Computable and computably enumerable languages

Peter Mayr

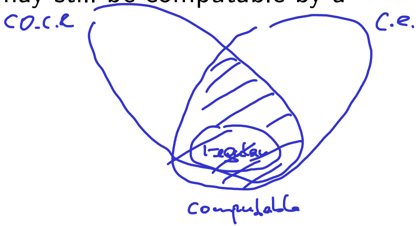
Computability Theory, September 13, 2023

Definition

- A DTM M with input alphabet Σ is **halting** if M halts on every $w \in \Sigma^*$.
- If M is halting, it **decides** its language $L(M)$.
- L is **computable** (also decidable, recursive) if there exists a halting DTM M such that $L = L(M)$.
- L is **computably enumerable (c.e.)** (also semi-decidable, recursively enumerable) if there exists a DTM M such that $L = L(M)$.

Note

- Even if M is not halting, $L(M)$ may still be computable by a different DTM.
- regular $\Rightarrow$ computable $\Rightarrow$ c.e.



Theorem

L is computable iff L and its complement $\overline{L}$ is c.e.

Proof.

$\Rightarrow$: Let $L = L(M)$ for a halting DTM M.

- Then L is c.e. by definition.
- Also $\overline{L} = L(M')$ is c.e. with M' like M but with accept and reject state flipped.

$\Leftarrow$: Let $M_1 = (Q_1, \ldots, \delta_1), M_2 = (Q_2, \ldots, \delta_2)$ be DTMs with $L = L(M_1), \overline{L} = L(M_2)$.

Construct M to run M_1, M_2 in parallel on input w:

- states $Q_1 \times Q_2$
- tape alphabet $\Gamma_1 \times \Gamma_2$
- transition function $\delta_1 \times \delta_2$ acting on 2 tapes
- accept states $\{t_1\} \times Q_2$ (M_1 accepts)
- reject states $Q_1 \times \{t_2\}$ (M_2 accepts)

Then M is halting and $L(M) = L$. □

Since each $w \in \Sigma^*$ is either in L or $\overline{L}$, either M_1 or M_2 accepts in finitely many steps.

Closure properties of computable languages

Theorem

The class of computable languages is closed under complements, union, intersection, concatenation, *.

Proof.

Construct the corresponding DTMs. □

Question

Which operations preserve c.e. languages?

Why "enumerable"?

Definition

An **enumerator** is a DTM M with $\sharp \in \Gamma$,

- a working tape and
- an **output tape** on which M moves only right (or stays) and writes only symbols from $\Gamma \setminus \{\sqcup\}$. printer

The **generated language** $\mathrm{Gen}(M)$ of M is the set of all words that M writes on the output tape when starting with empty tapes. Consecutive words are separated by $\sharp$.

Example

If M writes $\sharp 1 \sharp 11 \sharp 111 \sharp \ldots$, then $\mathrm{Gen}(M) = L(\epsilon, 1, 11, \ldots)$.

Theorem

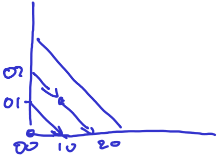
L is c.e. iff there exists an enumerator with $L = \text{Gen}(M)$.

Proof.

$\Rightarrow$: Let $L = L(N)$ for a DTM N.

Idea: Construct an enumerator M that runs through all $w \in \Sigma^*$ and prints w if N accepts it.

M loops through all pairs $(m, n) \in \mathbb{N}^2$ (countable!):



- For (m, n), M construct the m-th word w_m over Σ in length-lex order. (linear on Σ^*)
- Then N runs $\leq n$ steps with input w_m. If N accepts, then M prints w_m.

Then $\text{Gen}(M) = L(N)$.

Proof.

$\Leftarrow$: Let $L =$Gen(M) for an enumerator M.
The following DTM N accepts L:

- On input w, N starts M to enumerate L.
- If w appears in output of M, N accepts w.
- Else, N loops. □

Note

- Being able to generate a language L is equivalent to being able to accept L (but not necessarily to reject its non-elements).
- Generating L is "easier" than deciding L.

Why "computable"?

For sets $X \subseteq A$ and B we call $f\colon X \to B$ a **partial function** from A to B with $\text{domain}(f) = X$, denoted $f\colon A \to_p B$.

Example

$\sqrt{x}$ can be viewed as partial function $\mathbb{R} \to_p \mathbb{R}$ with domain $\mathbb{R}_0^+$.

Definition

A partial function $f\colon \Sigma^* \to_p \Sigma^*$ is **computable** if there exists a DTM M such that $\forall x, y \in \Sigma^*$: $(s, x\sqcup\ldots, 0) \vdash_M^* (t, y\sqcup\ldots, 0)$ iff $x \in \text{domain}(f)$ and $f(x) = y$.

Theorem

$f : \Sigma^* \to_p \Sigma^*$ is computable iff its graph

$$L_f := \{(x, y) \in (\Sigma^*)^2 \;:\; x \in \text{domain}(f), f(x) = y\}$$

is c.e.

Proof.

$\Rightarrow$: HW

$\Leftarrow$: Assume L_f =Gen(N) for some enumerator N.

Construct M that computes $f(x)$ as follows:

- M starts N to enumerate all pairs $(a, b) \in L_f$.
- If (x, y) appears for some y, then M returns y.
- Else M loops. □

Note

Computing a function is the same as accepting its graph.