# CSE2315 Slides week 6

Matthijs Spaan Stefan Hugtenburg

Algorithmics group

Delft University of Technology

25 March 2020

# This lecture

- Infinity
- Countable vs. uncountable
- Hotel Hilbert
- Countability of $\mathbb{Q}$
- Uncountability of $\mathbb{R}$
- Intuition undecidability of the “Halting problem”
- Undecidability acceptance problem
- Universal Turing machine
- Undecidability of $A_{\mathsf{TM}}$ and diagonalization method
- Co-Turing-recognizability
- A non-Turing-recognizable problem
- What is reduction?
- Direct reduction

# What is infinity?

(See also video Infinity: Countable and Uncountable )

- Natural numbers $\mathbb{N} = \{0, 1, 2, \ldots, n, \ldots\}$
- Hotel Hilbert
- Countability
- Countably infinite

(Sipser p. 202)

# Functions (Def. 4.12)

A function $f : A \to B$ is:

- surjective (onto): for each $b \in B$ there exists an $a \in A$ such that $f(a) = b$.
- injective (one-to-one): if $a \neq b$, then $f(a) \neq f(b)$.
- bijective (correspondence): $f$ is both surjective and injective.

# Equinumerosity (finite case)

Two finite sets $A$ and $B$ are equinumerous (the same size) if they contain equally many elements, so if there exists a bijection $f : A \to B$.

(Sipser, Def. 4.12)

# Equinumerosity (infinite case)

For infinite sets, this can be generalized as follows:

Two infinite sets $A$ and $B$ are equinumerous (the same size) if there exists a bijection $f : A \to B$.

(Sipser, Def. 4.12)

# Countability (Def. 4.14)

- A set $A$ is *countably infinite* (a.k.a. at most countable) if there exists a bijection between $A$ and $\mathbb{N}$.
- A set $A$ is *countable* if $A$ is finite, or if $A$ is countably infinite.
- Alternative definition: A set $A$ is *countable* if $A = \emptyset$ or if there exists a surjection from $\mathbb{N}$ to $A$.
- A set is *uncountable* if it is not countable.

# Countable sets

- $\mathbb{Z}$
- $\mathbb{N} \times \mathbb{N}$
- $\mathbb{Q}$
- The set of programs in language $X^{++}$

# Example $\mathbb{N} \times \mathbb{N}$

Enumeration:

$$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),(3,0),(2,1),(1,2),(0,3),\ldots$$

Define $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$:

$$f(m,n) = \frac{(m+n)(m+n+1)}{2} + n.$$

For instance: $(2,3) \mapsto 18$.

$f$ is an injection.

# Exercise

Think of an enumeration for all rational numbers greater than or equal to zero; in other words, think of an enumeration for the set

$$\{q \in \mathbb{Q} \mid q \geq 0\}.$$

# Alephs

$\aleph$ is the first letter of the Hebrew alphabet. $\aleph_0$ denotes the first degree of infinity, the amount of natural numbers. We can compute using $\aleph$s (remember Hotel Hilbert):

- $\aleph_0 + 1 = \aleph_0$
- $\aleph_0 + \aleph_0 = \aleph_0$
- $\aleph_0 \cdot \aleph_0 = \aleph_0$

# Clever construction needed

Given a table with rows of numbers:

$$\begin{array}{ccccc}
8 & 13 & 99 & 71 & \dots \\
101 & 94 & 99 & 34 & \dots \\
37 & 94 & 49 & 85 & \dots \\
20 & 54 & 67 & 12 & \dots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

How can one construct a row that does not occur in the table in a systematic way?

# Diagonal

Do 'something' to the diagonal, for instance:

$$\begin{array}{ccccc} 8+1 & 13 & 99 & 71 & \dots \\ 101 & 94+1 & 99 & 34 & \dots \\ 37 & 94 & 49+1 & 85 & \dots \\ 20 & 54 & 67 & 12+1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

This yields the row:
9, 95, 50, 13, ...
This row cannot occur in the table!

# $\mathbb{R}$ is uncountable (Th. 4.17)

Proof:
The default proof shows that $(0, 1)$ is uncountable. If $(0, 1)$ is not countable, then $\mathbb{R}$ is not either.

Suppose $(0, 1)$ is countable, derive a contradiction from this using diagonalization.

# $\mathbb{R}$ is uncountable (2)

Suppose $r_0, r_1, r_2, \ldots, r_m, \ldots$ is an enumeration of $(0,1)$.
Every element $r_m \in (0,1)$ can be written as:

$$r_m = 0.r_{m_0} r_{m_1} r_{m_2} \ldots r_{m_n} \ldots$$

Define:

$$d = 0.d_0 d_1 d_2 \ldots d_n \ldots$$

with

$$d_n = \begin{cases} 8 & \text{if } r_{nn} = 7, \\ 7 & \text{otherwise.} \end{cases}$$

Then we get a contradiction, since

1. $d \in (0,1)$, but
2. due to the construction, $d \neq r_m$ for all $m = 0, 1, 2, \ldots$, so $d$ does not occur in the enumeration.

