Warm-Up Problem

Please fill out your Teaching Evaluation Survey! Please comment on the warm-up problems if you haven't filled in your survey yet.

Warm up: Given a program P that accepts input, is there an input where the program runs forever?

Theorem: The Exists-Input-Runs-Forever Problem is undecidable.

Decidability
Introduction and Reductions to the Halting Problem

Carmen Bruni

Lecture 23

Based on slides by Jonathan Buss, Lila Kari, Anna Lubiw and Steve Wolfman with thanks to B. Bonakdarpour, A. Gao, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Last Time

- Define a decidable problem.
- Describe the Halting Problem.
- Show that problems are decidable.
- Give reductions to prove undecidability

Learning Goals

- Prove that the Halting Problem is Undecidable
- Prove other problems are undecidable based on reductions to the Halting Problem.

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? That is, why does $H(P, P)$ make sense?
  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 decides 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 decides the Halting problem for every program and every input. Therefore, H does not exist.

□

The Halting Problem is Undecidable.

The algorithm X:

- Takes a program Q as input
- Makes two copies of Q
- Runs $H(Q, Q)$:
  - If $H(Q, Q)$ outputs "yes", then X goes into an infinite loop and runs forever (ie. it doesn't halt)
  - If $H(Q, Q)$ outputs "no", then X halts and returns 0

Now, we ask "What happens if we try $X(X)$?"

Proof that $X(X)$ Does Halt Is Impossible

Suppose that $X(X)$ halted. Then $H(X, X)$ by construction must have outputted "no", that is, X does not halt when given itself as input. This contradicts our assumption that X did halt when given itself as input.

Proof that $X(X)$ Does Not Halt Is Impossible

Suppose that $X(X)$ did not halt. Then $H(X, X)$ by construction must have outputted "yes", that is, X does halt when given itself as input. This contradicts our assumption that X did not halt when given itself as input.

The combination of this slide and the next shows that this machine X cannot exist. However, every part of the machine X does exist except for possibly the machine H. Hence H cannot exist and we have a contradiction.

Diagonalization

The above method is related to Cantor's diagonalization method which will discuss at the end of today's lecture and the start of tomorrow's lecture.

Specific Input Run Forever Problem

Problem: Given a program P, does it run forever on input $n = 0$?

Theorem: The Specific Input Run Forever Problem is undecidable.

Specific Input Run Forever Problem

Problem: Given a program P, does it run forever on input $n = 0$?

Theorem: The Specific Input Run Forever Problem is undecidable.

Proof: Assume towards a contradiction that the specific input problem is decidable. That is, there is an algorithm B such that when I give it a program P', it returns "yes" if and only if it runs forever on the input $n = 0$.
Given a program P and an input I, construct a program Q that:

- Ignores the input and runs P on I.
- If P when run on I halts, then halt and return 0.

Note that this Q halts at 0 [in fact it halts on *all* inputs!] if and only if P halts on I. We now solve the Halting Problem. (Continued on next slide)

Specific Input Problem

Problem: Given a program P, does it run forever on input $n = 0$?

Theorem: The Specific Input Problem is undecidable.

Proof: (Continued...) We now solve the Halting Problem. Consider an algorithm A:

- It consumes P and I
- It creates Q
- It runs algorithm B on Q and then negates the output.

Now, algorithm B on Q returns "yes" if and only if Q runs forever on input $n = 0$. But, this can happen by construction if and only if P does not halt on I. So negating the final answer means that if we get "no" from the above algorithm, then we have that P does not halt on input I, a contradiction. Hence algorithm B cannot exist, that is, the Specific Input Run Forever Problem is undecidable.

Partial Correctness Problem

Problem: Given a Hoare triple $(\!| P_1 |\!)\ C\ (\!| Q_1 |\!)$, does C satisfy the triple under partial correctness?

Theorem: The Partial Correctness Problem is undecidable.

Partial Correctness Problem

Problem: Given a Hoare triple $(\!| P_1 |\!)\ C\ (\!| Q_1 |\!)$, does C satisfy the triple under partial correctness?

Theorem: The Partial Correctness Problem is undecidable.

Proof: Assume towards a contradiction that there is an algorithm B such that when $(\!| P_1 |\!)\ C\ (\!| Q_1 |\!)$, is satisfied under partial correctness it returns "yes". We claim that we can construct an algorithm A to decide the Halting-No-Input Problem. Let P be a program. Algorithm A is given by

- Run algorithm B on program $(\!| \text{true} |\!)\ P\ (\!| \text{false} |\!)$ and return the negated answer.

