CS310 : Automata Theory 2019

Lecture 23: Turing Machines and Computability

Instructor: S. Akshay

IITB, India

04-03-2019

Topics covered till date

I. Finite state automata

1. Deterministic finite-state automata (DFA)
2. Non-determinism, the subset construction, ϵ-transitions.
3. Regular expressions and equivalence with DFA
4. Regular languages and their properties
5. Pumping lemma
6. Myhill-Nerode and minimization

Topics covered till date

I. Finite state automata

1. Deterministic finite-state automata (DFA)
2. Non-determinism, the subset construction, ϵ-transitions.
3. Regular expressions and equivalence with DFA
4. Regular languages and their properties
5. Pumping lemma
6. Myhill-Nerode and minimization

II. Finite state automata with a stack

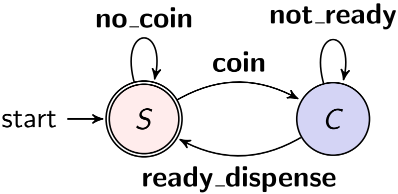
1. Pushdown automata (PDA)
2. Context-free grammars (CFG) and Parse trees
3. Equivalence of PDA and CFG, deterministic PDA
4. Chomsky normal form, Pumping lemma for CFLs
5. Applications of CFGs to parsing.

Today's lecture

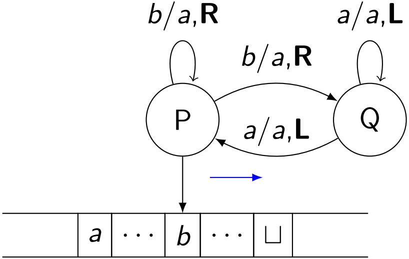
Turing Machines

From finite to infinite memory

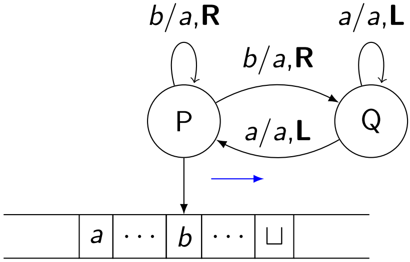
Finite instruction machine with finite memory (Finite State Automata)



Finite instruction machine with unbounded memory (Turing machine)

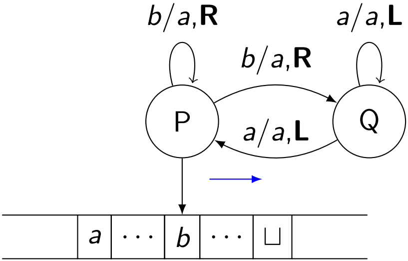


From NFA and PDA to Turing Machines



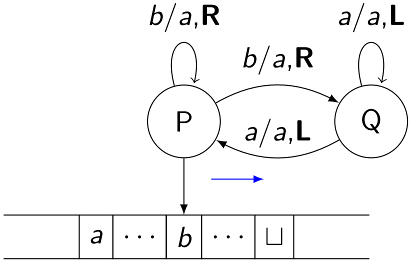
1. PDA are obtained from NFA by adding a stack
2. Turing machines are obtained from NFA by adding an infinite tape.

From NFA and PDA to Turing Machines



1. PDA are obtained from NFA by adding a stack
2. Turing machines are obtained from NFA by adding an infinite tape.
3. A Turing machine can both write on the tape and read from it.
4. The read-write head can move both to left and right.

From NFA and PDA to Turing Machines



1. PDA are obtained from NFA by adding a stack
2. Turing machines are obtained from NFA by adding an infinite tape.
3. A Turing machine can both write on the tape and read from it.
4. The read-write head can move both to left and right.
5. Initially all cells have special blank symbol, except where input is written.
6. Once accept/reject states are reached, computation terminates.

Example

Consider the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- Is it regular?
- How does one check if a given (really long) string is of this form?

Example

Consider the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- Is it regular?
- How does one check if a given (really long) string is of this form?

1. Scan the input once to make sure it does contain a single $\#$. Else reject.

Example

Consider the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- Is it regular?
- How does one check if a given (really long) string is of this form?

1. Scan the input once to make sure it does contain a single $\#$. Else reject.
2. Repeatedly do: Mark the first "unmarked" symbol from beginning and zig-zag across the tape to its corresponding position on the other side of $\#$ symbol to check if it is the same. If not, reject.

Example

Consider the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- Is it regular?
- How does one check if a given (really long) string is of this form?

1. Scan the input once to make sure it does contain a single $\#$. Else reject.
2. Repeatedly do: Mark the first "unmarked" symbol from beginning and zig-zag across the tape to its corresponding position on the other side of $\#$ symbol to check if it is the same. If not, reject.
3. Stop when there are no unmarked symbols remaining to the left of $\#$. If there are still symbols left on right, then reject. Else accept.

Example

Consider the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- Is it regular?
- How does one check if a given (really long) string is of this form?

1. Scan the input once to make sure it does contain a single $\#$. Else reject.
2. Repeatedly do: Mark the first "unmarked" symbol from beginning and zig-zag across the tape to its corresponding position on the other side of $\#$ symbol to check if it is the same. If not, reject.
3. Stop when there are no unmarked symbols remaining to the left of $\#$. If there are still symbols left on right, then reject. Else accept.

In general, we use the finite control to process the symbols that are being read on tape. How does one formalize this?

Formal definition of a Turing Machine

Definition

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

1. Q is a finite set of states,
2. Σ is a finite input alphabet,
3. Γ is a finite tape alphabet where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. q_0 is the start state,
5. q_{acc} is the accept state,
6. q_{rej} is the reject state,
7. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.

Formal definition of a Turing Machine

Definition

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

1. Q is a finite set of states,
2. Σ is a finite input alphabet,
3. Γ is a finite tape alphabet where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. q_0 is the start state,
5. q_{acc} is the accept state,
6. q_{rej} is the reject state,
7. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.

For a $q \in Q$, $a \in \Gamma$ if $\delta(q, a) = (q', b, L)$, then

- q' is the new state of the machine,
- b is the letter which replaces a on the tape,
- the head moves to Left of the current position.

Example: TM for $B = \{w\#w \mid w \in \{0,1\}^*\}$?

Example: TM for $B = \{w\#w \mid w \in \{0,1\}^*\}$?

From Sipser's book: What is $Q, \Sigma, \Gamma, \delta$, etc.?