# Exercise

Why can we not show that $\mathbb{Q}$ is uncountable in the same way we did with $\mathbb{R}$? After all, elements from $\mathbb{Q}$ also admit a decimal representation. In other words, what would go wrong in that proof?

# Characteristic function

Suppose we have a subset $A \subseteq \mathbb{N}$. $A$ can be represented in at least two ways:

1. as a set, for instance $A = \{0, 1, 4, 9, 16, 25, \ldots\} = \{n^2 \mid n \in \mathbb{N}\}$.
2. as a characteristic function $\chi_A : \mathbb{N} \to \{0, 1\}$:

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

For instance, $\chi_A(0) = 1, \chi_A(1) = 1, \chi_A(2) = 0, \ldots$.

# Characteristic function of a language

For a language $L \subseteq \Sigma^*$ we can also define a characteristic function $\chi_L : \Sigma^* \to \{0, 1\}$:

$$\chi_L(x) = \begin{cases} 1 & \text{if } x \in L, \\ 0 & \text{if } x \notin L. \end{cases}$$

# Characteristic sequence of a language

Instead of the characteristic function, Sipser uses the concept of characteristic sequence. This assumes a standard enumeration of $\Sigma^* = \{s_0, s_1, s_2, \ldots, s_i, \ldots\}$. Now $\chi_L$ is an infinite sequence of zeroes and ones: $\chi_L = b_0 b_1 b_2 \ldots b_i \ldots$ where:

$$b_i = \begin{cases} 1 & \text{if } s_i \in L, \\ 0 & \text{if } s_i \notin L. \end{cases}$$

(Sipser, p. 206)

# Other uncountable sets

- The set $\mathscr{P}(\mathbb{N}) = \{V \mid V \subseteq \mathbb{N}\}$ of all subsets of $\mathbb{N}$ is uncountable.
- The set of all functions $f : \mathbb{N} \to \mathbb{N}$ is uncountable.

# Non-recognizable languages (Cor. 4.18)

There exist languages that are not Turing-recognizable.

(See also video Languages That Are Not Turing Recognizable )

# Why?

1. $\Sigma^*$ is countably infinite.
2. For a language $L$ over $\Sigma$, we have: $L \subseteq \Sigma^*$.
3. The set of all languages over $\Sigma$ equals $\mathscr{P}(\Sigma^*)$.
4. The set $\mathscr{P}(\Sigma^*)$ is uncountable (by Cantor's theorem).
5. There exist countably many Turing machines with input alphabet $\Sigma$ since they can be encoded as a string over some alphabet.
6. Ergo!

# The Halting problem

Does a program $P(x,y)$ exist that decides for every input consisting of a program $x$ and input $y$, whether $x$ halts on input $y$ or not?

# Answer

NO

# Informal proof

Suppose $P(x, y)$ exists. Suppose now we have the following code:

```
procedure Q (x: string);
  function P (x, y: string): boolean;
    begin
      ...
    end;
  begin
    if not P(x, x)
      then return
      else loop
  end;
```

Does `Q(Q)` halt?

# The Acceptance problem, formally

(See also video The Universal Turing Machine )

Before reviewing the Halting problem, we treat the Acceptance problem, which is a variant of it:

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}.$$

$A_{\mathsf{TM}}$ is not decidable (Th. 4.11), but is Turing-recognizable.

# $A_{\mathsf{TM}}$ is Turing-recognizable

The following TM $U$ recognizes the language $A_{\mathsf{TM}}$:

$U$ = "On input $\langle M, w \rangle$, with $M$ a TM and $w$ its input:

1. Simulate $M$ on input $w$.
2. If $M$ reaches an accepting state, accept;
   if $M$ reaches a rejecting state, reject."

Note: $U$ does not necessarily terminate on every input $\langle M, w \rangle$!

# Universal Turing machine (UTM)

The TM $U$ from the previous slide is a Universal Turing machine (UTM). A UTM can run arbitrary TMs on arbitrary input. In fact, every modern computer is a UTM: we can use it to run arbitrary programs.

A UTM embodies the concept of stored program: the TM (the program) and its input (the data) are both stored on the tape of the UTM. The TM and its input together form the input of the UTM.

# $A_{\mathsf{TM}}$ is not decidable

(See also video The Undecidability of the [Acceptance] Problem ; the Acceptance problem is erroneously called the Halting problem there, but the proof is correct.)

Proof by contradiction: Suppose $A_{\mathsf{TM}}$ is decidable (the supposition).

We will show that we can derive a contradiction from this supposition.

# Result of the supposition

Then there exists a TM $H$ that decides $A_{\mathsf{TM}}$:

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts input } w, \\ \text{reject} & \text{if } M \text{ does not accept input } w. \end{cases}$$

# Construction based on $H$

Now construct the following TM $D$ that uses $H$:

$D$ = "On input $\langle M \rangle$, with $M$ a TM:

1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what $H$ outputs
   that is, if $H$ accepts, reject; if $H$ rejects, accept."

# Contradiction

We arrive at a contradiction if we ask ourselves:

What happens with $D(\langle D \rangle)$ ???

