Undecidable Problems

Lecture 34
Section 12.2

Robb T. Koether

Hampden-Sydney College

Wed, Nov 16, 2016

Outline

The Membership Problem

Definition (The Membership Problem)

Given a Turing machine M and a string w, is $w \in L(M)$?

The Membership Problem

Definition (The Membership Problem)

Given a Turing machine M and a string w, is $w \in L(M)$?

Theorem (The Membership Problem)

The Membership Problem is undecidable.

The Membership Problem

Proof.

- Suppose that the Membership Problem is decidable.

□

The Membership Problem

Proof.

- Suppose that the Membership Problem is decidable.
- Then there is a Turing machine A that decides it.

□

The Membership Problem

Proof.

- Suppose that the Membership Problem is decidable.
- Then there is a Turing machine A that decides it.
- Consider the language L that we created earlier:

$$L = \{w_i \mid w_i \in L(M_i)\}.$$

□

The Membership Problem

Proof.

- Suppose that the Membership Problem is decidable.
- Then there is a Turing machine A that decides it.
- Consider the language L that we created earlier:

$$L = \{w_i \mid w_i \in L(M_i)\}.$$

- We found that L is recursively enumerable, but not recursive.

$\square$

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.

$\square$

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.

□

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.
  - Read a string w.

□

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.
  - Read a string w.
  - Feed M_1 and w into A.

□

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.
  - Read a string w.
  - Feed M_1 and w into A.
- M_2 will accept w if $w \in L(M_1)$.

$\square$

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.
  - Read a string w.
  - Feed M_1 and w into A.
- M_2 will accept w if $w \in L(M_1)$.
- M_2 will reject w if $w \notin L(M_1)$.

$\square$

The Membership Problem

Proof.

- Let M_1 be a Turing machine that accepts L.
- Build a Turing machine M_2 that does the following.
  - Read a string w.
  - Feed M_1 and w into A.
- M_2 will accept w if $w \in L(M_1)$.
- M_2 will reject w if $w \notin L(M_1)$.
- Thus, L is recursive, which is a contradiction.

$\square$

Outline

The Emptiness Problem

Definition (The Emptiness Problem)

Given a Turing machine M, is $L(M) = \varnothing$?

The Emptiness Problem

Definition (The Emptiness Problem)

Given a Turing machine M, is $L(M) = \varnothing$?

Theorem (The Emptiness Problem)

The Emptiness Problem is undecidable.

Proof.

- Suppose that the Emptiness Problem is decidable.

□

The Emptiness Problem

Proof.

- Suppose that the Emptiness Problem is decidable.
- Then there is a Turing machine E that decides it.

□

The Emptiness Problem

Proof.

- Suppose that the Emptiness Problem is decidable.
- Then there is a Turing machine E that decides it.
- We will reduce the Membership problem (undecidable) to the Emptiness Problem.

□

Proof.

- Build a Turing machine that does the following.

□

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.

□

The Emptiness Problem

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.
  - Modify M to a Turing machine M_w whose language is $L(M) \cap \{w\}$.

$\square$

The Emptiness Problem

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.
  - Modify M to a Turing machine M_w whose language is $L(M) \cap \{w\}$.
  - Note that

$$L(M) \cap \{w\} = \begin{cases} \Sigma^*, & w \in L(M) \\ \varnothing, & w \notin L(M) \end{cases}$$

$\square$

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.
  - Modify M to a Turing machine M_w whose language is $L(M) \cap \{w\}$.
  - Note that

$$L(M) \cap \{w\} = \begin{cases} \Sigma^*, & w \in L(M) \\ \varnothing, & w \notin L(M) \end{cases}$$

  - Feed M_w to E.

$\square$

The Emptiness Problem

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.
  - Modify M to a Turing machine M_w whose language is $L(M) \cap \{w\}$.
  - Note that

$$L(M) \cap \{w\} = \begin{cases} \Sigma^*, & w \in L(M) \\ \varnothing, & w \notin L(M) \end{cases}$$

  - Feed M_w to E.
- If E accepts M_w, then $w \notin L(M)$.

□

The Emptiness Problem

Proof.

- Build a Turing machine that does the following.
  - Read as input any Turing machine M and any input w.
  - Modify M to a Turing machine M_w whose language is $L(M) \cap \{w\}$.
  - Note that

$$L(M) \cap \{w\} = \begin{cases} \Sigma^*, & w \in L(M) \\ \varnothing, & w \notin L(M) \end{cases}$$

  - Feed M_w to E.
- If E accepts M_w, then $w \notin L(M)$.
- If E rejects M_w, then $w \in L(M)$.

□

The Emptiness Problem

Proof.

- Therefore, if the Empiness Problem is decidable, then the Membership Problem is also decidable.

□

The Emptiness Problem

Proof.

- Therefore, if the Empiness Problem is decidable, then the Membership Problem is also decidable.
- But the Membership Problem is not decidable.

□

The Emptiness Problem

Proof.

- Therefore, if the Empiness Problem is decidable, then the Membership Problem is also decidable.
- But the Membership Problem is not decidable.
- Therefore, the Empiness Problem is not decidable.

□

Outline

Functional Properties

Definition (Functional Property)

A property of a Turing machine is called functional if for any two Turing machines M_1 and M_2 with the same language, either the both have the property or neither has the property.

- That is, the property depends only in the Turing machine's input and output, not on how it processes the input.

Rice's Theorem

Theorem (Rice's Theorem)

All nontrivial functional properties are undecidable.

Proof.

- Let P be a nontrivial functional property.

$\square$

Proof.

- Let P be a nontrivial functional property.
- Let M_1 be a Turing machine that rejects all inputs.

