neq \varepsilon$ or $y \neq \varepsilon$
- Because of 3) i.e. $|vxy| \geq p$, yielding these cases:



A. $vxy = a \ldots a \Rightarrow uv^2xy^2z = a^k b^p c^p$ with $k > p \notin B$
B. $vxy = a \ldots ab \ldots b \Rightarrow uv^2xy^2z = a^k b^l c^p$ with $k > p$ or $l > p \notin B$
C. $vxy = b \ldots b \Rightarrow uv^2xy^2z = a^p b^k c^p$ with $k > p \notin B$
D. $vxy = b \ldots bc \ldots c \Rightarrow uv^2xy^2z = a^p b^k c^l$ with $k > p$ or $l > p \notin B$
E. $vxy = c \ldots c \Rightarrow uv^2xy^2z = a^p b^p c^k$ with $k > p \notin B$

Note. Make sure you write down all the cases!! You need to disprove each case.

Example

$L = \{ww | w \in \{0,1\}^*\}$ is not context free.

Note. We don't know how to do split ww and we can't do that with the stack.

Proof by Contradiction: Assume that L is context free.

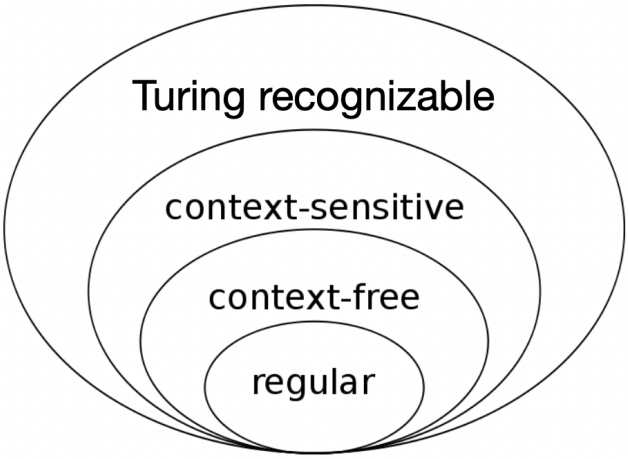
- p pumping length for L
- Choose $s = 0^p 1^p 0^p 1^p$
  - $0^p 1^p 0^p 1^p \in L$ for $s = ww$ with $w = 0^p 1^p$. $|s| = 4p \geq p$.
- Show there is not rewriting for s into $s = uvxyz$ such that PL conditions hold.
- Because of 3) vxy is of one of the following forms...

A. $vxy = 00 \ldots 0 \Rightarrow uv^2xy^2z = 0^k 1^p 0^l 1^p$ with either $k > p$ or $l > p$ (i.e., $0^k 1^p 0^p 1^p$ or $0^p 1^p 0^k 1^p$).
B. $vxy = 11 \ldots 1 \Rightarrow uv^2xy^2z = 0^p 1^k 0^p 1^l$ with either $k > p$ or $l > p$ (i.e., $0^p 1^k 0^p 1^p$ or $0^p 1^p 0^p 1^k$).

C. $vxy = 00\ldots011\ldots1 \Rightarrow uv^2xy^2z = 0^k1^l0^p1^p$ or $uv^2xy^2z = 0^p1^p0^k1^l$ with $k > p$ or $l > p$
D. $vxy = 11\ldots100\ldots0 \Rightarrow uv^2xy^2z = 0^p1^k0^l1^p$ with either $k > p$ or $l > p$

- **Conclusion**. There is not rewriting for s into $s = uvxyz$ and $uv^2xy^2z \in L$.

Chomsky Hierarchy



Context-Sensitive. Is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by context of terminal and nonterminal symbols. Context-Sensitive grammars are more general than Context-Free grammars, in the sense that there are languages that can be described by GSG but not by CFG.

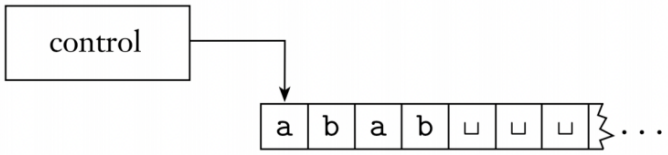
Turing Machines

Up Next. Church-Turing Thesis, Decidability, Reducibility, Complexity Theory (P vs NP).

The Turing Machine (TM) is much more powerful model than FA/PDA. First proposed by Alan Turing in 1936 (1912 - 1954). It is similar to finite automaton but it has unlimited & unrestricted memory. More accurate model of a general purpose computer. (A Turing Machine can do everything a (classical) computer can do).

What Does A Turing Machine Look Like?

Infinite Tape representing its unlimited memory. The **Tape Head** can read symbols, write symbols, and move around on the tape.



