Theory of Computation CS F351 (2020-11-15 at 22:30 GMT-8)

Definitions

Let $\sum_0$ be $\sum$ - {blank, left-end-marker}

1. We say that TM M ***decides*** a language $L \subseteq \sum_0^*$ if for any string $w \in \sum_0^*$ the following is true: **If $w \in L$ then M accepts w; if $w \notin L$ then M rejects w.**
2. A language L is recursive or Turing-decidable or decidable if there is a TM that decides it.
3. We say that TM M ***semi-decides*** a language $L \subseteq \sum_0^*$ if for any string $w \in \sum_0^*$ the following is true: **$w \in L$ iff M halts on input w.**
4. A language L is recursively enumerable or Turing recognizable iff there is a TM that semi-decides it.

Note that in the above definitions no guarantee is given if input contains blank symbol or left end marker symbol.

Example of recursively enumerable languages ??

Vishal Gupta

1:35 / 53:22

Closed under Union ??

"Decidable languages are closed under UNION operation". TRUE/FALSE.

Let L1 and L2 be two languages and M1 and M2 be respective deciders for them. Construct a machine M as follows which decides L1 U L2:

M on input x

- Simulate M1 on x. If M1 accepts, M halts in Accept state. Else,
- Simulate M2 on x. If M2 accepts, M halts in Accept state; else M halts in Reject state.

Vishal Gupta

Decidable and Recognizable Languages

A basic connection between Turing-recognizable and Turing-decidable languages:

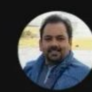
Theorem: L is Turing decidable if and only if L and L^c are both Turing-recognizable.

- Proof:
  - ⇒ – Suppose that L is Turing-decidable.
  - Then L is Turing-recognizable
  - Also, L^c is Turing-decidable
  - So L^c is Turing-recognizable

L is decidable

L^c is decidable.

- Proof:
  - ⇐ – Given M1 recognizing L, and M2 recognizing L^c.
  - – Produce a Turing Machine M that decides whether or not its input w is in L or L^c.



Decidable and Recognizable Languages

A basic connection between Turing-recognizable and Turing-decidable languages:

Theorem: L is Turing decidable if and only if L and L^c are both Turing-recognizable.

- Proof:
  - ⇐ – Given M1 recognizing L, and M2 recognizing L^c.
  - Produce a Turing Machine M that decides whether or not its input w is in L or L^c.
  - – Idea: Run both M1 and M2 on w.
    - One must accept.
    - If M1 accepts, then M accepts.
    - If M2 accepts, then M rejects.
  - – But, we can't run M1 and M2 one after the other because the first one might never halt.
  - – Run them in parallel, until one accepts?
  - – How? We don't have a parallel Turing Machine model.

Decidable and Recognizable Languages

A basic connection between Turing-recognizable and Turing-decidable languages:

Theorem: L is Turing decidable if and only if L and L^c are both Turing-recognizable.

- Proof:
  - ⇐ – Given M1 recognizing L, and M2 recognizing L^c.
    - – Produce a Turing Machine M that decides whether or not its input w is in L or L^c.
  - M = on input w:
  - 1. Run both M1 and M2 on input w in parallel.
  - 2. If M1 accepts, M accepts; if M2 accepts, M rejects.