Computability and Complexity

Lecture 5

Reductions
Undecidable problems from language theory
Linear bounded automata

given by Jiri Srba

Reduction

Informal Definition

A problem *A is reducible to problem B* iff the solution to problem B can be used to solve the problem A.

This means that solving A cannot be more difficult than solving B.

Reduction

Informal Definition

A problem *A is reducible to problem B* iff the solution to problem B can be used to solve the problem A.

This means that solving A cannot be more difficult than solving B.

In the terms of computability theory:

A reduces to B means that

- if B is decidable then A is decidable too, and
- if A is undecidable then B is undecidable too.

The way we will use reducibility:

If we can reduce e.g. A_{TM} to some other problem (language) B, then B is undecidable.

Typical Proof Structure to Show Undecidability

We want to show that a language B is undecidable using the fact that we already know that the language A is undecidable.

Proof idea (proof by contradiction):

1. Assume for a while that we have a decider M_B for the language B.
2. Using M_B we construct a decider M_A for the language A.
3. Because we know that M_A cannot exist (A is undecidable), this implies that M_B cannot exist either.
4. Conclusion is that the language B is undecidable.

Typical Proof Structure to Show Undecidability

We want to show that a language B is undecidable using the fact that we already know that the language A is undecidable.

Proof idea (proof by contradiction):

1. Assume for a while that we have a decider M_B for the language B.
2. Using M_B we construct a decider M_A for the language A.
3. Because we know that M_A cannot exist (A is undecidable), this implies that M_B cannot exist either.
4. Conclusion is that the language B is undecidable.

In the proof we provided a reduction from an undecidable language A to the language B. Hence B is undecidable too.

The Language $HALT_{TM}$

Problem: "Given a TM M and a string w, does M halt on w?"

Language formulation

$$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid \ M \text{ is a TM and } M \text{ halts on the input } w \ \}$$

The Language $HALT_{TM}$

Problem: "Given a TM M and a string w, does M halt on w?"

Language formulation

$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid \; M \text{ is a TM and } M \text{ halts on the input } w \;\}$

Theorem

The language $HALT_{TM}$ is undecidable.

Proof: We reduce A_{TM} to $HALT_{TM}$.

Undecidability of $HALT_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid\ M$ is a TM and M accepts the input $w\ \}$

$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid\ M$ is a TM and M halts on the input $w\ \}$

1. By contradiction. Assume there is a decider R for $HALT_{TM}$.

Undecidability of $HALT_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid$ M is a TM and M accepts the input w $\}$
$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid$ M is a TM and M halts on the input w $\}$

1. By contradiction. Assume there is a decider R for $HALT_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:
1. Run R on $\langle M, w\rangle$.
2. If R rejected then $\underline{S \text{ rejects}}$.
3. If R accepted then simulate M on w.
4. If M accepted then $\underline{S \text{ accepts}}$, else
   If M rejected then $\underline{S \text{ rejects.}}$ "

Undecidability of $HALT_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts the input } w\}$

$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on the input } w\}$

1. By contradiction. Assume there is a decider R for $HALT_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:
1. Run R on $\langle M, w\rangle$.
2. If R rejected then $\underline{S \text{ rejects}}$.
3. If R accepted then simulate M on w.
4. If M accepted then $\underline{S \text{ accepts}}$, else
   If M rejected then $\underline{S \text{ rejects.}}$ "

3. So S is a decider for A_{TM}, but we know that S does not exist.

Undecidability of $HALT_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts the input } w\}$

$HALT_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on the input } w\}$

1. By contradiction. Assume there is a decider R for $HALT_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:
1. Run R on $\langle M, w\rangle$.
2. If R rejected then S rejects.
3. If R accepted then simulate M on w.
4. If M accepted then S accepts, else
   If M rejected then S rejects. "

3. So S is a decider for A_{TM}, but we know that S does not exist.
4. Conclusion: the decider R does not exist either and so $HALT_{TM}$ is undecidable. $\square$

The Language E_{TM}

Problem: "Given a TM M is the language of M empty?"

Language formulation

$$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid \; M \text{ is a TM such that } L(M) = \emptyset \;\}$$

The Language E_{TM}

Problem: "Given a TM M is the language of M empty?"

Language formulation

$$E_{TM} \stackrel{\text{def}}{=} \{\langle M \rangle \mid \ M \text{ is a TM such that } L(M) = \emptyset \ \}$$

Theorem

The language E_{TM} is undecidable.

Proof: We reduce A_{TM} to E_{TM}.