Computation. Initially the tape contains only input string and blank everywhere else. If TM needs to store information, it may write this information on tape. To read information that TM has written the machine can move its head back over it. The TM continues computing until it decides to produce an output. The outputs `accept` and `reject` are obtained by entering designated accepting and rejecting states. If TM does not enter accepting or rejecting state, computation will continue (i.e., infinite loop).

The `reject` state helps us by allowing us not to read the whole thing if not necessary.

Differences Between Finite Automata and Turing Machines

TM	FA
TM can both write on and read from tape	FA can only read from tape
Read-write head can move both left and right	Read head can move right
Tape is infinite	
Special states for rejecting and accepting take effect immediately (no need to finish reading the input)	Special state accepting takes effect only after finishing reading of input

How Does a TM Operate?

TM M for testing membership in language $B = \{w\#w | w \in \{0,1\}^*\}$. (**Note**. the $\#$ is not necessary, but it makes things easier).

M can to move back and forth over input and make marks on it. M accepts if its input is a member of B (**That Is**. Input consists of two identical string separated by symbol $\#$). M rejects otherwise.

Strategy. Go forth and back to the corresponding places on the two sides of the # and determine wether or not they match.

To keep track of which places correspond, M places marks on tape. M crosses off each pair of symbols as it is examined. If M crosses off all the symbols, then

everything is matched successfully and M accepts. If M discovers a mismatch, M rejects.

Example

$B = \{w\#w | w \in \{0,1\}^*\}$

Input. 011000#011000

- Start at the leftmost position.

0	1	1	0	0	0	#	0	1	1	0	0	0
↑												

- Cross out 0, move to the right and look for the next symbol right of #. If 0 then cross out and move left.

x	1	1	0	0	0	#	0	1	1	0	0	0
							↑					

x	1	1	0	0	0	#	x	1	1	0	0	0

- Find first symbol not crossed out.

x	1	1	0	0	0	#	x	1	1	0	0	0
	↑											

- Cross out 1, move to the right and look for the next symbol right of $\#$. If 1 then cross out and move left.

x	x	1	0	0	0	#	x	1	1	0	0	0
								↑				

x	x	1	0	0	0	#	x	x	1	0	0	0

- Find first symbol not crossed out.

x	x	1	0	0	0	#	x	x	1	0	0	0
		↑										

- Cross out 1, move to the right and look for the next symbol right of #. If 1 then cross out and move left.

x	x	x	0	0	0	#	x	x	1	0	0	0
									↑			

x	x	x	0	0	0	#	x	x	x	0	0	0

- Find first symbol not crossed out.

x	x	x	0	0	0	#	x	x	x	0	0	0
			↑									

- Cross out 0, move to the right and look for the next symbol right of #. If 0 then cross out and move left.

x	x	x	x	0	0	#	x	x	x	0	0	0
										↑		

x	x	x	x	0	0	#	x	x	x	x	0	0

- Etc.

x	x	x	x	x	x	#	x	x	x	x	x	x
					↑							

x	x	x	x	x	x	#	x	x	x	x	x	x
						↑						

- Accept

x	x	x	x	x	x	#	x	x	x	x	x	x	⊔
													↑

Note. When we encounter a blank we know we are finished reading.

Turing Machine – Formal Definition

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

- Q, Σ, Γ are finite sets
  - Q set of states
  - Σ input alphabet not containing blank symbol $\sqcup$
  - Γ tape alphabet; $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ transition function
- $q_0 \in Q$ start state
- $q_{accept} \in Q$ accept state
- $q_{reject} \in Q$ reject state with $q_{reject} \neq q_{accept}$

Note. Considering the definition (i.e., transition function) is the TM deterministic or non-deterministic?

Note. The transition function of TM is deterministic!

Note. You can have more than one accept state and reject state, but we will use one of each for simplicity.

Configuration of a Turing Machine

- Given TM M, a **configuration** of M consists of a description of...
  - Its current state
  - Its current tape contents
  - Its current head location
- For a state q and strings $u, v \in \Gamma^*$, we write $\mathbf{u, q, v}$ for the configuration where
  - The current state is q
  - The current tape content is uv
  - The current head location is the first symbol of v
  - The tape contains only blanks following the last symbol of v

	↓	
u		v

Computation of the Turing Machine

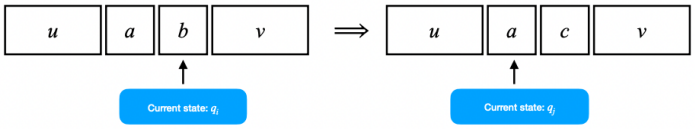
During computation changes occur in state, tape contents and head location.

We Define. Configuration(s) of TM and Computation Change in Configuration.

