THE UNIVERSAL TURING MACHINE

AND DIAGONALIZATION PROOFS OF UNDECIDABILITY

Abhijit Das
Sudeshna Kolay

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

March 24, 2022

One machine for all problems

- So far, we have built several machines, each solving a single problem.
- We have also embedded the finite control of a machine in the finite control of a simulating machine.
- Modern (stored-program) computers appear to be more flexible.
  - An executable file solves one problem.
  - The computer can run any executable file.
  - The executable file must be presented in a format understood by the CPU.
- Can we present the working of a Turing machine M as an executable file, and supply that executable file to the tape (not the finite control) of a Turing machine?
- The Universal Turing machine (UTM) U can do that.
- M must be **encoded** as a string that U can decode easily.
- U should also be supplied the input x for M.
- U simulates M on x by looking at the description (encoding) of M and x.
- U does not need to store the finite control of M in its own finite control.

Binary encoding of Turing machines and input strings

- Let $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ be a TM.
- Let $n = |Q|$, $m = |\Gamma|$, and $k = |\Sigma|$.
- Take $Q = \{0, 1, 2, \ldots, n-1\}$, $\Sigma = \{0, 1, 2, \ldots, k-1\}$, and $\Gamma = \{0, 1, 2, \ldots, m-1\}$.
- Then, $\vdash$ and $\sqcup$ are two different integers e and b in the range $[k, m-1]$.
- s, t, r are integers in the range $[0, n-1]$.
- All the components of M except δ can be specified by the string $0^n 1 0^k 1 0^m 1 0^e 1 0^b 1 0^s 1 0^t 1 0^r 1$.
- This is followed by the transitions of M listed one after another.
  - $\delta(p, a) = (q, b, L)$ is encoded as $0^p 1 0^a 1 0^q 1 0^b 1 0$.
  - $\delta(p, a) = (q, b, R)$ is encoded as $0^p 1 0^a 1 0^q 1 0^b 1 1$.

- Let $x = a_1 a_2 \ldots a_l \in \Sigma^*$.
- Each a_i is an integer in the range $[0, k-1]$.
- We encode x as $0^{a_1} 1 0^{a_2} 1 \ldots 1 0^{a_l} 1$.

All binary strings are encodings

- We are able to encode each TM M and each input x for M as binary strings.
- Any string w over $\{0,1\}$ can be treated it as a binary encoding of a TM M.
- If w does not correspond to a valid encoding of a TM, we assume that M is a Turing machine that, on any input, immediately rejects and halts.
- This machine is represented by all invalid strings.
- Valid encodings are also not unique (rename states/symbols, rearrange transitions).
- Any string $w \in \{0,1\}^*$ can be treated as a binary encoding of some $x \in \Sigma^*$.
- If w is invalid, we take $x = \varepsilon$.
- Multiple encodings (valid or invalid) may represent the same string x.
- Multiple encodings for a machine/string do not pose a problem.
- We can write the encoding of M and x as $\langle M \rangle$ and $\langle x \rangle$.
- By an abuse of notation, $\langle M \rangle$ and $\langle x \rangle$ are usually written as M and x.

The Universal Turing Machine U

- U is designed as a 3-tape (or 3-track) TM.
- M and x are both binary strings, so we supply both as $M \# x$ on the first tape of U.
- Without loss of generality, we may assume that M is a DTM.
- U uses its second tape to simulate the tape of M.
- U uses its third tape to store the state of M and the head position of M.
- U checks whether M given on the first tape is a valid encoding of a Turing machine.
  If not, it rejects and halts.
- U then checks whether x is a valid encoding of an input for M.
  If not, it erases x on its first tape, so x becomes ε.
- U copies x to its second tape, and s and 0 to its third tape.
- U is now ready to start the simulation of M on x.

The simulation of M on x by U

- U reads the state p of M from the third tape.
- U also knows the head position h from the third tape.
- U aligns its head to point to the h-th cell of the tape of M on its second tape.
- M reads the symbol a scanned by the head of M at position h.
- Since M is a valid encoding of a DTM, U locates the unique transition entry $\delta(p,a) = (q,b,d)$ from its first tape.
- U replaces a by b on its second tape, relocating the contents to the right if $a \neq b$.
- U replaces p by q on its third tape.
- Finally, depending upon the direction d (L or R), U changes the head position of M on its third tape.
- This completes the simulation by U of one step of M.

The language of U

- If M ever enters its accept state t, U accepts (and halts).
- If M ever enters its reject state r, U rejects (and halts).
- If M loops on x, U continues simulating the steps of M for ever.
- U is designed to detect whether M accepts x, that is, whether x is a member of $\mathscr{L}(M)$.
- U solves the **membership problem** for every TM M and for every input x for M.
- $\text{MP} = \mathscr{L}(U) = \left\{ M \# x \mid x \in \mathscr{L}(M) \right\} = \left\{ (M, x) \mid x \in \mathscr{L}(M) \right\}$.