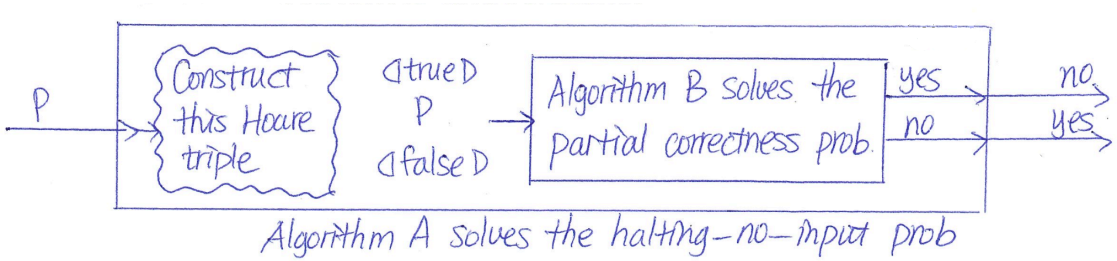
(continued on next slide)

Partial Correctness Problem

Problem: Given a Hoare triple $(\!| P_1 |\!)\ C\ (\!| Q_1 |\!)$, does C satisfy the triple under partial correctness?

Theorem: The Partial Correctness Problem is undecidable.

Proof: (Continued...) Now, by construction, A halts and returns "yes" if and only if program P halts. This is true since the Hoare triple $(\!| \text{true} |\!)\ P\ (\!| \text{false} |\!)$ is never satisfied and so by the definition of partial correctness, this triple satisfies partial correctness if and only if P does not halt. Thus algorithm B will return "no" if and only if program P halts. This is the negation of what we want our program do so flip the answer. Thus if algorithm B exists, then the Halting-No-Input Problem is decidable. This is a contradiction. Thus the Partial Correctness problem is undecidable.



Another Undecidable Problem

Provability of a formula in Predicate Logic:

> Given a formula φ, does φ have a proof?
> (Or, equivalently, is φ valid?)

No algorithm exists to solve provability:

Theorem Provability is undecidable.

Provability In Predicate Logic Is Undecidable

Outline of proof: Suppose that some algorithm Q decides provability.

1. Devise an algorithm to solve the following problem:

   Given a program P and input I, produce a formula $\varphi_{P,I}$ such that $\varphi_{P,I}$ has a proof iff P halts on input I.

2. Combine algorithm Q with the above algorithm to get an algorithm that decides the Halting Problem: On input (P, I), give the formula $\varphi_{P,I}$ as input to Q.

But no algorithm decides the Halting problem.

Thus no algorithm decides provability.

The Technique: "Diagonalization"

The technique used in the proof of the undecidability of the halting problem is called **diagonalization**.

It was originally devised by Georg Cantor (in 1873) for a different purpose.

Cantor was concerned with the problem of measuring the sizes of infinite sets. Are some infinite sets larger than others?

Example. The set of even natural numbers is the same size (!!) as the set of all natural numbers.

Example. The set of all infinite sequences over $\{0, 1\}$ is larger than the set of natural numbers.

(Idea of proof: (A) Assume that each sequence has a number. (B) find a sequence that is different from every numbered sequence.)

Countable sets

A set S is **countable** if there is a one-to-one correspondence between S and the set of natural numbers ($\mathbb{N}$).

How do we prove that a set is **uncountable**?

Cantor's diagonal argument

Consider an infinite sequence $S = (s_1, s_2, ...)$, where each element s_i is an infinite sequence of 1s or 0s (s_i is a binary string of infinite length).

$$
\begin{aligned}
s_1 &= (0, 0, 0, 0, 0, 0, ...) \\
s_2 &= (1, 1, 1, 1, 1, 1, ...) \\
s_3 &= (0, 1, 0, 1, 0, 1, ...) \\
s_4 &= (1, 0, 1, 0, 1, 0, ...) \\
s_5 &= (1, 1, 0, 0, 1, 1, ...) \\
s_6 &= (0, 0, 1, 1, 0, 0, ...) \\
&...
\end{aligned}
$$

Cantor's diagonal argument

There is a sequence $\overline{s}$ such that if $s_{n,n} = 1$, then $\overline{s}_n = 0$, and otherwise $\overline{s}_n = 1$.

$$
\begin{aligned}
s_1 &= (\mathbf{0}, 0, 0, 0, 0, 0, ...) \\
s_2 &= (1, \mathbf{1}, 1, 1, 1, 1, ...) \\
s_3 &= (0, 1, \mathbf{0}, 1, 0, 1, ...) \\
s_4 &= (1, 0, 1, \mathbf{0}, 1, 0, ...) \\
s_5 &= (1, 1, 0, 0, \mathbf{1}, 1, ...) \\
s_6 &= (0, 0, 1, 1, 0, \mathbf{0}, ...) \\
&...
\end{aligned}
$$

Define $\overline{s} = (1, 0, 1, 1, 0, 1, ...)$.

