UNDECIDABLE PROBLEMS

Mihai-Valentin DUMITRU
mihai.dumitru2201@upb.ro

October 2022

We have seen many examples of what Turing Machines can do and strongly suggested that there are things that they *cannot* do. A consequence of the Church-Turing thesis is that the set R represents exactly those decision problems that can be solved by an algorithm; so any decision problem outside of R would not be algorithmically solvable.

Indeed, we claimed that the problem *"Pairs"* introduced at the very first lecture was an example of such a problem, not belonging to R; but until now we have not rigorously proven this claim, nor have we proven that there are any decision problems that are undecidable.

In this lecture, we will formally show that there are *infinitely many* undecidable problems and identify a particular one, *the Halting Problem*.

In order to achieve the first result, we will take a short detour through set theory to introduce (recap?) the notion of *cardinality*, which is one way of extending the concepts of "smaller" or "larger" when comparing *infinite* sets.

1 Cardinality

We say that two sets A and B "have the same cardinality" if there exists some bijection between them. We denote this by the notation $|A| = |B|$.

For *finite sets*, this conforms very well with our intuition of "size of set" which associates a natural number to each set. For example, the two sets: $A = \{"alpha", "beta", "gamma"\}, B = \{2, 4, 9\}$ have the same cardinality as can be seen from the following bijection[†]:

$f_1 : A \to B$

$f_1("alpha") = 1$

$f_1("beta") = 2$

$f_1("gamma") = 3$

But if we were to compare A with some other set like $C = \{a, b, c, d\}$, we can find no bijection between them, so they have *different cardinalities*; $|A| \neq |C|$.

Because there exists *an injection* from set A to set C, we can say that set A is "smaller" than C; $|A| \leq |C|$. One possible injection is:

$f_2 : A \to B$

$f_2("alpha") = a$

$f_2("beta") = b$

$f_2("gamma") = c$

If $|A| \leq |C|$ and $|A| \neq |C|$, we say $|A| < |C|$.

When we move on into the realm of *infinite sets*, cardinality becomes less intuitive (but, nevertheless, very useful). For example, we can show that, cardinality-wise, the set of all integers $\mathbb{Z}$ has the same cardinality as the set of all natural numbers $\mathbb{N}$. Consider the bijection:[‡]

$f_3 : \mathbb{Z} \to \mathbb{N}$

$f_3(0) = 0$

[†]There are other bijections as well, but we only care that there's at least one.

[‡]This can also be expressed succinctly as $f_3(x) = 2 * |x| - sign(x)$, where $|x|$ is the absolute value of the integer and $sign(x)$ is 0 for positive integers and 1 for negative integers.

$f_3(-1) = 1$

$f_3(1) = 2$

$f_3(-2) = 3$

$f_3(2) = 4$

$f_3(-3) = 5$

$f_3(3) = 6$

...

This will probably seem weird, as the natural numbers are a proper subset of the integers ($\mathbb{N} \subsetneq \mathbb{Z}$); but they both have the same cardinality.

What's more, we can show that the *rational numbers* $\mathbb{Q}$ also have the same cardinality as the natural numbers $\mathbb{N}$, but this is outside the scope of this lecture.

An interesting result for us is that the naturals have the same cardinality as the set of all strings over some alphabet $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$.

$nts : \mathbb{N} \to \Sigma^*$

$nts(0) = \epsilon$

$nts(1) = \sigma_1$

$nts(2) = \sigma_2$

...

$nts(n) = \sigma_n$

$nts(n+1) = \sigma_1\sigma_1$

$nts(n+2) = \sigma_1\sigma_2$

...

Coming back to familiar sets of numbers, there exists no bijection from the naturals $\mathbb{N}$ to the reals $\mathbb{R}$. The cardinality of $\mathbb{R}$ is larger than that of $\mathbb{N}$.

In the next section, we shall show that the cardinality of the real interval $[0, 1)$ is larger than that of $\mathbb{N}$; that the same holds for $\mathbb{N}$ and $\mathbb{R}$ should then be obvious.

1.1 Diagonalization

Recall the bijection we used to show that the naturals and the integers have the same cardinality. Basically, in order to show that some set S has the same cardinality as the natural numbers, it is sufficient to show that we can *systematically place all elements of that set in a list* – because the items in a list can all be denoted as: 0, 1, 2, ... By the term "systematically", we mean in such a way as to be clear what element of S would be at any position n.