$D$ accepts $\langle D \rangle$ iff $D$ rejects $\langle D \rangle$

Because of this contradiction, we conclude that a decider $H$ cannot exist. Therefore, $A_{\mathsf{TM}}$ is not decidable.

In fact, this proof contains the diagonalization method in a hidden form. . .

# A non-Turing-recognizable language

(See also video A Language That Is Not Turing Recognizable )

A language $L$ is co-Turing-recognizable if $\overline{L}$ is Turing-recognizable.

A language is decidable iff it is both Turing-recognizable and co-Turing-recognizable (Th. 4.22).

The language $\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable (Cor. 4.23).

# Exercise

We say $C$ separates languages $A$ and $B$ iff $A \subseteq C$ and $B \subseteq \overline{C}$.

Prove that for every pair of disjoint co-Turing-recognizable languages, a decidable language exists that separates the two of them.

Hint: remember the proof of Th. 4.22.

# What is reduction?

(See also video Reducibility: A Technique for Proving Undecidability )

Suppose we have two problems $A$ and $B$, and there is a method to solve $B$.

Suppose we also have a method to reduce $A$ to problem $B$.

Conclusion: now we can also solve problem $A$.

# Example reduction

We know how to add natural numbers (primary school). Using this, we can also multiply, since:

$$\begin{cases} 0 \cdot m & = & 0 \\ (n+1) \cdot m & = & n \cdot m + m. \end{cases}$$

Using this recursive definition, multiplication has been reduced to repeated addition.

# Reduction

Let $A$ and $B$ be problems (languages). If $A$ can be reduced to $B$, this means:

1. a way to solve $B$ yields a way to solve $A$;
2. if there is no way to solve $A$, there cannot be a way to solve $B$.

Notation for reduction: $A \leq B$.

This can also be read as:

$A$ easier than or as difficult as $B$.

# Forms of reduction

Sipser gives three forms of reduction:

1. direct reduction (own terminology),
2. reduction via computation histories, and
3. reduction via mappings (mapping reducibility).

# The Halting problem (direct reduction)

The Halting problem

$$HALT_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}.$$

is not decidable (Th. 5.1).

Proof: Reduction from $A_{\mathsf{TM}}$ to $HALT_{\mathsf{TM}}$. We must show that $A_{\mathsf{TM}} \leq HALT_{\mathsf{TM}}$.
We suppose there is a decision procedure for $HALT_{\mathsf{TM}}$ and show how this would yield a decision procedure for $A_{\mathsf{TM}}$.

(See also video Halting Problem: A Proof by Reduction )

# Direct reduction 2

(See also video Does a TM Accept Any String? )

The language $E_{\mathsf{TM}}$ defined as

$$E_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

is not decidable (Th. 5.2).

Proof: via direct reduction:

$$A_{\mathsf{TM}} \leq E_{\mathsf{TM}}.$$

Suppose $R$ is a TM that decides $E_{\mathsf{TM}}$.

# Auxiliary construction

Suppose we have $\langle M, w\rangle$. To be able to use $R$, we modify $M$ into $M_1$:

$M_1$ = "On input $x$:

1. If $x \neq w$, reject.
2. If $x = w$, run $M$ on input $w$ and accept when $M$ accepts input $w$."

Now:

$$L(M_1) = \{w\} \neq \emptyset \quad \Leftrightarrow \quad M \text{ accepts } w.$$

This $M_1$ can be "fed" to the decider $R$.

# Nonregular languages

There exist nonregular languages:

Let $\Sigma = \{0, 1\}$. The language $L \subseteq \Sigma^*$ defined as

$$L = \{0^n 1^n \mid n \in \mathbb{N}\}$$

is not regular (Sipser, Ex. 1.73).

"A DFA cannot remember what it has read."

# Direct reduction 3

The language $REGULAR_{\mathsf{TM}}$ defined as

$$REGULAR_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$$

is not decidable (Th. 5.3).

Proof: via direct reduction:

$$A_{\mathsf{TM}} \leq REGULAR_{\mathsf{TM}}.$$

Suppose $R$ is a TM that decides $REGULAR_{\mathsf{TM}}$. Now use the fact that there exist nonregular languages.

# Construction

Based on $\langle M, w\rangle$, define the following $M_2$:

$M_2$ = "On input $x$:

1. If $x$ is of the form $0^n 1^n$, accept.
2. If $x$ is not of this form, run $M$ on $w$ and accept if $M$ accepts."

We now have:

$$L(M_2) = \Sigma^* \quad \Leftrightarrow \quad M \text{ accepts } w$$
$$L(M_2) = \{0^n 1^n \mid n \in \mathbb{N}\} \quad \Leftrightarrow \quad M \text{ does not accept } w$$

This $M_2$ can be "fed" to the decider $R$.

# Non-context-free languages

There exist non-context-free languages:

Let $\Sigma = \{a, b, c\}$. The language $L \subseteq \Sigma^*$ defined as

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

is not context-free (Sipser, Ex. 2.36).

“A PDA can only pop once what it has pushed.”

# Exercise

Use direct reduction to prove that

$$CF_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is context free}\}$$

is not decidable.

# Exercise

Use direct reduction to show that

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are equivalent TMs}\}$$

is not decidable.