Automata Theory :: Recursively Enumerable Languages

Jörg Endrullis

Vrije Universiteit Amsterdam

Recursively Enumerable Languages

A language L is **recursively enumerable** if L is accepted by a (deterministic) Turing machine.

Equivalently, a language L is recursively enumerable if there exists a Turing machine **enumerates all words** in L.

Intuitively, the Turing machine writes on the tape

$$\#w_1\#w_2\#w_3\#\cdots$$

such that $L = \{ w_1, w_2, w_3, \dots \}$.

If L is infinite, this computation never stops! Every word from L will eventually be written on the tape.

Then $w_1, w_2, \dots$ is called a **recursive enumeration** of L.

Turing Machines are Recursively Enumerable

Theorem

Turing machines are recursively enumerable.

Proof.

- A Turing machine can be represented as a word.
- A parser can check whether a word represents a TM.
  (If so, accept.)

□

Thus, there is a recursive enumeration of all Turing machines:

$$M_1, M_2, \ldots$$

Formally, we enumerate words describing Turing machines. But the does not matter since we have a universal Turing machine that can execute these descriptions.

Properties of Recursively Enumerable Languages

Union and Intersection

Theorem

If L_1 and L_2 are recursively enumerable languages, then so are

$$L_1 \cup L_2 \qquad\qquad L_1 \cap L_2$$

Let M_1 and M_2 be Turing machines such that

$$L(M_1) = L_1 \qquad\qquad L(M_2) = L_2$$

Create a Turing machine M that runs M_1 and M_2 in parallel.
(Alternating simulate one step of M_1 and one step of M_2.)

- For $L(M) = L_1 \cup L_2$,
  let M accept as soon as M_1 accepts or M_2 accepts.
- For $L(M) = L_1 \cap L_2$,
  let M accept as soon as both M_1 and M_2 accept.

What about $L_1 \setminus L_2$?

Complement

There exist recursively enumerable languages L, for which the complement $\overline{L}$ is not recursively enumerable.

Proof.

Let $M_1, M_2, M_3, \ldots$ be a recursive enumeration of all TMs.

Define the language L by

$$L = \{ a^i \mid a^i \in L(M_i), i \geq 1 \}$$

Then L is recursively enumerable.

If $\overline{L}$ was recursively enumerable: $\overline{L} = L(M_k)$ for some $k \geq 1$.

Then

$$a^k \in \overline{L} \iff a^k \in L(M_k) \iff a^k \in L$$

Contradiction. Hence $\overline{L}$ is not recursively enumerable. $\square$

$L_1 \setminus L_2$ is not always recursively enumerable since $\overline{L} = \Sigma^* \setminus L$.

Concatenation

Theorem

If L_1 and L_2 are recursively enumerable languages, then so is

$$L_1 L_2$$

Let M_1 and M_2 be Turing machines such that

$$L(M_1) = L_1 \qquad\qquad L(M_2) = L_2$$

A split of a word w is a pair (w_1, w_2) such that $w = w_1 w_2$.

We call the split good if M_1 accepts w_1 and M_2 accepts w_2.

Create a Turing machine N that

- computes all splits of the input word w
- checks **all splits in parallel** whether they are good
- accepts the input w as soon as a good split is found

Then $L(N) = L_1 L_2$.

Kleene Start

Theorem

If L is a recursively enumerable language, then so is

$$L^*$$

Let M be a Turing machine such that $L(M) = L$.

A partitioning of a word $w \neq \lambda$ are non-empty words $(w_1, \ldots, w_n)$ for some $n \leq |w|$ such that $w = w_1 w_2 \cdots w_n$.

The partitioning is good if M accepts all words $w_1, \ldots, w_n$.

Create a Turing machine N that

- computes all partitionings of the input word $w \neq \lambda$
- checks **all partitionings in parallel** whether they are good
- accepts the input w as soon as a good partitioning is found

Then $L(N) = L^*$.

Recursive Languages

Recursive Languages

A language L is **recursive** if

- L is recursively enumerable, and
- $\overline{L}$ is recursively enumerable.

Not every recursively enumerable language is recursive!

(See the a few slides back.)

Theorem

A language L is recursive $\iff$ L is accepted by a deterministic TM M that reaches for every input a halting state.

Proof on the next slide.

Proof

$(\Leftarrow)$ L accepted by deterministic TM M that always halts.

We show that $\overline{L}$ is recursively enumerable.

From M we construct a Turing machine N as follows:

- Add a fresh state q_f.
- For all $q \in Q \setminus F$ and $a \in \Gamma$:
  if $\delta(q, a)$ undefined, define $\delta(q, a) = (q_f, a, R)$.
- Make q_f the only final state.

Then $L(N) = \overline{L(M)}$, thus $L(M)$ is recursive.

$(\Rightarrow)$ L and $\overline{L}$ accepted by deterministic TMs M_1 and M_2.

Construct a TM M executes M_1 and M_2 in parallel:

- M accepts when M_1 accepts
- M has non-accepting halting state when M_2 accepts

Then $L(M) = L(M_1) = L$, and M halts for every input.

Properties of Recursive Languages

Theorem

If L, L_1 and L_2 are recursive, then so are

$$\overline{L} \qquad L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1 \setminus L_2 \qquad L^* \qquad L_1 L_2$$

Proof.

Let L, $\overline{L}$, L_1, $\overline{L_1}$, L_2 and $\overline{L_2}$ be recursively enumerable (r.e.).

- $\overline{L}$: $\overline{L}$ and $\overline{\overline{L}} = L$ are r.e.
- $L_1 \cup L_2$: $L_1 \cup L_2$ and $\overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$ are r.e.
- $L_1 \cap L_2$: $L_1 \cap L_2$ and $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$ are r.e.
- $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$
- $L_1 L_2$, L^*: same proof as for recursively enumerable languages. Observe that the constructed Turing machine halts on all inputs if M_1, M_2 and M do.

$\square$