CSCI 341–Fall 2024: Lecture Notes
Set 15: Undecidability & Infinity

Edward Talmage

November 14, 2024

Consider the acceptance problem for Turing Machines: Given a TM M and a string w, does M accept w? We can express this as a language,

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ a string, } w \in L(M)\}$$

and ask whether this language is decidable.

Theorem 1. *A_{TM} is recognizable.*

Proof. We prove this by construction, giving a TM recognizing A_{TM}:

```
On input $\langle M, w\rangle$:
1. Verify that $M$ is a TM, $w$ a string.
2. Let $b = M(w)$ (simulate running $M$ on $w$)
3. Decide $b$
```

□

Exercise: Why is this a recognizer, not a decider?

Theorem 2. *A_{TM} is undecidable.*

That is, given a program and an input, we cannot guarantee that we can, in finite time, tell whether the program will accept the input or not. Practically, this means that we cannot fully automate code tests, or finitely determine what another program will do.

To prove this this theorem, we need some understanding of infinity, so we will put aside all computing machines for a bit and focus on infinity.

1 Infinity

Recall: Given a function $f : D \to R$, f is

- *one-to-one (injective)* if $f(a) = f(b)$ implies that $a = b$
- *onto (surjective)* if for every $b \in R$, there is an $a \in D$ s.t. $f(a) = b$
- *bijective* if it is one-to-one and onto

A bijection uniquely pairs elements between the two sets D and R. This is why bijections are invertible: each element in either set is in exactly one pair, so given an element $y \in R$, we can return the other element in its pair, $x \in D$, giving the unique input in D such that $f(x) = y$. In finite sets, this makes complete sense, and it is straightforward to see that $|S| = |T|$ iff there is a bijection between S and T. To generalize this, $|S|$ denotes the *cardinality* of the set S, which for a finite set is the number of elements in S. If cardinality is not a non-negative integer, then we say that S is infinite. We keep the rule that $|S| = |T|$ iff there is a bijection between them, though.

Definition 1. An infinite set S is *countable* iff there is a bijection $f : \mathbb{N} \to S$, where $\mathbb{N}$ is the set of positive integers (natural numbers).

In general, a bijection from the natural numbers is just a way to count all the elements of the set $1, 2, 3, \ldots$. We must be sure that we do, in fact, get to every element and count it (assign it a number), and do not skip any elements.

1.1 Examples:

Claim 1. *The set of positive even integers is countable.*

Proof. Let $f : \mathbb{N} \to \mathbb{N}_2$, $f(x) = 2x$. This is a an injection as $f(x) \neq f(y)$ when $x \neq y$, and a surjection as every even integer is twice some positive integer, by definition. $\square$

Exercise: Prove that the set of all even integers is countable.

Claim 2. *The set of even integers is countable.*

Proof. Consider the function given by the following table:

n	1	2	3	4	5	6	...
$f(n)$	0	2	-2	4	-4	6	

f is a function, since it defines a value for each natural number n, injective because they are all distinct, and surjective because every even integer will appear. Thus, f is a bijection and the set of even integers is countable. $\square$

Claim 3. *For any finite set Σ, the set Σ^* is countable.*

We can list all strings in Σ^* in shortlex order and count them off.

Claim 4. *The set of positive rational numbers $\mathbb{Q} = \{m/n \mid m, n \in \mathbb{Z}\}$ is countable.*

Exercise: What happens if we count them in this order: $1/1, 2/1, 3/1, 4/1, 5/1, \ldots$. Try to come up with a scheme that counts them all.

Proof. List all rational numbers in a table, with numerator and denominator as coordinates.

$n\backslash m$	1	2	3	4	5	6	...
1	1/1	2/1	3/1	4/1	5/1	6/1	...
2	1/2	2/2	3/2	4/2	5/2	6/2	...
3	1/3	2/3	3/3	4/3	5/3	6/3	...
4	1/4	2/4	3/4	4/4	5/4	6/4	...
5	1/5	2/5	3/5	4/5	5/5	6/5	...
6	1/6	2/6	3/6	4/6	5/6	6/6	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

To count them all, we cannot go by row or column, as we will never finish counting the first row/column and return to count the rest, since each row/column is infinite. Instead, we can count them diagonally, counting all rational numbers m/n with $m + n = 2$ $(1/1)$, then all with $m + n = 3$ $(1/2, 2/1)$, then with $m + n = 4$, and so on. We will count every rational number exactly once, so this is a bijection from $\mathbb{N}$, proving that $\mathbb{Q}$ is countable. $\square$

Exercise: What if we wanted to include negative rational numbers, as well?

Claim 5. *The set of positive real numbers is* uncountable*, which means that it is infinite but not countable.*

Proof. We will simply show that the set of positive real numbers between 0 and 1 is not countable, which implies the desired result. Call the set of real numbers between 0 and 1 $\mathbb{R}^+_{0,1}$. Assume $\mathbb{R}^+_{0,1}$ is countable. Then there is a bijection $f : \mathbb{N} \to \mathbb{R}^+_{0,1}$. List (infinite) decimal representations of all elements x of $\mathbb{R}^+_{0,1}$ in order $f^{-1}(x)$. That is, put them in counting order, with the one we count as number 1 first, then the one counted as number 2, and so on.

We construct a real number which is not in this list. Construct x where the ith digit of x after the decimal point is 1 more than the ith digit of $f(i)$ after the decimal point, $bmod 10$. That is, x is different from the first listed x in the first decimal place, different from the second x in the second, and so on. Thus, x is simultaneously different than every x listed, so our counting does not contain x, and f is not a bijection. This contradiction implies that $\mathbb{R}^+_{0,1}$, and thus $\mathbb{R}^+$, is not countable. $\square$

This proof technique is known as a *Cantor diagonalization*, named after Georg Cantor, a 19th-century mathematician. His general result is the following theorem:

Theorem 3 (Cantor's Theorem)**.** *For any set S, $|\mathcal{P}(S)| \neq |S|$.*

Proof. If S is finite, $|\mathcal{P}(S)| = 2^{|S|} \neq |S|$.