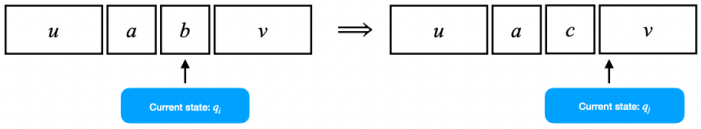
Configuration of a Turing Machine

For TM M, let $a, b, c \in \Gamma, u, v \in \Gamma^*$ and $q_i, q_j \in Q$, and let $ua\ q_i\ bv$ and $u\ q_j\ acv$ be two **configurations**. We say...

- $ua\ q_i\ bv$ **yields** $u\ q_j\ acv$ if
  - $\delta(q_i, b) = (q_j, c, L)$ (i.e., M moves leftward).



- $ua\ q_i\ bv$ **yields** $uac\ q_j\ v$ if
  - $\delta(q_i, b) = (q_j, c, R)$ (i.e., M moves rightward).

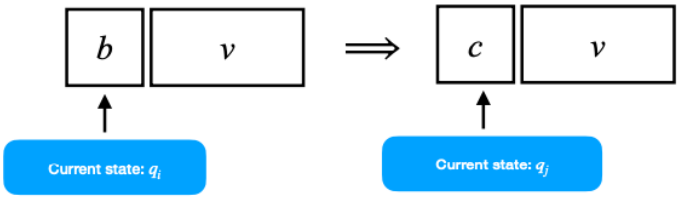


Note. M is in state q_i, reads symbol b, changes into state q_j, replaces b by c, and move its head one cell to the left.

Note. M is in state q_i, reads symbol b, changes into state q_j, replaces b by c, and move its head one cell to the right.

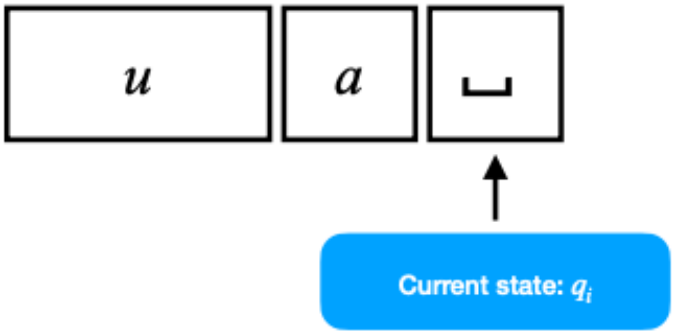
Special Cases

- **Left-Hand End** (Head is at leftmost cell)
  - Configuration $q_i\ bv$ yields configuration $q_j\ cv$ if transition is left-moving prevents M from going off left-hand end of tape (i.e., $\delta(q_i, b) = (q_j, c, L)$).

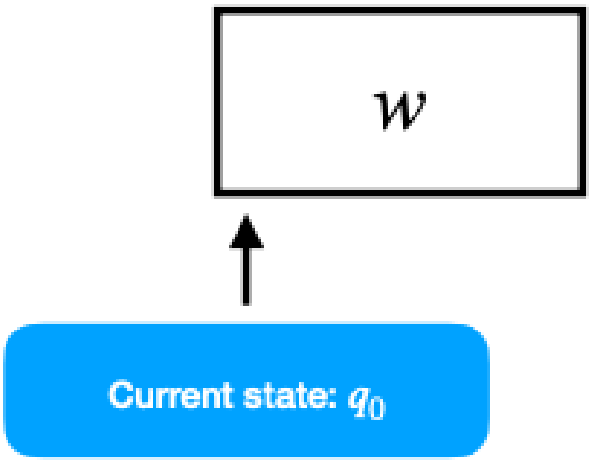


Note. We overwrite.

- $q_i\, bv$ yields $c\, q_j\, v$ if transition is right-moving (i.e., $\delta(q_i, b) = (q_j, c, R)$). (As expected).
- **Right-Hand End**
  - Configuration $ua\, q_i$ is equivalent to $ua\, q_i \sqcup$ because we assume that blanks follow part of tape represented in configuration.



- **Start Configuration**
  - $q_0 w$: input is w, M is in start state q_0 with its head at leftmost position on the tape.



- **Halting Configurations**
  - **Accepting Configuration** the state is q_{accept}
  - **Rejecting Configuration** state is q_{reject}

Computation of a Turing Machine

TM M accepts input w if a sequence of configurations $C_1, C_2 \ldots C_k$ exists, where...

- C_i is start configuration of M on input w
- each C_i yields C_{i+1}, and
- C_k is an accepting configuration

Language $L(M)$ of M, or the **language recognized** by M, is the collection of all strings that M accepts.

Possible Outcomes of a Computation

Possible outcomes of a TM on input w

- Accept, Reject, Loop (Machine also fails to accept w).

A TM that halts on every input is called a **decider**.

Turing Machines & Their Languages

If a language L is recognized by some TM, we call L **Turing-Recognizable**.