Then $\overline{s}$ is not any of the sequences s_i on the list S.

Cantor's diagonal argument

By construction, $\overline{s}$ is not contained in the countable sequence S.

Let T be a set consisting of all infinite sequences of 0s and 1s. By definition, T must contain S and $\overline{s}$.

Since $\overline{s}$ is not in S, the set T cannot coincide with S.

Therefore, T is uncountable; it cannot be placed in one-to-one correspondence with $\mathbb{N}$.

Cantor's diagonal argument

Therefore, the set of even integers is smaller than the set of all infinite sequences over $\{0, 1\}$.

$\mathbb{R}$ is Uncountable

We will show that a subset of $\mathbb{R}$, per example $(0, 1)$, is uncountable.

Let S be the following countable sequence of elements from $(0, 1)$:

$e_1 = 0.3245890172245\ldots$
$e_2 = 0.1478562003640\ldots$
$e_3 = 0.7129601400235\ldots$
$e_4 = 0.0024533889974\ldots$
$e_5 = 0.5364578912235\ldots$
$e_6 = 0.8581488004222\ldots$
$\ldots$

$\mathbb{R}$ is Uncountable

It is possible to build a new number $\overline{e} = (0.\overline{e}_1\overline{e}_2\overline{e}_3 \ldots)$ in such a way that if $e_{n,n} = k$, where k is one digit natural number, then $\overline{e}_n$ is an one-digit natural number different than k.

$e_1 = 0.\mathbf{3}245890172245 \ldots$
$e_2 = 0.1\mathbf{4}78562003640 \ldots$
$e_3 = 0.71\mathbf{2}9601400235 \ldots$
$e_4 = 0.002\mathbf{4}533889974 \ldots$
$e_5 = 0.5364\mathbf{5}78912235 \ldots$
$e_6 = 0.85814\mathbf{8}8004222 \ldots$
$\ldots$

$\overline{e} = (0.580135 \ldots)$

$\mathbb{R}$ is Uncountable

By the construction, $\overline{e} \in (0,1)$ is not contained in the countable set S.

Therefore the countable set S cannot coincide with $(0,1)$.
We obtain that $(0,1)$ is uncountable.

$\mathbb{R}$ is Uncountable (long proof)

We build an one-to-one correspondence between the set T and a subset of $\mathbb{R}$.

Let function $f(t) = 0.t$, where t is a string in T.
For example, $f(0111\ldots) = 0.0111\ldots$.

Observe that $f(1000\ldots) = 0.1000\ldots = 1/2$, and
$f(0111\ldots) = 0.0111\ldots = 1/4 + 1/8 + 1/16 + \ldots = \sum_{n=1}^{\infty} \frac{1}{n} = 1/2$.

Hence, f is not a bijection.

Cantor's diagonal argument (long proof)

To produce a bijection from T to the interval $(0, 1) \subset \mathbb{R}$:

- From $(0, 1)$ remove the numbers having two binary expansions and form $a = (1/2, 1/4, 3/4, 1/8, 3/8, 5/8, 7/8, ...)$.
- From T, remove the strings appearing after the binary point in the binary expansions of 0, 1, and the numbers in sequence a and form $b = (000\ldots, 111\ldots, 1000\ldots, 0111\ldots, 01000\ldots, 00111\ldots, ...)$.

$g(t)$ from T to $(0, 1)$ is defined by: If t is the nth string in sequence b, let $g(t)$ be the nth number in sequence a; otherwise, let $g(t) = 0.t$.

Cantor's diagonal argument (long proof)

To build a bijection from T to $\mathbb{R}$, we use $tan(x)$, a bijection from $(\pi/2, \pi/2)$ to $\mathbb{R}$.

The linear function $h(x) = \pi.x - \pi/2$ provides a bijection from $(0, 1)$ to $(-\pi/2, \pi/2)$.

The composite function $tan(h(x))$ provides a bijection from $(0, 1)$ to $\mathbb{R}$.

The function $tan(h(g(t)))$ is a bijection from T to $\mathbb{R}$.

Cantor's diagonal argument

Using diagonalization, one can show that (for example):

- $|\mathbb{Q}| = |\mathbb{N}| = |\mathbb{Z}|$
- $|\mathbb{N}| < |2^{\mathbb{N}}|$
- The set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable.