Monash University
Faculty of Information Technology

Lecture 17
Undecidable Problems

Slides by Graham Farr (2013-2017).

FIT2014 Theory of Computation

Overview

- Halting Problem (or Entscheidungsproblem)
- Proof of its undecidability
- Mapping reductions
- Other undecidable problems

Halting Problem

Input: Turing machine ***P***, input ***x***

Question: If ***P*** is run with input ***x***, does it eventually **halt**?

Theorem

The Halting Problem is undecidable.

Proved by:

Alonzo Church (1936): lambda calculus

Alan Turing (1936-37): Turing machines

Halting Problem

Theorem

The Halting Problem is undecidable.

Proof is by contradiction, and includes a more elaborate version of the Liar Paradox:

"This sentence is false."

Halting Problem is Undecidable

Proof: (by contradiction)

Assume there is a Decider, ***D***, for the Halting Problem.

So it can tell, for any ***P*** and ***x***, whether or not ***P*** eventually halts after being given input ***x***.

So it can tell, for any ***P***, whether or not ***P*** eventually halts after being given input ***P*** !

Construct another program (Turing machine) ***E*** as follows …

Halting Problem is Undecidable (cont'd)

E

Input: ***P***
Use **D** to determine what happens if ***P*** runs on itself.
If **D** ***says, "P*** halts, with input ***P"***: loop forever.
If **D** ***says, "P*** loops forever, with input ***P"***: Stop.

What happens when ***E*** is given itself as input?

If ***E*** halts, for input ***E***: then ***E*** loops forever, for input ***E***.

If ***E*** loops forever, for input ***E***: then ***E*** halts, for input ***E***.

Contradiction! Q.E.D.

YouTube film of proof: https://www.youtube.com/watch?v=92WHN-pAFCs

Other Undecidable Problems

DIAGONAL HALTING PROBLEM

Input: Turing machine ***P***

Question: Does ***P*** eventually halt, for input ***P*** ?

Above proof already shows this.

HALT FOR INPUT ZERO

Input: Turing machine ***P***

Question: Does ***P*** eventually halt, for input 0?

HALT FOR INPUT ZERO is undecidable.

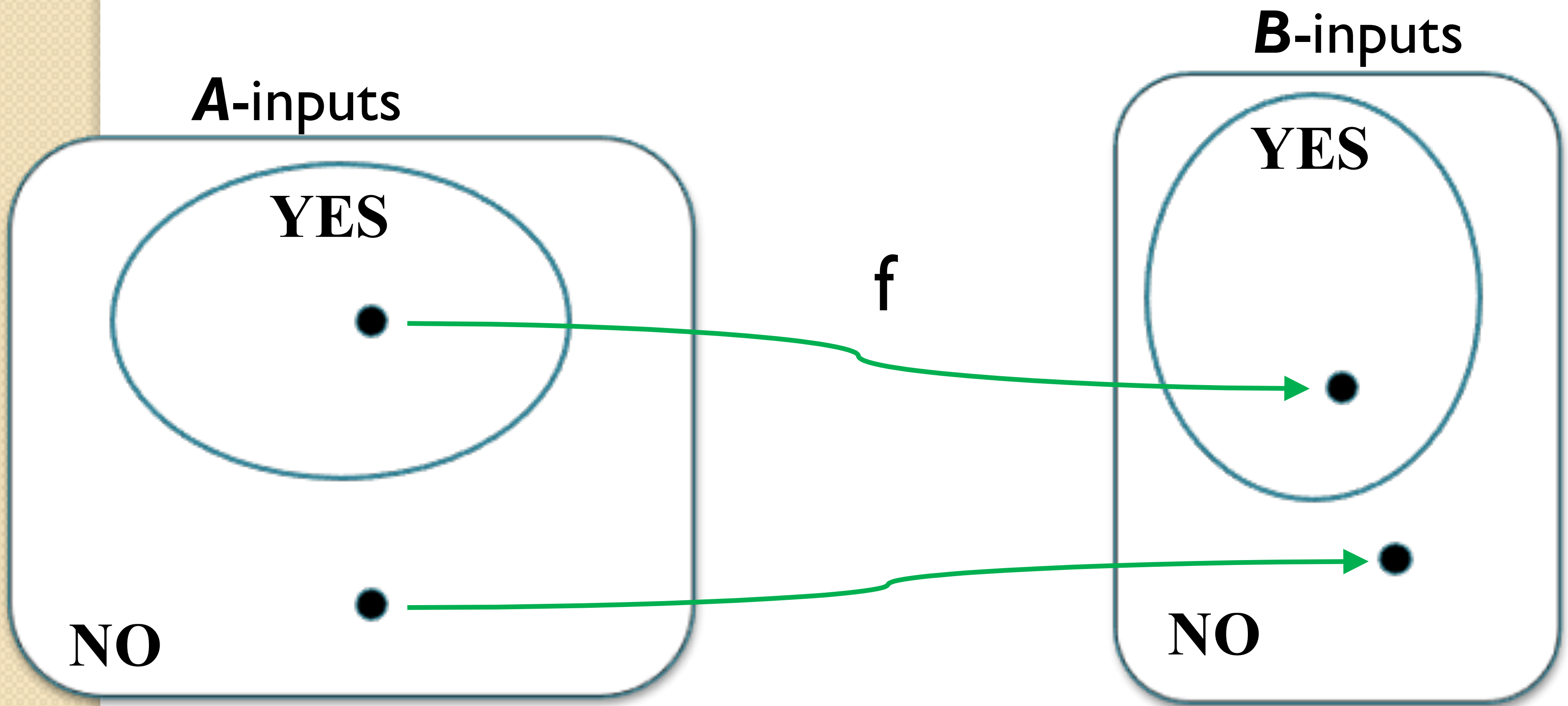
We'll prove this by **reduction** from the Diagonal Halting Problem.