We call a language L **Turing-Decidable** (or **decidable**) if there exists a decider M that recognizes L. (We also say that M **decides** L).

Note. Every Turing-Decidable language is also Turing-Recognizable. (It does not always work the other way around). The Turing-Decidable is a special case of TM.

Question. Can a decider enter an infinite loop? NO!

Decidable Languages - Informal Description

Let $L = \{0^{2^n} | n \geq 0\}$. We describe M that decides L.

On input string $w \in \{0^*\}$

1. Sweep left to right, crossing off every other 0
2. If in step 1 the tape contained exactly one 0: accept
3. If in step 1 the tape contained more than one 0, and the number of zeros was odd: reject
4. Otherwise return the head to the left-hand end of the tape
5. Go to step 1.

Note. 0, 00, 0000, 00000000 are in the language.

Formal Description of a TM M

$L = \{0^{2^n} | n \geq 0\}$

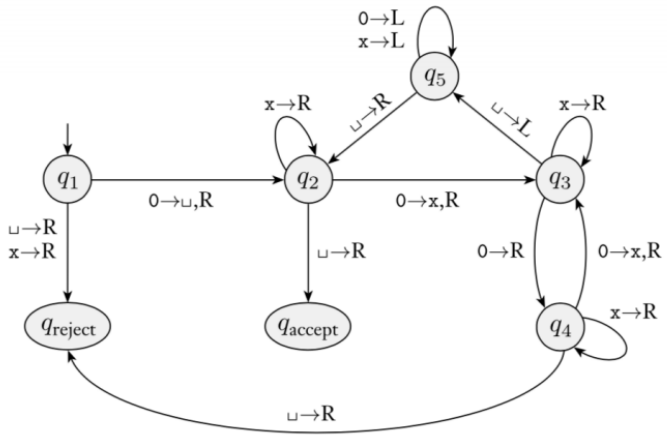
Let $M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ with...

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$ where...

    - q_1 is start state
    - q_{accept} is accept state and q_{reject} is reject state
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$

- δ transition function

State Diagram



Example – 0000

State q_1

↓			
0	0	0	0

- First 0 replaced by ⊔ to mark left-hand.

State q_2

	↓		
⊔	0	0	0

- Cross out every other 0

State q_3

		↓	
⊔	x	0	0

State q_4

			↓
⊔	x	0	0

State q_3

				↓
⊔	x	0	x	⊔

- Move to the left

State q_5

			↓	
⊔	x	0	x	⊔

State q_5

		↓		
⊔	x	0	x	⊔

State q_5

	↓			
⊔	x	0	x	⊔

State q_5

↓				
⊔	x	0	x	⊔

- Look for remaining 0s

State q_2

	↓			
⊔	x	0	x	⊔

- Cross out every other remaining 0

State q_2

		↓		
⊔	x	0	x	⊔

State q_3

			↓	
⊔	x	x	x	⊔

- Move to the left

State q_3

				↓
⊔	x	x	x	⊔

State q_5

			↓	
⊔	x	x	x	⊔

State q_5

		↓		
⊔	x	x	x	⊔

State q_5

	↓			
⊔	x	x	x	⊔

State q_5

↓				
⊔	x	x	x	⊔

- Look for remaining 0s

State q_2

	↓			
⊔	x	x	x	⊔

State q_2

		↓		
⊔	x	x	x	⊔

State q_2

			↓	
⊔	x	x	x	⊔

- No 0 left

State q_2

				↓
⊔	x	x	x	⊔

- Accept

State q_{accept}

					↓
⊔	x	x	x	⊔	⊔

Example – 00000

State q_1

↓				
0	0	0	0	0

- Firs 0 replaced by ⊔ to mark left-hand

State q_2

	↓			
⊔	0	0	0	0

- Cross out every other 0

State q_3

		↓		
⊔	x	0	0	0

State q_4

			↓	
⊔	x	0	0	0

State q_3

				↓
⊔	x	0	x	0

- Number if 0s encountered not even

State q_4

					↓
⊔	x	0	x	0	⊔

State q_{reject}

						↓
⊔	x	0	x	0	⊔	⊔

Example – 000000

Note. Do this as practice! It will be rejected!

Example

$L = \{w\#w | w \in \{0,1\}^*\}$

Decider $M = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$

- $Q = \{q_1, \ldots, q_8, q_{accept}, q_{reject}\}$
- $\Sigma = \{0, 1, \#\}$
- $\Gamma = \{0, 1, \#, x, \sqcup\}$

Describe TM L as high level description and state diagram.

Example

$L = \{\#x_1\#x_2\#\ldots\#x_l | each x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j\}$

Describe a TM that recognizes L. What kind of states would we want? (Next Assignment)?

Previous Lecture

Lecture10b

Next Lecture

Lecture12