Advanced Topics in Theoretical Computer Science

Part 1: Turing Machines and Turing Computability (2)

10.11.2021

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Last time

- Deterministic Turing Machine (DTM)
- Configuration, transition between configurations, computation
  To halt, to hang
- Representation of Turing machines
  - as in definition
  - diagram (flow-chart) representation
- Definitions: TM-computable functions
- Types of Turing machines

Universal Turing machines

Comparison between Turing machines and "normal" computer

Turing machines are very powerful.

How powerful?

- A Turing machine has a given "program" (set of rules, δ)
- "Normal" computer can execute arbitrary programs.

Universal Turing machines

Comparison between Turing machines and “normal” computer

Turing machines are very powerful.

How powerful?

- A Turing machine has a given “program” (set of rules, δ)
- “Normal” computer can execute arbitrary programs.

Actually, this is possible also with Turing machines

Universal Turing Machines

Turing machine which simulates other Turing machines

- Universal Turing machine $\mathcal{U}$ receives as input
  (i) the rules of an arbitrary TM $\mathcal{M}$ and
  (ii) a word w.
- $\mathcal{U}$ simulates $\mathcal{M}$, by always changing the configurations (according to the transition function δ) the way $\mathcal{M}$ would change them.

Universal Turing Machines

Turing machine which simulates other Turing machines

- Universal Turing machine $\mathcal{U}$ receives as input
  (i) the rules of an arbitrary TM $\mathcal{M}$ and
  (ii) a word w.
- $\mathcal{U}$ simulates $\mathcal{M}$, by always changing the configurations (according to the transition function δ) the way $\mathcal{M}$ would change them.

Problem: Turing machines take words (or numbers) as inputs.

Question:
Can we encode an arbitrary Turing machine as a number or as a word?

Universal Turing Machines

Gödelisation

Method for assigning with every Turing machine a number or a word (Gödel number or Gödel word) such that the Turing machine can be effectively reconstructed from that number (or word).

Universal Turing Machines

Gödelisation

Method for assigning with every Turing machine a number or a word (Gödel number or Gödel word) such that the Turing machine can be effectively reconstructed from that number (or word).

We can construct a universal Turing machine.

Acceptance and Decidability

We now formalize notions such as:

- Acceptable language
- Recursively enumerable language
- Enumerable language
- Decidable language

and present the relationships between these notions.

Acceptance and Decidability

Acceptance

A DTM $\mathcal{M}$ accepts a language L if

- for every input word $w \in L$, $\mathcal{M}$ halts;
- for every input word $w \notin L$, $\mathcal{M}$ computes infinitely or hangs.

Acceptance and Decidability

Acceptance

A DTM $\mathcal{M}$ accepts a language L if

- for every input word $w \in L$, $\mathcal{M}$ halts;
- for every input word $w \notin L$, $\mathcal{M}$ computes infinitely or hangs.

Deciding

A DTM $\mathcal{M}$ decides a language L if

- for every input word $w \in L$, $\mathcal{M}$ halts with band contents Y (yes)
- for every input word $w \notin L$, $\mathcal{M}$ halts with band contents N (no)

Acceptance and Decidability

Definition (Decidable language)

Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$.

Let $\mathcal{M} = (K, \Sigma, \delta, s)$ be a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ decides L if for all $w \in \Sigma_0^*$:

$$s, \#w\underline{\#} \vdash_{\mathcal{M}}^* \begin{cases} h, \#Y\underline{\#} & \text{if } w \in L \\ h, \#N\underline{\#} & \text{if } w \notin L \end{cases}$$

- L is called decidable if there exists a DTM which decides L.

Acceptance and Decidability

Definition (Acceptable language)

Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$.

Let $\mathcal{M} = (K, \Sigma, \delta, s)$ be a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ accepts a word $w \in \Sigma_0^*$ if $\mathcal{M}$ always halts on input w.
- $\mathcal{M}$ accepts the language L if for all $w \in \Sigma_0^*$, $\mathcal{M}$ accepts w iff $w \in L$.
- L is called acceptable (or semi-decidable) if there exists a DTM which accepts L.

Recursively enumerable

Definition (Recursively enumerable language)

Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$.

Let $\mathcal{M} = (K, \Sigma, \delta, s)$ be a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ enumerates L if there exists a state $q_B \in K$ (the blink state) such that:

$$L = \{w \in \Sigma_0^* \mid \exists u \in \Sigma^*; s, \underline{\#} \vdash_{\mathcal{M}}^* q_B, \#w\underline{\#}u\}$$

- L is called recursively enumerable if there exists a DTM $\mathcal{M}$ which enumerates L.

Recursively enumerable

Attention: recursively enumerable $\neq$ enumerable!

Recursively enumerable

Attention: recursively enumerable $\neq$ enumerable!

Difference:

- L **enumerable:** there exists a surjective map of the natural numbers onto L.
- L **recursively enumerable:** the surjective map can be computed by a Turing machine.

Because of the finiteness of the words and of the alphabet, all languages are enumerable. But not all languages are recursively enumerable.

Recursively enumerable

Attention: recursively enumerable $\neq$ enumerable!

Difference:

- ***L* enumerable:** there exists a surjective map of the natural numbers onto L.
- ***L* recursively enumerable:** the surjective map can be computed by a Turing machine.

Because of the finiteness of the words and of the alphabet, all languages are enumerable. But not all languages are recursively enumerable.

$\mapsto$ Set of all languages is not enumerable;
Turing machines can be enumerated.

Recursively enumerable

Attention: recursively enumerable $\neq$ decidable!

Recursively enumerable

Attention: recursively enumerable $\neq$ decidable!

Examples:

The following sets are recursively enumerable, but not decidable:

- The set of the Gödelisations of all halting Turing machines.
- The set of all terminating programs.
- The set of all valid formulae in predicate logic.

Acceptable/Recursively enumerable/Decidable

Theorem (Acceptable = Recursively enumerable)
A language is recursively enumerable iff it is acceptable.

Proposition
Every decidable language is acceptable.

Proposition
The complement of any decidable language is decidable.

Proposition (Characterisation of decidability)
A language L is decidable iff L and its complement are acceptable.

Recursively enumerable = Type 0

Formal languages are of type 0 if they can be generated by arbitrary grammars (no restrictions).

Proposition

The recursively enumerable languages (i.e. the languages acceptable by DTMs) are exactly the languages generated by arbitrary grammars (i.e. languages of type 0).

Undecidability of the halting problem

$\mathcal{M}$ Turing machine $\mapsto$ $G(\mathcal{M})$ Gödelisation

$$HALT = \{(G(\mathcal{M}), w) \mid \mathcal{M} \text{ halts on input } w\}$$

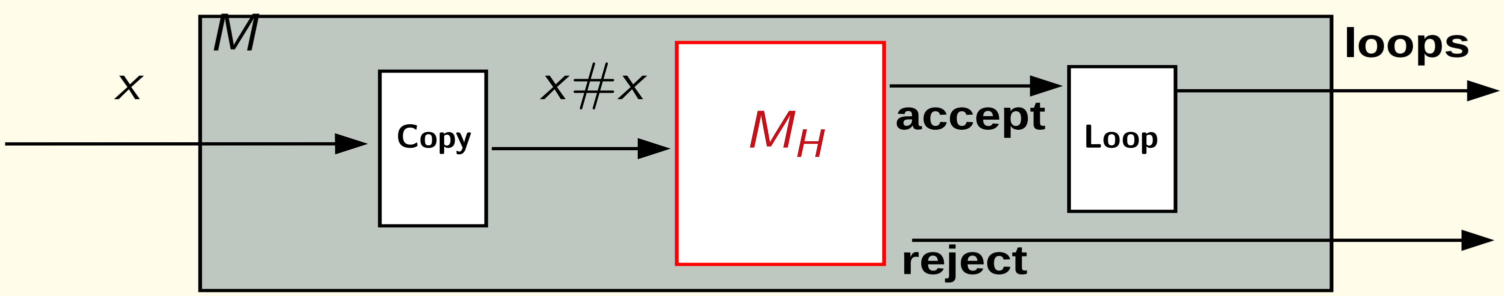
Is $HALT$ decidable?

Undecidability of the halting problem

Proposition:
$HALT = \{(G(\mathcal{M}), w) \mid \mathcal{M}$ halts on input $w\}$ is not decidable.

Proof: Assume, in order to derive a contradiction, that there exists a TM M_H which halts on every input and accepts only inputs in $HALT$.

