

# Introduction to Undecidability

Alice Gao

Lecture 23

Based on work by J. Buss, L. Kari, A. Lubiw, B. Bonakdarpour, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

# Outline

Introduction to Undecidability

Learning Goals

Introduction to Undecidability

Examples of Decision Problems

The Halting Problem is Undecidable

Revisiting the Learning Goals

# Learning Goals

By the end of this lecture, you should be able to:

Introduction to undecidability

- ▶ Define decision problem.
- ▶ Define decidable problem.
- ▶ Define undecidable problem.
- ▶ Prove that a decision problem is decidable by giving an algorithm to solve it.

The halting problem

- ▶ Describe the halting problem.
- ▶ Prove that the halting problem is undecidable.

# Exploring the limitation of computation

Most of CS focuses on what we CAN compute.

Are there problems that CANNOT be solved by a computer even with unlimited time and space?

The answer is yes. This was proved by Alan Turing in 1936.



# What is a computer program/algorithm?

In the old days, there were no electronic computers. A computer refers to a person who computes. (Watch Hidden Figures.)

Turing's idea of a "computer program" was a list of instructions that a person could follow.

For us, an algorithm could refer to any of the following:

- ▶ Racket, C, and C++ programs
- ▶ Turing machines
- ▶ High-level pseudo-code

# Decidable and undecidable problems

We focus on decision problems. **A decision problem** is a question with a yes/no answer.

**An algorithm solves a problem** iff it produces the correct output for the problem for every input.

A decision problem is

- ▶ **Decidable** iff there exists an algorithm that solves the problem.
- ▶ **Undecidable** iff there does not exist an algorithm that solves the problem.

## CQ 1 Examples of decision problems

**CQ 1:** Given a Propositional formula, is it satisfiable?

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.

## CQ 2 Examples of decision problems

**CQ 2:** Given a Predicate formula, is it valid? (Take a guess. All answers will be marked correct.)

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.



## CQ 3 Examples of decision problems

**CQ 3:** Given a positive integer, is it prime?

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.

## CQ 4 Examples of decision problems

**CQ 4:** Given a Hoare triple, is it satisfied under partial correctness? (Take a guess. All answers will be marked correct.)

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.

## CQ 5 Examples of decision problems

**CQ 5:** Given a Hoare triple, is it satisfied under total correctness?  
(Take a guess. All answers will be marked correct.)

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.

## CQ 6 Examples of decision problems

**CQ 6:** Given a program  $P$  and an input  $I$ , does  $P$  terminate when run with the input  $I$ ? (Take a guess. All answers will be marked correct.)

- (A) This problem is decidable.
- (B) This problem is undecidable.
- (C) I don't know.

# The Halting problem

The decision problem: Given a program  $P$  and an input  $I$ , will  $P$  halt when run with input  $I$ ?

- ▶ “Halts” means “terminates” or “does not get stuck.”
- ▶ One of the first known undecidable problems

The Halting problem is undecidable.

There does not exist an algorithm  $H$ , which gives the correct answer for the Halting problem for every program  $P$  and every input  $I$ .

Exercise: Translate the above statement into a Predicate formula.

# The Halting Problem is Undecidable

A proof by video

<https://www.youtube.com/watch?v=92WHN-pAFCs>

## Common questions about the video

- ▶ Why can we feed a program as an input to itself?  
We can convert any program to a string, then we can feed the string of the program to itself as input.
- ▶ What does the negator do?  
It negates the behaviour of the machine. If  $H$  predicts that the program halts, then the negator goes into an infinite loop and does not halt. If  $H$  predicts that the program does not halt, then the negator halts.  
The negator is designed to make  $H$  fail at its prediction task.
- ▶ Why do we need the photocopier?  
In the video,  $H$  takes two inputs. We need to make two copies of the input. In code, we do not need the photocopier. We simply need to call  $H(P,P)$ .

# The Halting Problem is Undecidable

Theorem: The Halting problem is undecidable.

Proof by contradiction.

Assume that there exists an algorithm  $H$ , which solves the Halting problem for every program and every input.

We will construct an algorithm  $X$ , which takes program  $P$  as input.

We will show that  $H$  gives the wrong answer when predicting whether the program  $X$  halts when run with input  $X$ . This contradicts the fact that  $H$  solves the Halting problem for every program and every input. Therefore,  $H$  does not exist.





# Revisiting the learning goals

By the end of this lecture, you should be able to:

Introduction to undecidability

- ▶ Define decision problem.
- ▶ Define decidable problem.
- ▶ Define undecidable problem.
- ▶ Prove that a decision problem is decidable by giving an algorithm to solve it.

The halting problem

- ▶ Describe the halting problem.
- ▶ Prove that the halting problem is undecidable.