We shall prove, by *reductio ad absurdum*, that we cannot define such a list of all the real numbers in the interval $[0, 1)$.

Theorem 1.1. $|\mathbb{N}| < |[0, 1)|$

Proof. Our proof will consist of two parts; first we will show that there can be no bijection between these two sets, so their cardinalities differ.

Assume that we can make a list of the numbers in $[0, 1)$:

0 $0.d_{00}d_{01}d_{02}d_{03}d_{04}...$

1 $0.d_{10}d_{11}d_{12}d_{13}d_{14}...$

2 $0.d_{20}d_{21}d_{22}d_{23}d_{24}...$

3 $0.d_{30}d_{31}d_{32}d_{33}d_{34}...$

...

Where each d_{ij} is a single digit: the j^{th} decimal figure of the i^{th} number in the list. We consider each number to have infinite decimal digits; for those with a finite representation, such as 0.253, simply consider a neverending trail of 0s, i.e. 0.2530000...

Now consider the following real number:

$X = 0.x_0x_1x_2x_3x_4...$

Where x_i is a single digit, as follows:

- if d_{ii} is 0, then x_i is 1
- if d_{ii} is not 0, then x_i is 0

This number is not in our initial list!

- x_0 is different than d_{00}, so X differs at least in the first decimal place from the first number in the list.
- x_1 is different than d_{11}, so X differs at least in the second decimal place from the second number in the list.
- etc.

Generally, x_i is different than d_{ii} so X differs at least in the i^{th} decimal from the i^{th} number in the list.

Thus X differs from any number in the list, so it's not on the list.

But our initial assumption was that our list contains all numbers in $[0, 1)$ and we forced no constraints on it. So our list cannot exist – there is no bijection between $\mathbb{N}$ and $[0, 1)$; $|\mathbb{N}| \neq |[0, 1)|$.

Consider the following function:

$i : \mathbb{N} \to [0, 1)$

$i(n) = 0.digits(n)$

Where $digits(n)$ represents the sequence of base10 digits used to represent n. For example, $i(12) = 0.12$, $i(734837) = 0.734837$. As i is an injection, $|\mathbb{N}| \leq |[0, 1)|$.

□

2 There exist undecidable problems

Now that we are somewhat familiar with the concept of cardinality, we will apply it in our context of problems and machines, to show that *there are functions for which no Turing Machine that decides them exists.*

2.1 Cardinality of Turing Machines

Let $\mathbb{M}$ be the set of all Turing Machines.

Theorem 2.1. $|\mathbb{M}| \leq |\mathbb{N}|$

(A stronger statement is true: $|\mathbb{M}| = |\mathbb{N}|$, but we have no need for it, so we won't prove it.)

Proof. In the previous lecture we showed how we can encode each Turing Machine as a string over the binary alphabet $\{0, 1\}$, i.e. we provided an injection from $\mathbb{M}$ to $\{0, 1\}^*$; $|\mathbb{M}| \leq |\{0, 1\}^*|$.

In the previous section, we showed how the set of all binary strings has the same cardinality as the set of natural numbers; $|\{0, 1\}^*| = |\mathbb{N}|$.

□

2.2 Cardinality of decision problems

Let $\mathbb{D}$ be the set of all decision problems.

Theorem 2.2. $|\mathbb{N}| < |\mathbb{D}|$

Proof. To represent a decision problem f, we will use an infinite sequence of 0s and 1s; the i^{th} element of the sequence is 1 if $f(nts(i)) = TRUE$, or 0 if $f(nts(i)) = FALSE$, where nts is our bijection from strings to naturals, presented in section 1.

Assume we can make a list of all decision problems.

0 $a_{00}a_{01}a_{02}a_{03}a_{04}...$

1 $a_{10}a_{11}a_{12}a_{13}a_{14}...$

2 $a_{20}a_{21}a_{22}a_{23}a_{24}...$

3 $a_{30}a_{31}a_{32}a_{33}a_{34}...$

...

For example, decision problem f_0, on input $nts(2)$ has an answer corresponding to a_{02}.

