CS310 : Automata Theory 2019

Lecture 34: Linear Bounded Automata

Instructor: S. Akshay

IITB, India

01-04-2019

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis
2. Undecidability
   (i) A proof technique by diagonalization
   (ii) Via reductions
   (iii) Rice's theorem

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis
2. Undecidability
   (i) A proof technique by diagonalization
   (ii) Via reductions
   (iii) Rice's theorem
3. Applications: showing (un)decidability of other problems
   (i) A string matching problem: Post's Correspondance Problem
   (ii) A problem for compilers: Unambiguity of Context-free languages

Another restriction of Turing machines

Definition

A *linear bounded automaton (LBA)* is a TM where the tape head cannot move off the portion of the tape containing the input.

Another restriction of Turing machines

Definition

A *linear bounded automaton (LBA)* is a TM where the tape head cannot move off the portion of the tape containing the input.

- Thus, a limited amount of memory.
- But we can use larger tape alphabet!

Another restriction of Turing machines

Definition

A *linear bounded automaton (LBA)* is a TM where the tape head cannot move off the portion of the tape containing the input.

- Thus, a limited amount of memory.
- But we can use larger tape alphabet! Does this help?

Another restriction of Turing machines

Definition

A *linear bounded automaton (LBA)* is a TM where the tape head cannot move off the portion of the tape containing the input.

- Thus, a limited amount of memory.
- But we can use larger tape alphabet! increases memory only by a constant factor.

Another restriction of Turing machines

Definition

A *linear bounded automaton (LBA)* is a TM where the tape head cannot move off the portion of the tape containing the input.

- Thus, a limited amount of memory.
- But we can use larger tape alphabet! increases memory only by a constant factor.
- given input of length n, memory available is a linear fn of n

Linear bounded automata (LBA)

How powerful are LBA? What do they capture?

- regular languages?
- context free languages?
- decidable languages?
- All languages?

Linear bounded automata (LBA)

How powerful are LBA? What do they capture?

- regular languages?
- context free languages?
- decidable languages?
- All languages?

Chocolate problem: Give an example of a language which is decidable, but not accepted by any LBA.

Linear bounded automata (LBA)

How powerful are LBA? What do they capture?

- regular languages?
- context free languages?
- decidable languages?
- All languages?

Chocolate problem: Give an example of a language which is decidable, but not accepted by any LBA.

What about the acceptance and emptiness problems?

- $A_{LBA} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts string } w\}$.
- $E_{LBA} = \{\langle M\rangle \mid M \text{ is an LBA with } L(M) = \emptyset\}$.

Are they decidable?

How powerful are LBA?

- $A_{LBA} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts string } w\}$.
- $E_{LBA} = \{\langle M\rangle \mid M \text{ is an LBA with } L(M) = \emptyset\}$.

How powerful are LBA?

- $A_{LBA} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts string } w\}$.
- $E_{LBA} = \{\langle M\rangle \mid M \text{ is an LBA with } L(M) = \emptyset\}$.

Pop Quiz

1. Let M be an LBA with $|Q| = m, |\Gamma| = r$, with input length n. How many distinct configurations D of M are possible?
2. Can you simulate an LBA with a halting TM, i.e., is A_{LBA} decidable?
3. Can you describe a reduction from A_{TM} to E_{LBA}, i.e., is E_{LBA} undecidable?

Two more proofs

Decidability of A_{LBA}

Two more proofs

Decidability of A_{LBA}

- Simulate LBA M on w for D steps (unless it halts earlier).
- If it accepts or rejects, do the same.
- If run does not stop in D steps, declare reject (loop detected)!

Two more proofs

Decidability of A_{LBA}

- Simulate LBA M on w for D steps (unless it halts earlier).
- If it accepts or rejects, do the same.
- If run does not stop in D steps, declare reject (loop detected)!

Claim: A_{LBA} accepts w iff it accepts w in at most D steps.

- One direction trivial.

Two more proofs

Decidability of A_{LBA}

- Simulate LBA M on w for D steps (unless it halts earlier).
- If it accepts or rejects, do the same.
- If run does not stop in D steps, declare reject (loop detected)!

Claim: A_{LBA} accepts w iff it accepts w in at most D steps.

- One direction trivial.
- For the other, if M on w didn't stop in D steps, by PHP there must be a config visited twice, i.e., a loop. hence M cannot accept w.

Two more proofs

Decidability of A_{LBA}

- Simulate LBA M on w for D steps (unless it halts earlier).
- If it accepts or rejects, do the same.
- If run does not stop in D steps, declare reject (loop detected)!

Claim: A_{LBA} accepts w iff it accepts w in at most D steps.

- One direction trivial.
- For the other, if M on w didn't stop in D steps, by PHP there must be a config visited twice, i.e., a loop. hence M cannot accept w.

Undecidability of E_{LBA}

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$
- Idea: B accepts ip x iff x is a string describing *sequence of accepting computations* of M on w.

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$
- Idea: B accepts ip x iff x is a string describing *sequence of accepting computations* of M on w.
- Break x into $\#C_1\#C_2 \ldots \#C_n\#$, and check if C_1 is start, C_n is acc and each transition is valid (how?).

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$
- Idea: B accepts ip x iff x is a string describing *sequence of accepting computations* of M on w.
- Break x into $\#C_1\#C_2 \ldots \#C_n\#$, and check if C_1 is start, C_n is acc and each transition is valid (how?).
- i.e., C_i and C_{i+1} are same on all except positions near the head. And they are correctly updated acc transition of M. Use markers to keep track of positions.

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$
- Idea: B accepts ip x iff x is a string describing *sequence of accepting computations* of M on w.
- Break x into $\#C_1\#C_2\ldots\#C_n\#$, and check if C_1 is start, C_n is acc and each transition is valid (how?).
- Now, show that $w \in L(M)$ iff $L(B) \neq \emptyset$

Two more proofs

Decidability of A_{LBA}

Undecidability of E_{LBA}

- Reduction from A_{TM}: define map from TM (M, w) to LBA B, s.t., $w \in L(M)$ iff $L(B) \neq \emptyset$
- Idea: B accepts ip x iff x is a string describing *sequence of accepting computations* of M on w.
- Break x into $\#C_1\#C_2 \ldots \#C_n\#$, and check if C_1 is start, C_n is acc and each transition is valid (how?).
- Now, show that $w \in L(M)$ iff $L(B) \neq \emptyset$
- Thus, non-emptiness is undecidable. What about emptiness?