23. Undecidability

Goal: • Develop an understanding of which problems are not decidable and a theory of how to derive new undecidability results from known ones.

• Three results: • Universal Turing machines
  • Undecidability of the halting problem (Turing)
  • Reduction

23.1 A Universal Turing Machine

Goal: • Build a computer / an interpreter:
  Build a Turing machine
  that takes other Turing machines as input
  and simulates them.
• The existence of this universal Turing machine
  is seen as another confirmation of Church's thesis.

Approach: • First encode TMs into strings over 0, 1.
  • Then build a TM that simulates the encoding.

Encoding (of Turing machines as strings):

Let $M = (Q, \Sigma, \Gamma, q_0, \sqcup, \delta, Q_F)$.

We number the elements in Q and Γ:

$$Q = \{q_0, \ldots, q_n\}$$
$$\Gamma = \{a_0, \ldots, a_m\}.$$

We assume the numbers of initial and final states are well-distinguishable for an algorithm (e.g. the first is the initial state, odd numbers are final states) and so are the numbers for $\sqcup$ and for Σ.

With this assumption, we only have to encode the transitions.

- Every transition
$$(q_i, a_j, a_{j'}, d, q_{i'}) \in Q \times \Gamma \times \Gamma \times \{L, N, R\} \times Q$$
yields the word
$$w_{i,j,j',d,i'} := \#\# \, bin(i) \# bin(j) \# bin(j') \# bin(y) \# bin(i'),$$
$$\text{where } y := \begin{cases} 0, & \text{if } d = L \\ 1, & \text{if } d = N \\ 2, & \text{if } d = R. \end{cases}$$

Here, $bin(-)$ is the <u>binary encoding</u> of a number.

[We could have used a <u>unary encoding</u> as well,
but it is customary in computer science (in particular in complexity theory)
to represent the input in a <u>succinct</u> way
(if one blows-up the input, one gets lower complexity measures
that do not properly reflect the computational effort
(essentially, one tricks away an exponent
by making the input large)).]

- We write the words $w_{i,j,j',d,i'}$ for all transitions one after the other
and thus obtain an <u>encoding of M</u> in the alphabet $\{\#, 0, 1\}$.
- To get rid of $\#$, we apply the following homomorphism:
$$0 \mapsto 00 \qquad 1 \mapsto 01 \qquad \# \mapsto 11.$$
- Not every word over $0,1$ is the encoding of a TM.
To get around this, let $\tilde{M}$ be a fixed TM with $L(\tilde{M}) = \emptyset$.
For every $w \in \{0,1\}^*$, we define
$$M_w := \begin{cases} M, & \text{if } w \text{ is the encoding of the TM } M \\ \tilde{M}, & \text{otherwise.} \end{cases}$$

<u>Remark:</u>
The literature also uses M for a TM and $\langle M \rangle$ for its encoding in 0,

Theorem (Turing '36):

Given the above encoding, we can construct a universal TM $\mathcal{U}$ (the computer or the interpreter) with

$$\mathcal{L}(\mathcal{U}) = \{ w\#x \mid x \in \mathcal{L}(M_w) \}.$$

Note:

The universal TM is nowadays called a computer.
It is able to simulate any program that is given.
it is not restricted to fixed functionality.

→ Turing's idea of a universal TM was of enormous importance for the invention of computers.
Essentially, computers were the idea to turn Turing's machine into practice.

→ The book "Turing's Cathedral" by George Dyson is an excellent exposition of the history of computer science.

Proof (Sketch):

Given an input $w\#x$, the universal TM

(1) Checks that w indeed encodes a TM.

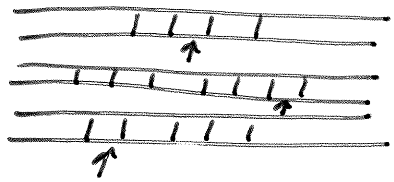
(2) Checks that x encodes a word in the TM input alphabet.

(3) If (1) or (2) fails, it rejects.
Otherwise, it accepts $w\#x$ iff M_w accepts x.

• Note that we just skipped the encoding of input words mentioned in (2)
We can either consider such an encoding
or right away assume the TM uses $0,1$ as input alphabet.

· For the simulation in (3), the behavior of U is as follows:

↳ The tape is split into <u>three tracks</u> with one head each:



It is not difficult to see that the three tracks can be simulated with one track and a single head.

↳ The description w of M_w is copied onto the top tape.

↳ The middle tape holds the initially given string x and, at runtime, the content of M_w's tape.

↳ The third track holds M_w's current state and current position of the read/write head.

Machine U simulates M_w on x one step at a time, first finding an appropriate transition (by inspecting all three tracks) and then executing the transition on Track 2 and updating Track 3.

If M_w halts and accepts, so does U. □

23.2 <u>Undecidability via Diagonalization</u>

<u>Definition:</u>

The <u>halting problem</u> is the language of U above:

$$HP := \{ w\#x,\ w,x \in \{0,1\}^* \mid x \in L(M_w)\}.$$

<u>Goal:</u> Establish Turing's famous theorem.

<u>Theorem (Turing '36):</u>

HP is semi-decidable (because of U) but <u>not decidable</u>.

As a consequence of the theorem, semi-deciders accept strictly more languages than decision procedures.
With this perspective, the undecidability result can be thought of as a huge pumping lemma.
Note that $L(U)$ is a <u>variant</u> of the famous <u>halting problem</u>:

Replace $x \in L(M_w)$ by M_w halts on x.

It is not difficult to give constructions from M_w to M_w' and also in the other direction so that

$$x \in L(M_w) \quad \text{iff} \quad M_w' \text{ halts on } x.$$

So the precise formulation of the halting problem does not really matter.

<u>Proof</u>:
Towards a contradiction, assume TM was decidable.

- Then there is a TM H that on input $w\#x$ halts and
  - ↳ accepts, if $x \in L(M_w)$ and
  - ↳ rejects, if $x \notin L(M_w)$.
- We now construct a new machine D that <u>inverts</u> the answer of H: on input w it halts and
  - ↳ rejects, if $w \in L(M_w)$ and
  - ↳ accepts, if $w \notin L(M_w)$

Clearly, D can be built using H as a subroutine:
1. Run H on $w\#w$ until acceptance or rejection.
2. Output the opposite of what H outputs.

· What happens if we run D on a description w_D of itself?

D accepts w_D iff $w_D \notin L(\underbrace{M_{w_D}}_{= D})$

iff D does not accept w_D.

↯ This is a contradiction, D and hence H cannot exist. □

The proof is indeed an instance of <u>diagonalization</u>:

TM \ Encoding	w_{M_1}	w_{M_2}	w_{M_3}	...	w_D
M_1	accept	reject	accept		
M_2	accept	accept	reject		
M_3	reject	reject	reject		
...				...	
D					?

· The entries are $H(w\#x)$.

· D inverts the diagonal entries.

· The contradiction arises at ?.

<u>Note:</u>

· We only need the diagonal to derive a contradiction.

Therefore, already the following problem is undecidable:

<u>Theorem:</u>

The <u>special halting-problem</u>

$$SHP := \{ w \in \{0,1\}^* \mid w \in L(M_w) \}$$

is undecidable.

· It is, however, not very handy having to inspect the proof of some undecidability result to conclude from this further undecidabilities.

23.3 Reduction

Goal: Derive further undecidability results from known ones:
- No more direct diagonalization proofs.
- Without having to inspect proofs.

Approach: **Reduction**

To show that a new problem B is undecidable show that it embeds, as a special case, a problem A from which we know it is undecidable. The problem A is said to reduce to B. As a consequence, a decision procedure for B would in particular work as a decision procedure for A (after we have computed the reduction). Hence, as A is not decidable, B cannot be decidable.

Definition:

Consider languages $A \subseteq \Sigma^*$ and $B \subseteq T^*$.
Then A is reducible to B, denoted by $A \leq B$,
if there is a (total and) computable function

$$f : \Sigma^* \to T^*$$

so that for all $x \in \Sigma^*$:

$$x \in A \quad \text{iff} \quad f(x) \in B.$$

Function f is called a reduction, and we read $A \leq B$ as A is at least as hard as B.

Lemma:
If $A \leq B$ and B is (semi-) decidable, then A is (semi-) decidable

The lemma is usually applied in contraposition:

If A is not decidable (semi-decidable),

then B cannot be decidable (semi-decidable).

__Proof:__

We show the case of decidability.

Consider $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$

where $A \leq B$ via reduction $f: \Sigma^* \to \Gamma^*$.

By the assumption that B is decidable,

function χ_B is computable.

The composition of two computable functions $\chi_B \circ f$

is again a computable function. (Exercise: Check this.)

We have for all $x \in \Sigma^*$:

$$\chi_A(x) = 1 \qquad (0 \text{ otherwise})$$

iff $x \in A$

(Reduction) iff $f(x) \in B$

iff $\chi_B(f(x)) = 1 \qquad (0 \text{ otherwise}).$

Hence, $\chi_A = \chi_B \circ f$ and is thus computable.

In the case of semi-decidability:

- χ_A and χ_B are replaced by χ'_A and χ'_B.
- 0 otherwise is replaced by undef otherwise. $\square$

To give an example of reductions, we define the following problem.

__HP_ε__ __(Halting on empty tape):__

__Given__: $w \in \{0,1\}^*$

__Question__: $\varepsilon \in L(M_w)$?

Lemma: $HP \leq HP_\varepsilon$.

Proof:

Given a word $w\#x$, we construct a TM M_w^x that behaves as follows:

Started on the empty tape, M_w^x first writes x onto the tape, moves to the left, and then behaves like M_w.

The function
$$f: \{0,1\}^*\#\{0,1\}^* \to \{0,1\}^*$$
$$w\#x \mapsto w_{M_w^x}$$
is computable (think of a Java program where we have a method toString for object M_w^x).

We have

$$\begin{aligned} w\#x \in HP \quad & \text{iff} \quad x \in L(M_w) \\ & \text{iff} \quad \varepsilon \in L(M_w^x) \\ & \text{iff} \quad f(w\#x) = w_{M_w^x} \in HP_\varepsilon. \end{aligned}$$
$\square$