Reductions

A **mapping reduction** from problem ***A*** to problem ***B*** is a computable function f : {***A***-inputs} ➔ {***B***-inputs} such that:

for every ***A***-input x,

x is a YES-input for ***A*** if and only if f(x) is a YES-input for ***B***



Reductions

If there is a mapping reduction f from ***A*** to ***B***, then:

- If ***B*** is decidable, then ***A*** is decidable.

  Decider for ***A***:

  1. Input: x.
  2. Compute f(x).
  3. Call the Decider for ***B*** to determine if whether f(x) is a YES-input or a NO-input for ***B***.
  4. *The same answer works for* ***A***.

- If ***A*** is undecidable, then ***B*** is undecidable.
  - contrapositive of previous statement.

Back to showing that HALT FOR INPUT ZERO is undecidable …

Let ***M*** be any program, which we regard as an input to the Diagonal Halting Problem.

Define ***M'*** as follows:

M'

Input: ***x***
Run ***M*** on input ***M***.

Observe:

- The construction of ***M'*** from ***M*** is computable.
- ***M*** halts on input ***M*** if and only if ***M'*** halts on input 0.

So, the function that sends ***M*** *to* ***M'*** is a **mapping reduction** from DIAGONAL HALTING PROBLEM to HALT FOR INPUT ZERO.

Therefore HALT FOR INPUT ZERO is undecidable.

Other Undecidable Problems

There's nothing special about zero, here.

So we get a whole lot of undecidability results.

For example:

HALT FOR INPUT 42

Input: Turing machine ***P***

Question: Does ***P*** eventually halt, for input 42?

Proof of undecidability is virtually identical to the previous one …

Use a mapping reduction.

HALT FOR INPUT 42 is undecidabe.

Proof:

Let ***M*** be any program, which we regard as an input to the Diagonal Halting Problem.

Define ***M'*** as follows:

M'

Input: ***x***
Run ***M*** on input ***M***.

Observe:

- The construction of ***M'*** from ***M*** is computable.
- ***M*** halts on input ***M*** if and only if ***M'*** halts on input **42**.

So, the function that sends ***M*** *to* ***M'*** is a **mapping reduction** from DIAGONAL HALTING PROBLEM to HALT FOR INPUT **42**.

Therefore HALT FOR INPUT **42** is undecidable. Q.E.D.

