Recursive and Recursively Enumerable Languages

Prof. (Dr.) K.R. Chowdhary
Email: kr.chowdhary@iitj.ac.in

Formerly at department of Computer Science and Engineering
MBM Engineering College, Jodhpur

Monday 10th April, 2017

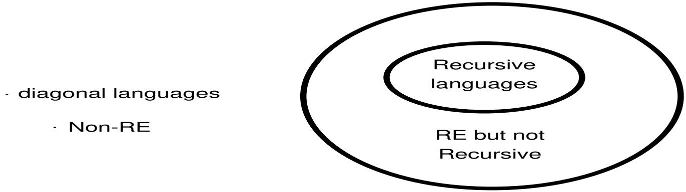
Defining R and RE languages

- Recursive: They allow a function to call itself. Or, a recursive language is a recursive subset in the set of all possible words over alphabet Σ of that language.
- Non-recursive should not be taken as simpler version of computation, i.e., e.g., obtaining factorial value without recursion method.
- Regular languages $\subseteq$ context free languages $\subseteq$ context sensitive languages $\subseteq$ recursive languages $\subseteq$ recursive enumerable languages.
- A language is Recursively Enumerable (RE) if some Turing machine accepts it.

- A *TM M* with alphabet Σ *accepts L* if $L = \{w \in \Sigma^* | M \text{ halts with input w }\}$
- Let L be a *RE* language and M the Turing Machine that accepts it. $\therefore$, for $w \in L$, M halts in final state. For $w \notin L$, M halts in non-final state or *loops for ever*.

- A language is Recursive (R) if some Turing machine M recognizes it and halts on every input string, $w \in \Sigma^*$. *Recognizable* $=$ *Decidable*. Or A language is recursive if there is a membership algorithm for it.
- Let L be a recursive language and M the Turing Machine that accepts (i.e. recognizes) it. For string w, if $w \in L$, then M halts in final state. If $w \notin L$, then M halts in non-final state. (*halts* always!)

Relation between Recursive and RE languages



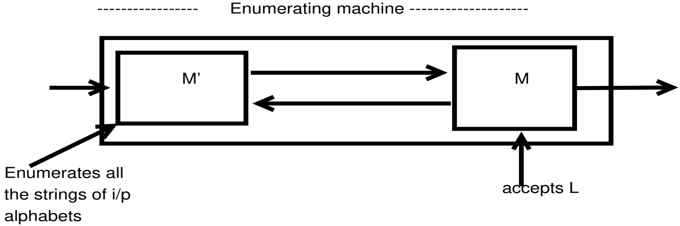
- Every *Recursive* language is *RE*. ∴, if M is *TM* recognizing L, the M can be easily modified so its accepts L.
- The languages which are non-RE cannot be recognized by TM. These are diagonal (L_d) languages of the diagonal of $x - y$, where x_i is language string w_i, and y_i is TM M_i.
- Language $< M, w >$, where M is *TM* and w is string, is not *RE* language, since its generalized form is not Turing decidable (undecidability proof), ∴, it is *non-RE* language.

Theorem

If a language L is recursive then there exists an enumeration procedure for it.

Proof.

- If $\Sigma = \{a, b\}$, then M' can enumerate strings: $a, b, aa, ab, ba, bb, aaa, \ldots$.



- Enumeration procedure: M' generates string w. M checks, if $w \in L$; if yes, output w else ignore w.
- Let $L = \{a, ab, bb, aaa, \ldots\}$. M' output $= \{a, b, aa, ab, ba, bb, aaa, \}$; $L(M) = \{a, ab, bb, aaa, \ldots\}$; enumerated output $= a, ab, bb, aaa, \ldots$

Class of Languages

- recursive = decidable, their *TM* always halts
- recursive enumerable (semi-decidable) but not recursive = their *TM* always halt if they accept, otherwise halts in non-final state or loops.
- non-recursively enumerable (*non-RE*) = there are no *TMs* for them.

Recursive languages are closed under complementation.

Theorem

If L is recursive then $\bar{L}$ is also recursive.

Proof.

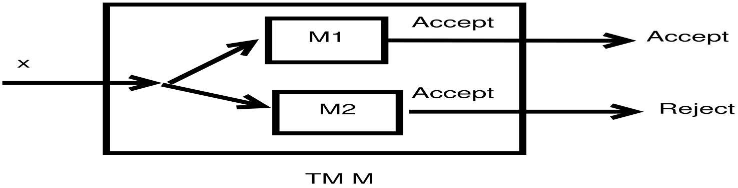
- The accepting states of M are made non-accepting states of M' with no transitions, i.e., here M' will halt without accepting.
- If s is new accepting state in M', then there is no transition from this state.
- If L is recursive, then $L = L(M)$ for some *TM* M, that always halts. Transform M into $M´$ so that $M´$ accept when M does not and vice-versa. So $M´$ always halts and accepts $\bar{L}$. Hence $\bar{L}$ is recursive.

Theorem

If L and $\overline{L}$ are RE, then L is recursive.

Proof.

- Let $L = L(M_1)$ and $\overline{L} = L(M_2)$. Construct a *TM* M that simulates M_1 and M_2 in parallel, using two tapes and two heads. If i/p to M is in L, then M_1 accepts it and halts, hence M accepts it and halts. If input to M is not in L, hence it is in $\overline{L}$, $\therefore$, M_2 accepts and halts, hence M halts without accepting. Hence M halts on every i/p and $L(M) = L$. So L is recursive.