Consider the following decision problem $f_d : \Sigma^* \to \{FALSE, TRUE\}$:

$$f_d(w) = \begin{cases} TRUE & f_{nts^{-1}(w)}(w) = FALSE \\ FALSE & f_{nts^{-1}(w)}(w) = TRUE \end{cases}$$

For $w = nts(0)$, the value of $f_d(w)$ is $TRUE$ if $f_0(w) = FALSE$ and vice-versa; so f_d is not f_0, because they differ at least for $w = nts(0)$.

For $w = nts(1)$, the value of $f_d(w)$ is $TRUE$ if $f_1(w) = FALSE$ and vice-versa; so f_d is not f_1, because they differ at least for $w = nts(1)$.

In general, for any $i \in \mathbb{N}$, $w = nts(i)$, the value of $f_d(w)$ is $TRUE$ if $f_i(w) = FALSE$ and vice-versa; so f_i is not f_i, because they differ at least for $w = nts(1)$.

f_d does not appear in the list, so our assumption that a list that can contain all decision problems exists leads to a contradiction; such list can't exist. $|\mathbb{N}| \neq |\mathbb{D}|$.

We will now show an injection from $\mathbb{N}$ to $\mathbb{D}$. For each natural number n, we can obtain a unique binary encoding consisting of the symbols 0 and 1, preceded by an infinite string of leading 0s. Number the last digits "0, the second to last "1", the third to last "2" and so on. Let $f_n(nts(i)) = TRUE$ if the i^{th} digit of n is 1, $FALSE$ otherwise; $\forall i \in \mathbb{N}$.

For example, given the number 6, its binary encoding will be: "...0000110". $f_6(nts(0)) = FALSE$, $f_6(nts(1)) = TRUE$, $f_6(nts(2)) = TRUE$ and, for all other values f_6 is $FALSE$.

The function g that takes a number n and maps it to f_n is an injection from $\mathbb{N}$ to $\mathbb{D}$; $|\mathbb{N}| \leq |\mathbb{D}|$.

□

2.3 There are "fewer" machines than problems

We showed that $|\mathbb{M}| \leq |\mathbb{N}|$, then $|\mathbb{N}| < |\mathbb{D}|$; this gives us$|\mathbb{M}| < |\mathbb{D}|$.

Thus, there is no injection from $\mathbb{D}$ to $\mathbb{M}$ – there are some decision problems to which we cannot associate a Turing Machine. So there are infinitely many decision problems, for which there is no Turing Machine that can decide them.

In fact, we have proven a stronger statement. Remember the weaker concept of "solving a decision problem": *acceptance*. This was also defined in terms of existence of some Turing Machines. The fact that the cardinality of decision problems is strictly greater than that of all Turing Machines means that there are infinitely many decision problems for which there is no Turing Machine that can accept them.

Given an undecidable problem f, we have no hope of devising an algorithm that can solve it; i.e. always produce the correct answer in a finite amount of steps.

But if we know that f is undecidable, but *acceptable,* then, by definition, there exists a Turing Machine M, such that, for all words w for which $f(w) = TRUE$, $M[w] \to TRUE$. When $f(w) = FALSE$, the machine will either provide the correct answer or loop forever. So a clever idea is to check if the machine loops forever; if yes, then we can conclude that the answer to $f(w)$ is $FALSE$.

Unfortunately, such a check cannot itself be performed algorithmically, as we shall show in the next section.

3 The Halting problem

Definition 3.1. The *"halting problem"* is the decision problem:

$$\texttt{HALT}(enc((M,w))) = \begin{cases} TRUE & M[w] \text{ halts} \\ FALSE & otherwise \end{cases}$$

Here is part of the reason why this problem is important to us:

If $HALT$ were decidable, we could build a machine M_{HALT} that decides it. We could then take any machine M and input w, serve them as input to M_{HALT} and find out if $M[w]$ halts. Only if we know it halts, we run $M[w]$ and get the result.

If M is a machine that accepts a problem f, then the fact that $M[w]$ doesn't halt assures us that $f(w) = FALSE$; in effect, we can now decide f, i.e. always obtain the correct answer in a finite amount of time.

3.1 HALT is undecidable

Theorem 3.1. $\texttt{HALT} \notin R$

Proof. Assume that there exists a machine M_H, s.t. M_H decides HALT.