We construct the following TM:



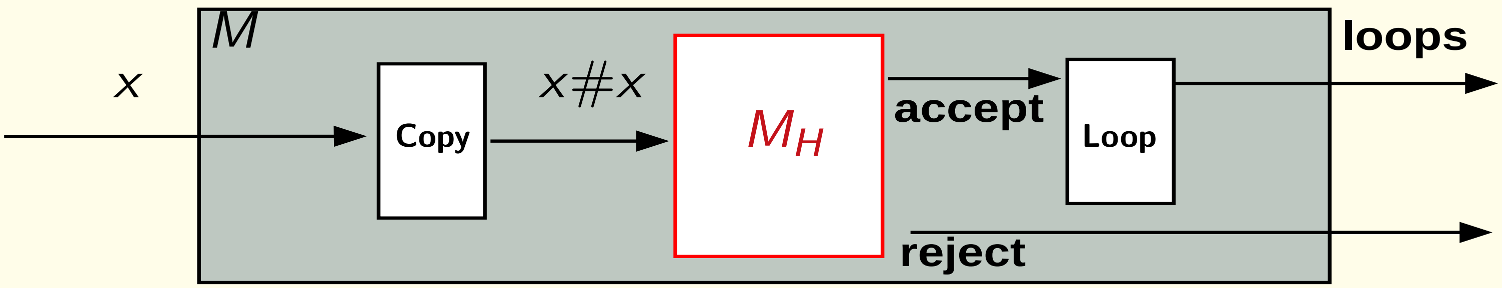
1. Let x be the input.
2. Copy the input. Let $x\#x$ be the result.
3. Decide using M_H if $(x, x) \in HALT$
4. If yes: loop
5. If no: halt

Undecidability of the halting problem

Proposition:
$HALT = \{(G(\mathcal{M}), w) \mid \mathcal{M}$ halts on input $w\}$ is not decidable.

Proof: Assume, in order to derive a contradiction, that there exists a TM M_H which halts on every input and accepts only inputs in $HALT$.

What happens when we start M with input $G(M)$?



Case 1: M started with $G(M)$ halts: Then $(G(M), G(M)) \not\in HALT$ Contradiction!

Case 2: M started with $G(M)$ does not halt: Then $(G(M), G(M)) \in HALT$ Contradiction!

Undecidability proofs: Example

Theorem. $K = \{G(M) \mid M \text{ halts for input } G(M)\}$
is acceptable but undecidable.

Proof: Undecidable: Similar to the undecidability proof for $HALT$.

Acceptable: $M_K := M_{\text{prep}}\mathcal{U}$,
($\mathcal{U}$ universal TM; M_{prep} brings tape in form required by $\mathcal{U}$).

Reformulation using numbers instead of words:

Gödelization $\mapsto$ Gödel numbers

Let $M_0, M_1, \ldots, M_n, \ldots$ be an enumeration of all Turing Machines

M_n is the TM with Gödel number n.

$$K = \{n \mid M_n \text{ halts on input } n\}$$

Today

- How to prove that a language is undecidable?

Undecidability proofs

Proof via reduction

- L_1, L_2 languages
- L_1 known to be undecidable
- To show: L_2 undecidable
- Idea: Assume L_2 decidable. Let M_2 be a TM which decides L_2. Show that then we can construct a TM which decides L_1.

  For this, we have to find a computable function f which transforms an instance of L_1 into an instance of L_2

  $$\forall w (w \in L_1 \text{ iff } f(w) \in L_2)$$

  Let M_f be the TM which computes f. Construct $M_1 = M_f M_2$. Then M_1 decides L_1.

Undecidability proofs

Proof via reduction

Definition. L_1, L_2 languages. $L_1 \leq L_2$ (L_1 is reducible to L_2) if there exists a computable function f such that:

$$\forall w (w \in L_1 \text{ iff } f(w) \in L_2)$$

Theorem. If $L_1 \leq L_2$ and L_1 is undecidable then L_2 is undecidable.

Undecidability proofs: Example

Theorem. $H_0 = \{n \mid M_n \text{ halts for input } 0\}$ is undecidable.

Proof: We show that K can be reduced to H_0, i.e. that there exists a TM computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$i \in K \quad \text{iff} \quad f(i) \in H_0.$$

Only main idea here, we will come back to this example later

Undecidability proofs: Example

Theorem. $H_0 = \{n \mid M_n \text{ halts for input } 0\}$ is undecidable.

Proof: We show that K can be reduced to H_0, i.e. that there exists a TM computable function $f : \mathbb{N} \to \mathbb{N}$ such that $i \in K$ iff $f(i) \in H_0$.

Want: $f(i) = j$ iff (M_i halts for input i iff M_j halts for input 0).

For every i there exists a TM A_i s.t.: $s, \#\underline{\#} \vdash^*_{A_i} h, \#|^i\underline{\#}$.
Let M_K be the TM which accepts K.

We define $f(i) := j$ where j is the Gödel number of $M_j = A_i M_K$.
f is TM computable. We show that f has the desired property:

$$f(i) = j \in H_0 \quad \text{iff} \quad M_j = A_i M_K \text{ halts for input } 0\ (\#\underline{\#})$$
$$\text{iff} \quad M_K \text{ halts for input } i \quad \text{iff} \quad i \in K.$$