Closure Properties:

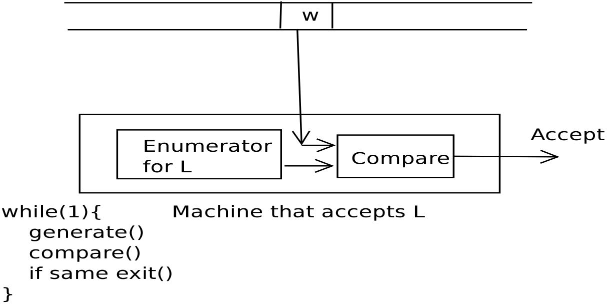
Recursive languages are closed under union, concatenation, intersection and Kleene star, complement, set difference $(L_1 - L_2)$

Theorem

A language L is recursive enumerable iff there exists an enumeration procedure for it.

Proof.

- If there is an enumeration procedure, then we can enumerate all the strings, and compare each with w each time till it is found.
- If the language is RE, then we can follow an enumerature procedure to systematically generate all the strings.



Intersection of RE and R languages

- Given a *Recursive* and a *RE* languages: Their Union is *RE*, Intersection is *RE*, Concatenation is *RE*, and Kleene's closure is *RE*.
- if L_1 is *Recursive* and L_2 is *RE*, then $L_2 - L_1$ is *RE* and $L_1 - L_2$ is not *RE*.

Theorem

The intersection R and RE languages is RE.

Proof.

- *Let* L_1 *and* L_2 *be languages recognized by Turing machines* M_1 *and* M_2*, respectively.*
- *Let a new TM* $M_\cap$ *is for the intersection* $L_1 \cap L_2$. $M_\cap$ *simply executes* M_1 *and* M_2 *one after the other on the same input* w*: It first simulates* M_1 *on* w*. If* M_1 *halts by accepting it,* $M\cap$ *clears the tape, copies the input word* w *on the tape and starts simulating* M_2*. If* M_2 *also accepts* w *then* $M_\cap$ *accepts.*
- *Clearly,* $M_\cap$ *recognizes* $L_1 \cap L_2$*, and if* M_1 *and* M_2 *halt on all inputs then also* $M_\cap$ *halts on all inputs.*

□

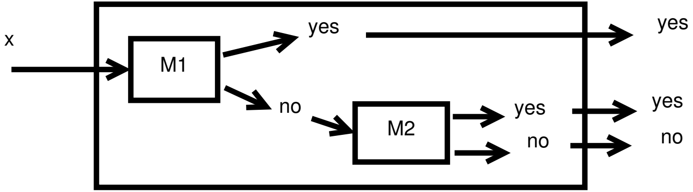
closure properties . . .

Theorem

The union of two Recursive languages is recursive.

Proof.

- *The TM corresponding to this must halt always. Let L_1 and L_2 be sets accepted by M_1 and M_2, respectively. Then $L_1 \cup Ł_2$ is accepted by TM M, where $x = w_1 \cup w_2$, for $w_1 \in L_1$ and $w_2 \in L_2$.*



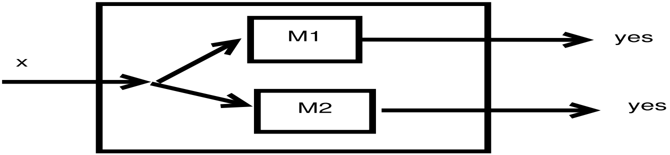
□

Theorem

The union of two RE languages is RE.

Proof.

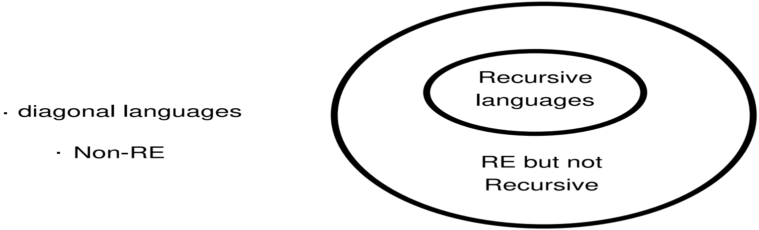
- *Let L_1 and L_2 be sets accepted by M_1 and M_2, respectively. Then $L_1 \cup L_2$ is accepted by TM M, where $x = w_1 \cup w_2$, for $w_1 \in L_1$ and $w_2 \in L_2$.*
- *To determine if M_1 or M_2 accepts x we run both M_1 and M_2 simultaneously, using a two-tape TM M. M simulates M_1 on the first tape and M_2 on the second tape. If either one enters the final state, the input is accepted.*



TM M

□

Summary of R and RE



- Both L and $\overline{L}$ are recursive, then both are in the inner circle. *Palindrome* and *CFG* are recursive.
- Neither L or $\overline{L}$ are RE, the both are outside the outer ring.
- L is RE but not recursive, and $\overline{L}$ is *non-RE*; then first is in outer circle, and second is in outer most space.
- There are languages which are neither recursive nor RE (Ref: Countable algorithms(TM) but uncountable languages)
- Closure of recursive language in $L_1 - L_2$ follows from the fact that these set difference can be expressed in terms of intersection and complement.
- **Weak Result:** If a language is recursive then there is an enumeration procedure.
- **Strong Result:** A language is RE iff there is an Enumeration