We construct a machine D, which takes as input the encoding of some Turing Machine M; the behavior of D is as follows:

1. on input $enc(M)$, simulate $M_H[enc((M, enc(M))]$
2. if $M_H[enc((M, enc(M)))] \to FALSE$, then transition to state Y
3. else (if $M_H[enc((M, enc(M)))] \to TRUE$) transition to a new state *loop*, s.t. $\forall g \in \Gamma_{M_H}, \delta_{M_H}(loop, g) = (loop, g, \to)$

Thus, for any machine M, $D[enc(M)]$ either halts in state Y or it doesn't ever halt. What about $D[enc(D)]$?

Assume that it halts; then $M_H[enc((D, enc(D)))] \to TRUE$, so (see the 3rd step), D will transition to state *loop* and never halt. This is a contradiction!

Assume $D[enc(D)]$ doesn't halt; then $M_H[enc((D, enc(D)))] \to FALSE$, so (see the 2nd step), D will transition to state Y, halting. This is a contradiction!

There is no consistent behavior for $D[enc(D)]$. Let's look at the "components" of D and check that it was properly constructed:

- it simulates the operation of another machine on some input
- it branches based on the output of the simulation
- it can transition to final state Y
- it can transition to a state that goes to the right forever, leaving the tape unchanged, never halting

From previous lectures and labs, it should be evident that a machine can be designed to perform all these four actions, so D is properly defined.

The first bullet is the most complex operation; but recall the construction from the lab of a machine $M_{f \circ g}$ that computes the functional composition of two computable functions f and g, based on the machines that compute them: M_f and M_g. It's possible to have a machine contain the behavior of another machine *"baked into it"*.

Then the contradiction must stem from the initial assumption about the existence of M_H, the machine that decides HALT. So M_H cannot exist; $\texttt{HALT} \notin R$.

□

3.2 HALT is acceptable

Theorem 3.2. $\texttt{HALT} \in RE$

This should be an intuitive result. Remember what it means for a problem f to be *acceptable*: there has to exist a machine M_f that halts in state Y for any input for which $f(w) = TRUE$; for others it can either halt in state N or loop forever.

We will prove this theorem by constructing a machine that accepts HALT.

Let M_H be a Turing Machine with the following behavior:

- on input v, check if v is a valid encoding of a tuple (M, w) where M is a Turing Machine and w is a word. If it isn't, transition to N.
- simulate $M[w]$; transition to Y.

Theorem 3.3. M_H accepts HALT.

Proof. Remember, from the "Computing" lecture, what it means for a machine to accept a function. For all tuples (M, w) consisting of a machine and a word:

$\texttt{HALT}(enc((M, w)) = TRUE \Leftrightarrow M_H[enc((M, w))] \rightarrow TRUE$

This statement is made of two implications; we will have to prove both separately.

"$\Rightarrow$":

$\texttt{HALT}((enc((M, w))) = TRUE \Rightarrow M[w]$ halts $\Rightarrow$ in the second step, M_H transitions to $Y \Rightarrow M_H[enc((M, w))] \rightarrow TRUE$

"$\Leftarrow$":

$M_H[enc((M, w))] \rightarrow TRUE \Rightarrow M[w]$ halts $\Rightarrow \texttt{HALT}(enc((M, w))) = TRUE$

□

4 References and further reading

The concept of cardinality and the diagonal argument used in 1.1 were first presented by Georg Cantor (1845-1918) in his 1891 article: "Uber eine elementare Frage der Mannigfaltigkeitslehre." [1].

The idea that there are functions for which no machine exists to compute them is central to Turing's 1936 paper [2]. However, Turing's formalism and conventions were very different than in this lecture.

The modern formulation of "the halting problem", originates from Martin Davis's textbook "Computability and unsolvability" [3].

The proof of the undecidability of HALT presented in this lecture is adapted from section "An undecidable language", in chapter 4.2 of Michael Sipser's "Introduction to the Theory of Computation" [4].

Bibliography

[1] Georg Cantor. "Uber eine elementare Frage der Mannigfaltigkeitslehre". In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 1 (1891), pp. 75–78.

[2] Alan Mathison Turing et al. "On computable numbers, with an application to the Entscheidungsproblem". In: *J. of Math* 58.345-363 (1936), p. 5.

[3] Martin Davis. *Computability and unsolvability*. McGraw-Hill Book Company, 1958, p. 70.

[4] Michael Sipser. *Introduction to the Theory of Computation, Third Edition*. CENGAGE Learning, 2012. Chap. 4 Undecidability, pp. 207–209.