Undecidability of E_{TM} by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid\ M \text{ is a TM and } M \text{ accepts the input } w\ \}$

$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid\ M \text{ is a TM such that } L(M) = \emptyset\ \}$

1. By contradiction. Assume there is a decider R for E_{TM}.

Undecidability of E_{TM} by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid \ M \text{ is a TM and } M \text{ accepts the input } w\ \}$

$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid \ M \text{ is a TM such that } L(M) = \emptyset\ \}$

1. By contradiction. Assume there is a decider R for E_{TM}.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:

1. Using M and w construct the following TM M_1

   $M_1 =$ "On input x:

   1. If $x \neq w$ then $\underline{M_1 \text{ rejects}}$
   2. If $x = w$ then simulate M on w.

      If M accepted then $\underline{M_1 \text{ accepts}}$,

      if M rejected then $\underline{M_1 \text{ rejects}}$."

2. Run R on $\langle M_1\rangle$.
3. If R accepted, $\underline{S \text{ rejects}}$; if R rejected, $\underline{S \text{ accepts}}$. "

Undecidability of E_{TM} by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts the input } w\}$

$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = \emptyset\}$

1. By contradiction. Assume there is a decider R for E_{TM}.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:

1. Using M and w construct the following TM M_1

   $M_1 =$ "On input x:

   1. If $x \neq w$ then M_1 rejects
   2. If $x = w$ then simulate M on w.

      If M accepted then M_1 accepts,

      if M rejected then M_1 rejects."
2. Run R on $\langle M_1\rangle$.
3. If R accepted, S rejects; if R rejected, S accepts. "

3. So S is a decider for A_{TM}, but we know that S does not exist.

Undecidability of E_{TM} by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts the input } w\}$

$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = \emptyset\}$

1. By contradiction. Assume there is a decider R for E_{TM}.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:

1. Using M and w construct the following TM M_1

   $M_1 =$ "On input x:

   1. If $x \neq w$ then $\underline{M_1 \text{ rejects}}$
   2. If $x = w$ then simulate M on w.
      If M accepted then $\underline{M_1 \text{ accepts}}$,
      if M rejected then $\underline{M_1 \text{ rejects}}$."
2. Run R on $\langle M_1\rangle$.
3. If R accepted, $\underline{S \text{ rejects}}$; if R rejected, $\underline{S \text{ accepts}}$. "

3. So S is a decider for A_{TM}, but we know that S does not exist.
4. Conclusion: R cannot exist and hence E_{TM} is undecidable. □

The Language EQ_{TM}

Problem: "Given two TMs M_1 and M_2, do they recognize the same language?"

Language formulation

$$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid \ M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\ \}$$

The Language EQ_{TM}

Problem: "Given two TMs M_1 and M_2, do they recognize the same language?"

Language formulation

$$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$$

Theorem

The language EQ_{TM} is undecidable.

Proof: We reduce E_{TM} to EQ_{TM}.

Undecidability of EQ_{TM} by Reduction from E_{TM}

$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$
$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = \emptyset\}$

1. By contradiction. Assume there is a decider R for EQ_{TM}.

Undecidability of EQ_{TM} by Reduction from E_{TM}

$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid\ M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\ \}$
$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid\ M \text{ is a TM such that } L(M) = \emptyset\ \}$

1. By contradiction. Assume there is a decider R for EQ_{TM}.
2. Using the decider R, we construct a decider S for E_{TM}:

$S =$ " On input $\langle M\rangle$:
1. Let M_1 be a TM that rejects all inputs.
2. Run R on $\langle M, M_1\rangle$.
3. If R accepted, S accepts; if R rejected, S rejects. "

Undecidability of EQ_{TM} by Reduction from E_{TM}

$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$
$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = \emptyset\}$

1. By contradiction. Assume there is a decider R for EQ_{TM}.
2. Using the decider R, we construct a decider S for E_{TM}:

$S =$ " On input $\langle M\rangle$:
1. Let M_1 be a TM that rejects all inputs.
2. Run R on $\langle M, M_1\rangle$.
3. If R accepted, <u>S accepts</u>; if R rejected, <u>S rejects</u>. "

3. So S is a decider for E_{TM}, but we know that S does not exist.

Undecidability of EQ_{TM} by Reduction from E_{TM}

$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$
$E_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = \emptyset\}$

1. By contradiction. Assume there is a decider R for EQ_{TM}.
2. Using the decider R, we construct a decider S for E_{TM}:

$S =$ " On input $\langle M\rangle$:
1. Let M_1 be a TM that rejects all inputs.
2. Run R on $\langle M, M_1\rangle$.
3. If R accepted, S accepts; if R rejected, S rejects. "

3. So S is a decider for E_{TM}, but we know that S does not exist.
4. Conclusion: R cannot exist and so EQ_{TM} is undecidable. □

Problem: "Given a TM M, is $L(M)$ regular?"

Language formulation

$$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid \; M \text{ is a TM s.t. } L(M) \text{ is regular } \}$$

Problem: "Given a TM M, is $L(M)$ regular?"

Language formulation

$$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M \rangle \mid \ M \text{ is a TM s.t. } L(M) \text{ is regular } \}$$

Theorem

The language $REGULAR_{TM}$ is undecidable.

Proof: We reduce A_{TM} to $REGULAR_{TM}$.

Undecidability of $REGULAR_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM such that } M \text{ accepts } w\}$

$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) \text{ is regular}\}$

1. By contradiction. Assume a decider R for $REGULAR_{TM}$.

Undecidability of $REGULAR_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM such that } M \text{ accepts } w\}$

$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) \text{ is regular}\}$

1. By contradiction. Assume a decider R for $REGULAR_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:
1. Construct the following TM M_1:
   $M_1 =$ "On input x:
   1. If x of the form 0^n1^n then $\underline{M_1 \text{ accepts}}$, else
   2. run M on w and $\underline{M_1 \text{ accepts}}$ iff M accepted."
2. Run R on $\langle M_1\rangle$.
3. If R accepted, $\underline{S \text{ accepts}}$; if R rejected, $\underline{S \text{ rejects}}$."

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM such that } M \text{ accepts } w\}$

$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) \text{ is regular}\}$

1. By contradiction. Assume a decider R for $REGULAR_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

S = " On input $\langle M, w\rangle$:
1. Construct the following TM M_1:
   M_1 = "On input x:
   1. If x of the form 0^n1^n then M_1 accepts, else
   2. run M on w and M_1 accepts iff M accepted."
2. Run R on $\langle M_1\rangle$.
3. If R accepted, S accepts; if R rejected, S rejects."

3. So S is a decider for A_{TM}, but we know that S does not exist.

Undecidability of $REGULAR_{TM}$ by Reduction from A_{TM}

$A_{TM} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is a TM such that } M \text{ accepts } w\}$

$REGULAR_{TM} \stackrel{\text{def}}{=} \{\langle M\rangle \mid M \text{ is a TM such that } L(M) \text{ is regular}\}$

1. By contradiction. Assume a decider R for $REGULAR_{TM}$.
2. Using the decider R, we construct a decider S for A_{TM}:

$S =$ " On input $\langle M, w\rangle$:
 1. Construct the following TM M_1:
  $M_1 =$ "On input x:
   1. If x of the form 0^n1^n then M_1 accepts, else
   2. run M on w and M_1 accepts iff M accepted."
 2. Run R on $\langle M_1\rangle$.
 3. If R accepted, S accepts; if R rejected, S rejects."

3. So S is a decider for A_{TM}, but we know that S does not exist.
4. So R cannot exist and $REGULAR_{TM}$ is undecidable. $\square$

Linear Bounded Automaton

Idea:

- Limit the memory (tape cells) of a TM.
- The available memory is proportional (by a constant factor) to the length of the input string.

Linear Bounded Automaton

Idea:

- Limit the memory (tape cells) of a TM.
- The available memory is proportional (by a constant factor) to the length of the input string.

Definition

Linear bounded automaton (LBA) is a restricted Turing machine M such that when M runs on any input string w, its head always stays within the first $|w|$ cells (should the head move to the right of the string, it stays at the end instead).

Linear Bounded Automaton

Idea:

- Limit the memory (tape cells) of a TM.
- The available memory is proportional (by a constant factor) to the length of the input string.

Definition

Linear bounded automaton (LBA) is a restricted Turing machine M such that when M runs on any input string w, its head always stays within the first $|w|$ cells (should the head move to the right of the string, it stays at the end instead).

Lemma

Let M be an LBA with q states and g tape symbols. When M is run on w then there are at most distinct configurations of M where $n = |w|$.

Linear Bounded Automaton

Idea:

- Limit the memory (tape cells) of a TM.
- The available memory is proportional (by a constant factor) to the length of the input string.

Definition

Linear bounded automaton (LBA) is a restricted Turing machine M such that when M runs on any input string w, its head always stays within the first $|w|$ cells (should the head move to the right of the string, it stays at the end instead).

Lemma

Let M be an LBA with q states and g tape symbols. When M is run on w then there are at most qng^n distinct configurations of M where $n = |w|$.

Acceptance Problem of Linear Bounded Automaton

Problem: "Does a given LBA accept a given string?"

Language formulation

$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid \ M \text{ is an LBA such that } M \text{ accepts } w\ \}$

Acceptance Problem of Linear Bounded Automaton

Problem: "Does a given LBA accept a given string?"

Language formulation

$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid$ M is an LBA such that M accepts w $\}$

Theorem

The language A_{LBA} is decidable.

Proof: The following algorithm decides A_{LBA}:

"On input $\langle M, w\rangle$ where M is an LBA and w a string:

1. Simulate M on w for at most $q \cdot |w| \cdot g^{|w|}$ steps where
   q is the number of states in M, and
   g the number of tape symbols in M.
2. If M accepted, then accept.
   If M rejected, then reject.
   If M did not halt (in $q \cdot |w| \cdot g^{|w|}$ steps), then reject."

□

Exam Questions

- Notion of reduction from problem A to problem B.
- Undecidability proofs using the reduction.
- Linear bounded automata: definition, decidability of the acceptance problem.