$\square$

Proof.

- Let P be a nontrivial functional property.
- Let M_1 be a Turing machine that rejects all inputs.
- That is, $L(M_1) = \varnothing$.

$\square$

Proof.

- Let P be a nontrivial functional property.
- Let M_1 be a Turing machine that rejects all inputs.
- That is, $L(M_1) = \varnothing$.
- Without loss of generality, assume that M_1 does not have property P.

$\square$

Proof.

- Let P be a nontrivial functional property.
- Let M_1 be a Turing machine that rejects all inputs.
- That is, $L(M_1) = \varnothing$.
- Without loss of generality, assume that M_1 does not have property P.
- (Otherwise, we reverse the roles of P and $\overline{P}$.)

$\square$

Proof.

- Let P be a nontrivial functional property.
- Let M_1 be a Turing machine that rejects all inputs.
- That is, $L(M_1) = \varnothing$.
- Without loss of generality, assume that M_1 does not have property P.
- (Otherwise, we reverse the roles of P and $\overline{P}$.)
- Let M_2 be a Turing machine that has property P.

□

Proof.

- Suppose that P is decidable.

$\square$

Proof.

- Suppose that P is decidable.
- Let D_P be a decider for P.

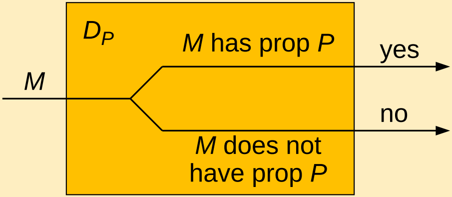
Proof.

- Suppose that P is decidable.
- Let D_P be a decider for P.
- That is, given any Turing machine M, D_P will decide whether M has property P.

□

Proof.

- Suppose that P is decidable.
- Let D_P be a decider for P.
- That is, given any Turing machine M, D_P will decide whether M has property P.



□

Proof.

- We will build a decider D_M for the Membership Problem.

$\square$

Proof.

- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.

□

Proof.

- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.
- Given M and w, build a Turing machine M_w that does the following:

$\square$

Proof.

- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.
- Given M and w, build a Turing machine M_w that does the following:
  - Given input x, simulate M on w.

□

Proof.

- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.
- Given M and w, build a Turing machine M_w that does the following:
  - Given input x, simulate M on w.
  - If M halts and rejects w, then M_w rejects x.

□

Proof.

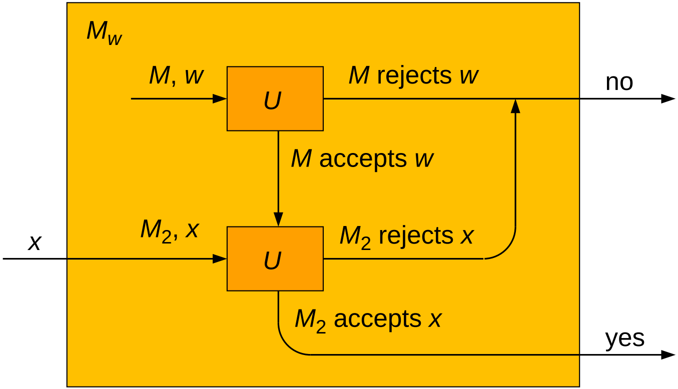
- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.
- Given M and w, build a Turing machine M_w that does the following:
  - Given input x, simulate M on w.
  - If M halts and rejects w, then M_w rejects x.
  - If M halts and accepts w, then M_w simulates M_2 on x.

□

Proof.

- We will build a decider D_M for the Membership Problem.
- But first we describe how to build another Turing machine M_w, which we will use as a module.
- Given M and w, build a Turing machine M_w that does the following:
  - Given input x, simulate M on w.
  - If M halts and rejects w, then M_w rejects x.
  - If M halts and accepts w, then M_w simulates M_2 on x.
  - If M loops, then (obviously) M_w loops.

□



M_w
M, w
U
M rejects w
no
M accepts w
x
M_2, x
U
M_2 rejects x
M_2 accepts x
yes

Proof.

- Therefore,

$$L(M_w) = \begin{cases} L(M_1), & w \notin L(M) \\ L(M_2), & w \in L(M) \end{cases}$$

$\square$

Proof.

- Therefore,

$$L(M_w) = \begin{cases} L(M_1), & w \notin L(M) \\ L(M_2), & w \in L(M) \end{cases}$$

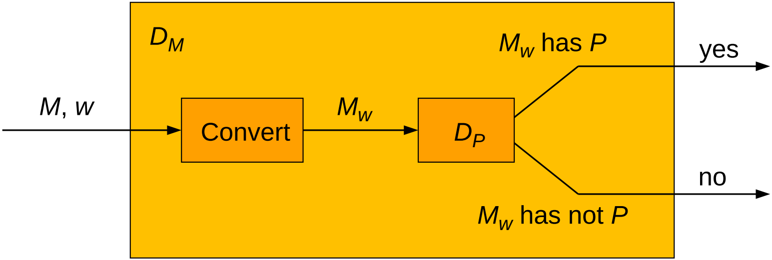
- That is, M_w has property P if and only if $w \in L(M)$.

$\square$

Proof.

- Now let D_A use D_M to decide whether M_w has property P:

$\square$


D_M
M_w has P
yes
M, w
Convert
M_w
D_P
no
M_w has not P

Proof.

- That is, D_A decides the Membership Problem, which is a contradiction.

$\square$

Proof.

- That is, D_A decides the Membership Problem, which is a contradiction.
- Therefore, P is undecidable.

$\square$

Outline

Assignment

Homework

- Section 12.2 Exercise 4.