- U may be slightly modified to U' as follows.
- If M ever enters t or r, U' accepts (and halts).
- U' solves the **halting problem** for every TM M and for every input x for M.
- $\text{HP} = \mathscr{L}(U') = \left\{ M \# x \mid M \text{ halts on } x \right\} = \left\{ (M, x) \mid M \text{ halts on } x \right\}$.

An immediate question

- The UTM solves the membership (or halting) problem by blindly simulating M on x.
- In particular, if M does not halt x, the simulation by U also does not halt.
- U is a recognizer, not a decider.
- Is there a more intelligent way to solve the problem(s)?
- In special cases, the problem can be solved without a simulation.
  - The encoding of M is invalid, so no simulation is necessary.
  - M has no transitions of the form $\delta(p,a) = (t,b,d)$.
  - M never writes a symbol A (not in $\Sigma \cup \{\vdash, \sqcup\}$) on its tape, but the only transitions that allow M to accept are of the form $\delta(p,A) = (t,b,d)$.
  - $\ldots$
- In general, there is no better way of solving the membership (or halting) problem than doing blind simulation.
- **Theorem:** MP and HP are (recursively enumerable but) **not recursive**.

HP is not recursive: Preparation for the proof

- A similar proof works for MP as well.

- $\{0,1\}^*$ is countably infinite.
- Let $\alpha_1, \alpha_2, \alpha_3, \ldots$ be an exhaustive enumeration of all the binary strings.
- Example: $\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \ldots$
- Every $\alpha \in \{0,1\}^*$ (even if invalid) is the encoding of a Turing machine M_α.
- Every $\alpha \in \{0,1\}^*$ (even if invalid) is the encoding of an input x_α for any TM.
- $\alpha_1, \alpha_2, \alpha_3, \ldots$ is an exhaustive list of all Turing machines.
- There are repetitions in the list, but there are **no other Turing machines**.
- $\alpha_1, \alpha_2, \alpha_3, \ldots$ is an exhaustive list of all inputs.
- There are repetitions in the list, but there are **no other inputs**.

A diagonalization proof

- Suppose that HP is recursive. Let D be a decider for HP.
- Given any two binary strings α_i, α_j, the hypothetical TM D decides (in finite time) whether M_{α_i} halts on x_{α_j}.
- Consider a two-dimensional table of all machines on all inputs.

	x_{α_1}	x_{α_2}	x_{α_3}	$\cdots$	x_{α_n}	$\cdots$
M_{α_1}	H	H	H	$\cdots$	H	$\cdots$
M_{α_2}	H	L	L	$\cdots$	H	$\cdots$
M_{α_3}	L	L	H	$\cdots$	L	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	
M_{α_n}	L	H	H	$\cdots$	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	
E	L	H	L	$\cdots$	L	$\cdots$

- Given any α_i and α_j, D can compute the (i, j)-th entry of the table in finite time.

Converting D to a Turing machine E

- E takes a single binary string α as input.
- E generates the input $\alpha \# \alpha$ for D.
- E simulates D on this input.
- D is a decider, so a finite-time simulation gives the answer H (accept) or L (reject).
- If D outputs H, then E forcibly enters an infinite loop (like always move right in a looping state).
- If D outputs L, then E immediately accepts and halts.
- E is a Turing machine, so can be found (at least once) in the exhaustive list of TM encodings. Let E have an encoding α_n.
- The rows marked M_{α_n} and E must be identical.
- But the rows must differ in the n-th column, a contradiction.
- So E cannot exist, and so D cannot exist too.

Tutorial exercises

1. Modify the diagonalization proof for HP to prove that MP is not recursive.
2. Use a diagonalization argument to prove that the following language is not recursive.

$$\left\{ M \# x \mid M \text{ reenters its start state on input } x \right\}$$

3. For two languages A and B over the same alphabet Σ, define the language

$$A/B = \left\{ \alpha \in \Sigma^* \mid \alpha\beta \in A \text{ for some } \beta \in B \right\}.$$

Prove that if A and B are recursively enumerable, then so also is A/B.
Prove/disprove: If A and B are recursive, then so also is A/B.

4. A *shuffle* of two strings α and β is a string γ of length $|\alpha| + |\beta|$, in which α and β are non-overlapping subsequences (not necessarily substrings). For example, all shuffles of *ab* and *cd* are *abcd*, *cabd*, *cdab*, *acbd*, *acdb*, and *cadb*. For two languages A and B, we define shuffle(A, B) as the language consisting of all shuffles of all $\alpha \in A$ and all $\beta \in B$. Prove that recursively enumerable languages are closed under the shuffle operation, that is, if A and B are r.e. languages, then so also is the language

$$\text{shuffle}(A, B) = \left\{ \gamma \mid \gamma \text{ is a shuffle of some } \alpha \in A \text{ and } \beta \in B \right\}.$$

Is shuffle(A, B) recursive if A and B are recursive? Justify.