If S is infinite, we must show that there is no bijection from S to $\mathcal{P}(S)$. We will show that there can be no surjection from S to $\mathcal{P}(S)$.

Assume in contradiction that f is a surjection $f : S \to \mathcal{P}(S)$ for some arbitrary infinite set S. Consider the set $T = \{s \mid s \in S, s \notin f(s)\}$, the set of elements which are not in their own image under f. T is an element of $\mathcal{P}(S)$, since it is a set of elements from S. Thus, since f is surjective, there is some $x \in S$ s.t. $f(x) = T$. But, by construction, $x \in T \Leftrightarrow x \notin f(x) = T$, which is a contradiction, so f cannot exist. Since there is no surjection, there can be no bijection, and we have the claim. □

2 Undecidability

What were we doing? We want to show that A_{TM} is undecidable. But first, it behooves us to show that there are, in fact, undecidable languages.

Theorem 4. *Not all languages are decidable.*

Proof. Every decidable language has a TM which decides it. There are countable many TMs, since each can be expressed as a string using tuple notation. The set of all languages is the power set of Σ^*. Σ^* is countably infinite, as we showed above, so Cantor's theorem tells us its power set is uncountable. Thus, there is not a bijection from the set of TMs to the set of languages (or we could compose that with the counting of TMs to count languages), so some language is undecidable. □

This is a non-constructive proof: We now know that there are undecidable languages (technically just that there is at least one undecidable language), but we do not know what any of them are. We will next show that a specific, useful language is undecidable.

Theorem 5. $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM, } w \in L(M)\}$ *is undecidable.*

Proof. Assume in contradiction that A_{TM} is decidable and *Acc* is a TM which decides it. That is,

$$\mathit{Acc}(\langle M, w\rangle) = \begin{cases} \text{Accept if } M \text{ accepts } w \\ \text{Reject if } M \text{ does not accept } w \text{ (rejects or loops)} \end{cases}$$

Since *Acc* is a TM, we can use it to build other TMs. Consider, for example, the TM *Opp*:

On input $\langle M\rangle$, where M is a TM:

1. Run $\mathit{Acc}(\langle M, \langle M\rangle\rangle)$
2. Accept if *Acc* rejects and reject if *Acc* accepts.

Consider the behavior of *Opp*:

$$\mathit{Opp}(\langle M\rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M\rangle \\ \text{Reject if } M \text{ does accept } \langle M\rangle \end{cases}$$

Opp may seem silly, but if *Acc* exists, then *Opp* must also exist. Now, just for kicks, run $\mathit{Opp}(\langle \mathit{Opp}\rangle)$.

$$\mathit{Opp}(\langle \mathit{Opp}\rangle) = \begin{cases} \text{Accept if } \mathit{Opp} \text{ does not accept } \langle \mathit{Opp}\rangle \\ \text{Reject if } \mathit{Opp} \text{ accepts } \langle \mathit{Opp}\rangle \end{cases}$$

In English, *Opp* accepts its own encoding if and only if it does not accept its own encoding. This is a contradiction, so *Opp* cannot exist, which means *Acc* cannot exist. Therefore, A_{TM} is undecidable. □

Compare this to Cantor's diagonalization: Build a table where each row corresponds to a Turing Machine and each column is a string. We only really care about strings which encode Turing Machines. Set table cell $(M_i, \langle M_i\rangle)$ to $\mathit{Acc}(\langle M_i, \langle M_i\rangle\rangle)$.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
M_1	$\underline{\text{A}}$	R	A	A	...
M_2	R	$\underline{\text{R}}$	A	A	...
M_3	R	R	$\underline{\text{A}}$	R	...
M_4	R	R	R	$\underline{\text{R}}$	...
$\vdots$					

On input $\langle M_i, \langle M_i \rangle \rangle$, *Opp* does the opposite of *Acc*, but then *Opp* is different than any of the M_i's in this table, so our list of every possible Turing Machine does not contain a solution to every problem!

Theorem 6. *L and $\overline{L}$ are recognizable iff L is decidable.*

Proof. First, we will prove the reverse implication: If L is decidable, then L and $\overline{L}$ are recognizable. For L, we can simply run the decider for L, because a decider is also a recognizer. For $\overline{L}$, we run the decider for L and decide the opposite. This is a decider for $\overline{L}$, and thus also a recognizer.

Next, consider the reverse implication. Assume that L and $\overline{L}$ are both recognizable, by TMs M_1 and M_2, respectively. We build a decider M for L:

On input W:

1. Alternate between simulating M_1 and M_2, one step of each at a time. For example, we can have two tapes and update the first as M_1 would act and the second as M_2 would act.

2. If M_1 accepts, accept. If M_2 accepts, reject.

Since either M_1 or M_2 will accept in finite time, M will reach that accept state in at most twice as many steps as the accepting machine, which is also finite, so M is a decider. ☐

Corollary 1. *$\overline{A_{TM}}$ is unrecognizable.*

Exercise: Prove this claim.

Proof. A_{TM} is recognizable, since given $\langle M, w \rangle$ we can run $M(w)$ and return the result. A_{TM} is undecidable, as we just proved, so $\overline{A_{TM}}$ cannot be recognizable, or A_{TM} and its complement both being recognizable would imply that A_{TM} is decidable. ☐

This means that, in general, given an piece of code and an input, we cannot in a finite time determine whether the code will fail or run forever. There can be no general-purpose infinite loop detectors!