CS420

Introduction to the Theory of Computation

Lecture 24: Undecidable problems

Tiago Cogumbreiro

Today we learn

- Decidability results
- Halting problem
- Emptiness for TM is undecidable

Section 4.2, 5.1

Decidability and Recognizability

Understanding the limits of decision problems

> ***Implementation:*** algorithm that answers a decision problem, that is algorithm says YES whenever decision problem says YES.

Concept	***Intuition***	***Example***
Recognizable	Can we implement the problem?	A_{TM}
Decidable	Can we implement the problem and prove it terminates?	A_{REX}
Undecidable	Impossible to say NO without looping	A_{TM}
Unrecognizable	Impossible to say YES and NO without looping	???

> Why is A_{TM} recognizable?

Decidability and Recognizability

Understanding the limits of decision problems

Concept	*YES without looping*	*NO without looping*
Recognizable	Possible	Maybe
Decidable	Possible	Possible
Undecidable	Maybe	Impossible
Unrecognizable	Impossible	Impossible

- Possible: we known an implementation ($\exists$)
- Impossible: no implementation is possible ($\forall$)

UMass Boston

Warmup

```
Require Import Turing.Turing.

Lemma decidable_to_recognizable:
  forall L,
  Decidable L ->
  Recognizable L.
Proof.
Admitted.

Lemma unrecognizable_to_undecidable:
  forall L,
   ~ Recognizable L ->
   ~ Decidable L.
Proof.
Admitted.
```

Corollary 4.23

$\overline{A_{TM}}$ is unrecognizable

Corollary 4.23: $\overline{A_{TM}}$ is unrecognizable

```
Lemma co_a_tm_not_recognizable:
  ~ Recognizable (compl A_tm).
```

Done in class...

Corollary 4.18

Some languages are unrecognizable

Corollary 4.18 Some languages are unrecognizable

Proof.

Corollary 4.18 Some languages are unrecognizable

Proof. An example of an unrecognizable language is: $\overline{A_{TM}}$

If L is decidable,

then $\overline{L}$ is decidable

On pen-and-paper proofs

THEOREM 4.22

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

In other words, a language is decidable exactly when both it and its complement are Turing-recognizable.

PROOF We have two directions to prove. First, if A is decidable, we can easily see that both A and its complement $\overline{A}$ are Turing-recognizable. Any decidable language is Turing-recognizable, and the complement of a decidable language also is decidable.

Proof of Theorem 4.22 Taken from the book.

> First, if A is decidable, we can easily see that both A and its complement A are Turing-recognizable.

- A is decidable, then A is recognizable by definition.
- A is decidable, then $\overline{A}$ is recognizable? **Why?**

> Any decidable language is Turing-recognizable,

- Yes, by definition.

> and the complement of a decidable language also is decidable.

- **Why?**

If L is decidable, then $\overline{L}$ is decidable

1. Let M decide L.
2. Create a Turing machine that negates the result of M.

```
Definition inv M w :=
  mlet b ← Call m w in Ret (negb b).
```

3. Show that `inv M` recognizes $\textbf{Inv}(L) = \{w \mid M \textbf{ rejects } w\}$
4. Show that the result of `inv M` for any word w is the negation of running M with m, where negation of accept is reject, reject is accept, and loop is loop.
5. The goal is to show that `inv M` recognizes $\overline{L}$ and is decidable.

> What about loops? If M loops on some word w, then `inv M` would also loop. How is does `inv M` recognize $\overline{L}$?

If L is decidable, then $\overline{L}$ is decidable

1. Let M decide L.
2. Create a Turing machine that negates the result of M.

```
Definition inv M w :=
  mlet b ← Call m w in Ret (negb b).
```

3. Show that `inv M` recognizes $\text{Inv}(L) = \{w \mid M \text{ rejects } w\}$
4. Show that the result of `inv M` for any word w is the negation of running M with m, where negation of accept is reject, reject is accept, and loop is loop.
5. The goal is to show that `inv M` recognizes $\overline{L}$ and is decidable.

> What about loops? If M loops on some word w, then `inv M` would also loop. How is does `inv M` recognize $\overline{L}$?

Recall that L is decidable, so M will never loop.

If L is decidable, then $\overline{L}$ is decidable

Continuation...

Part 1. Show that `inv M` recognizes $\overline{L}$

We must show that: If M decides L and `inv M` recognizes $\mathbf{Inv}(L)$, then `inv M` is decidable.

It is enough to show that if M decides L, then $\mathbf{Inv}(L) = \overline{L}$.

Show proof `inv_compl_equiv`.

Part 2. Show that `inv M` is a decider

Show proof `decides_to_compl`.

Chapter 5: Undecidability

$HALT_{\mathsf{TM}}$: Termination of TM

Will this TM halt given this input?

(The Halting problem)

$HALT_{\mathsf{TM}}$ is undecidable

Theorem 5.1: `HALT_TM` loops for some input

Set-based encoding

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

Function-based encoding

```
def HALT_TM(M,w):
  return M halts on w
```

Proof

Proof idea: Given Turing machine `acc`, show that `acc` decides A_{TM}.

```
def acc(M, w):
  if HALT_TM(M,w):
    return M(w)
  else:
    return False
```

$HALT_{\mathsf{TM}}$ is undecidable

Theorem 5.1: Proof overview

```
Definition acc (solve_HALT:input→prog) p :=
  let (M, w) := decode_mach_input p in
  mlet b ← solve_HALT p in (* HALT(M, w)*)
  if b then Call M w else Ret false.
```

Apply Thm 4.11 to (H) "`acc` decides A_{TM}" and reach a contradiction. To prove H:

1. Show that `acc` recognizes A_{TM}
2. Show that `acc` is decidable

E_{TM}: Emptiness of TM

(Is the language of this TM empty?)

Theorem 5.2: E_{TM} is undecidable

Set-based

$$E_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

Function-based

```
def E_TM(M):
  return L(M) == {}
```

Proof overview: show that acc decides A_{TM}

```
def build_M1(M,w):
  def M1(x):
    if x == w:
      return M accepts w
    else:
      return False
  return M1
```

```
def acc(M, w):
  b = E_TM(build_M1(M, w))
  return not b
```

- $w \in L(\mathtt{M1}) \iff \langle \mathtt{M1} \rangle \notin E_{TM}$
- $w \in L(\mathtt{M1}) \iff w \in L(M)$

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Goal: E_{TM} decidable implies A_{TM} decidable

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Goal: E_{TM} decidable implies A_{TM} decidable

Let D decide E_{TM}.

1. Show that `acc` recognizes A_{TM}

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Goal: E_{TM} decidable implies A_{TM} decidable

Let D decide E_{TM}.

1. Show that `acc` recognizes A_{TM}
   1. Show that $A_{\mathsf{TM}} = \mathbf{Acc}_D$ where $\mathbf{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$ (`e_tm_a_tm_spec`)

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Goal: E_{TM} decidable implies A_{TM} decidable

Let D decide E_{TM}.

1. Show that `acc` recognizes A_{TM}
   1. Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$ (`e_tm_a_tm_spec`)
   2. Show that `acc` recognizes Acc_D (`E_tm_A_tm_recognizes`)

Theorem 5.2: E_{TM} is undecidable

Proof follows by contradiction.

1. Show that E_{TM} decidable implies A_{TM} decidable.
2. Reach contradiction by applying Thm 4.11 to (1)

Goal: E_{TM} decidable implies A_{TM} decidable

Let D decide E_{TM}.

1. Show that `acc` recognizes A_{TM}
   1. Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$ (`e_tm_a_tm_spec`)
   2. Show that `acc` recognizes Acc_D (`E_tm_A_tm_recognizes`)
2. Show that `acc` is a decider (`decider_E_tm_A_tm`)

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathbf{Acc}_D$ where $\mathbf{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `not_empty_to_accept`

1. Show that: If $L(\mathtt{M1}_{M,w}) \neq \emptyset$, then M accepts w.

Theorem 5.2: E_{TM} is undecidable

> Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `not_empty_to_accept`

1. Show that: If $L(\mathtt{M1}_{M,w}) \neq \emptyset$, then M accepts w.
   - Case analysis on running M with input w:

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `not_empty_to_accept`

1. Show that: If $L(\mathtt{M1}_{M,w}) \neq \emptyset$, then M accepts w.
   - Case analysis on running M with input w:
     - Case (a) M accepts w: use assumption to conclude

Theorem 5.2: E_{TM} is undecidable

> Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `not_empty_to_accept`

1. Show that: If $L(\mathtt{M1}_{M,w}) \neq \emptyset$, then M accepts w.
   - Case analysis on running M with input w:
     - Case (a) M accepts w: use assumption to conclude
     - Case (b) M rejects w: we can conclude that $L(\mathtt{M1}_{M,w}) = \emptyset$ from (b)

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `not_empty_to_accept`

1. Show that: If $L(\mathtt{M1}_{M,w}) \neq \emptyset$, then M accepts w.
   - Case analysis on running M with input w:
     - Case (a) M accepts w: use assumption to conclude
     - Case (b) M rejects w: we can conclude that $L(\mathtt{M1}_{M,w}) = \emptyset$ from (b)
     - Case (c) M loops with w: same as above

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `accept_to_not_empty`

2. Show that: If M accepts w, then $L(\mathtt{M1}_{M,w}) \neq \emptyset$.

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathbf{Acc}_D$ where $\mathbf{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `accept_to_not_empty`

2. Show that: If M accepts w, then $L(\mathtt{M1}_{M,w}) \neq \emptyset$.
   1. Proof follows by contradiction: assume $L(\mathtt{M1}_{M,w}) = \emptyset$.

Theorem 5.2: E_{TM} is undecidable

> Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `accept_to_not_empty`

2. Show that: If M accepts w, then $L(\mathtt{M1}_{M,w}) \neq \emptyset$.
    1. Proof follows by contradiction: assume $L(\mathtt{M1}_{M,w}) = \emptyset$.
    2. We know that $\mathtt{M1}_{M,w}$ does not accept w from (2.1)

Theorem 5.2: E_{TM} is undecidable

> Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `accept_to_not_empty`

2. Show that: If M accepts w, then $L(\mathtt{M1}_{M,w}) \neq \emptyset$.
   1. Proof follows by contradiction: assume $L(\mathtt{M1}_{M,w}) = \emptyset$.
   2. We know that $\mathtt{M1}_{M,w}$ does not accept w from (2.1)
   3. To contradict 2.2, we show that $\mathtt{M1}_{M,w}$ **accepts** w

UMass Boston

Theorem 5.2: E_{TM} is undecidable

Part 1.1: Show that $A_{\mathsf{TM}} = \mathrm{Acc}_D$ where $\mathrm{Acc}_D = \{\langle M, w\rangle \mid L(\mathtt{M1}_{M,w}) \neq \emptyset\}$

Theorem `accept_to_not_empty`

2. Show that: If M accepts w, then $L(\mathtt{M1}_{M,w}) \neq \emptyset$.
   1. Proof follows by contradiction: assume $L(\mathtt{M1}_{M,w}) = \emptyset$.
   2. We know that $\mathtt{M1}_{M,w}$ does not accept w from (2.1)
   3. To contradict 2.2, we show that $\mathtt{M1}_{M,w}$ **accepts** w
      1. Since $x = w$ and (2.1), then $\mathtt{M1}_{M,w}$ **accepts** w