Other Undecidable Problems

ALWAYS HALTS
Input: Turing machine ***P***
Question: Does ***P*** always halt eventually, for any input?

Proof of undecidability is virtually identical to the previous one …

ALWAYS HALTS is undecidabe.

Proof:

Let ***M*** be any program, which we regard as an input to the Diagonal Halting Problem.

Define ***M'*** as follows:

M'

Input: ***x***
Run ***M*** on input ***M***.

Observe:

- The construction of ***M'*** from ***M*** is computable.
- ***M*** halts on input ***M*** if and only if ***M'*** halts **for all x**.

So, the function that sends ***M*** to ***M***' is a **mapping reduction** from DIAGONAL HALTING PROBLEM to **ALWAYS HALTS**.

Therefore **ALWAYS HALTS** is undecidable. Q.E.D.

Other Undecidable Problems

SOMETIMES HALTS

Input: Turing machine ***P***

Question: Is there some input for which ***P*** eventually halts?

SOMETIMES HALTS is undecidable.

Proof: by reduction from the Diagonal Halting Problem.

SOMETIMES HALTS is undecidabe.

Proof:

Let ***M*** be any program, which we regard as an input to the Diagonal Halting Problem.

Define ***M'*** as follows:

M'

Input: ***x***
Run ***M*** on input ***M***.

Observe:

- The construction of ***M'*** from ***M*** is computable.
- ***M*** halts on input ***M*** if and only if ***M'*** halts **for some x**.

So, the function that sends ***M*** *to* ***M'*** is a **mapping reduction** from DIAGONAL HALTING PROBLEM to **SOMETIMES HALTS**.

Therefore **SOMETIMES HALTS** is undecidable. Q.E.D.

Other Undecidable Problems

NEVER HALTS

Input: Turing machine ***P***

Question: Does ***P*** always loop forever, for any input?

NEVER HALTS is undecidable.

Proof: by a more general type of reduction, from SOMETIMES HALTS.

If D is a decider for NEVER HALTS, then switching the outputs YES and NO gives a decider for SOMETIMES HALTS. But we now know that SOMETIMES HALTS is undecidable. Contradiction. So NEVER HALTS is undecidable too. Q.E.D.

Other Undecidable Problems

Input: Turing machine ***P*** and ***Q***

Question: Do ***P*** and ***Q*** always both halt, or both loop?

I.e., for all ***x***, ***P*** halts on input ***x*** iff ***Q*** halts on input ***x***.

Input: Turing machine ***P***

Question: If ***P*** is run on the input "**What's the answer?**", does it output "**42**"?

Decidable or Undecidable?

Input: Turing machine ***P***, input ***x***.

Question: Does ***P*** accept ***x***?

Input: Turing machine ***P***, input ***x***, positive integer ***t***

Question: When ***P*** is run on ***x***, does it halt in $\leq$ ***t*** steps?

Input: Turing machine ***P***, positive integer ***s.***

Question: Does P have $\leq$ ***s*** states?

Input: Turing machine ***P***, positive integer ***k***

Question: Does ***P*** halt for some input of length $\leq$ ***k***.

Other Undecidable Problems

Input: a Turing machine P

Question: Is Accept(P) regular?

I.e., is P equivalent to a Finite Automaton?

Input: a CFG

Question: is the language it generates regular?

Input: a CFG

Question: is there any string that it doesn't generate?

(over same alphabet)

Input: two CFGs.

Question: Do they define the same language?

Other Undecidable Problems

Input: a polynomial (in several variables)

Question: Does it have an integer root?

(Y. Matiyasevich, 1970)

Yuri Matiyasevich (b. 1947)

http://www-history.mcs.st-and.ac.uk/Biographies/Matiyasevich.html

Post Correspondence Problem

(a problem about string matching;
see Sipser, Section 5.2)

Emil Post (1897-1954)

http://www-history.mcs.st-andrews.ac.uk/Biographies/Post.html

Revision

- Know and understand the Halting Problem.
- Be able to use mapping reductions.
- Be able to show that some problems are undecidable
- Know examples of undecidable problems.

Reading

- Sipser, pp. 207-209, 216-220, 234-236.