4CS4-6: Theory of Computation (Recursive and Recursively Enumerable Langs.)

Lecture 23: April, 15, 2019

Prof. K.R. Chowdhary *: Professor of CS*

23.1 Introduction

A language acceptable by Turing machine is called *Recursively Enumerable* (RE), which means that the set of strings in the language accepted by the Turing Machine can be enumerated. Recursively enumerable language is a type of formal language which is also called *partially decidable* or *Turing-recognizable*. It is known as *type-0* language in the Chomsky hierarchy of formal languages. The RE languages are always countably infinite. The class of RE languages has a broad coverage of languages, and they include some languages, which cannot be defined by a mechanical algorithm. TMs will fail to halt on some input not in these languages. If w is a string in RE language then the TM will eventually halt on w.

There exists three equivalent major definitions for the concept of a RE language.

Definition 23.1 RE Language. *A RE formal language is a recursively enumerable subset in the set of all possible words over the alphabet of the language.*

Definition 23.2 RE Language. *A RE language is a formal language for which there exists a Turing machine (or other computable function) which will enumerate all valid strings of the language.*

Note that, if the language is infinite, the enumerating algorithm provided can be chosen so that it avoids repetitions, since we can test whether the n-th string produced is "already" produced for some number less than n. If it is already produced, use the output for input $n + 1$ instead (recursively), but again, test whether it is "new".

Definition 23.3 RE Language. *A RE language is a formal language for which there exists a Turing machine (or other computable function) that will halt and accept when presented with any string in the language as input. But may either halt and reject or loop forever when presented with a string not in the language.*

Contrast this to **recursive** languages, which require that the Turing machine halts in all cases.

All regular, context-free, context-sensitive and recursive languages are recursively enumerable.

However, if M is still running on some input, we can never tell whether M will ultimately accept if it is allowed to run long enough or M will run forever. Therefore, it is appropriate to separate the subclass of RE languages accepted by at least one TM that halts on all inputs. However, halting may or may not be preceded by acceptance.

23.2 Recursive and Recursively Enumerable Languages

Definition 23.4 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.*

A language is *Recursive (R)* if some Turing machine M recognizes it and halts on every input string, $w \in \Sigma^*$. Note that Recognizable is equal to 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!).

Non-recursive should not be taken as simpler version of computation, i.e., e.g., obtaining factorial value without recursion method. We have the relation:

$$\begin{aligned}\text{Regular languages} \subseteq CF \text{ languages} \subseteq CS \text{ languages} \\ \subseteq R \text{ languages} \subseteq RE \text{ languages.}\end{aligned} \tag{23.1}$$

Definition 23.5 Recursively Enumerable. *A language is* Recursively Enumerable *(RE) if some Turing machine accepts it.*

A Turing Machine M with alphabet Σ, *accepts* language L if,

$$L = \{w \mid w \in \Sigma^* \text{ and } M \text{ halts with input } w\}. \tag{23.2}$$

Let L be a RE language and M the Turing Machine that accepts it. Therefore, for $w \in L$, the TM M halts in final state, and for $w \notin L$, M halts in non-final state or *loops for ever.*

Relation between Recursive and RE Languages Every Recursive (R) language is Recursively Enumerable (RE). Therefore, if M is Turing Machine recognizing language L, which is R, then M can be easily modified so its accepts language that is RE. 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. (see Fig. 23.1). Language $\langle M, w\rangle$, where M is TM and w is string, is not RE language, since its generalized form is not Turing decidable (undecidability proof), therefore, it is *non-RE* language. Every is recursive language can be enumerated. The following theorem proves this.

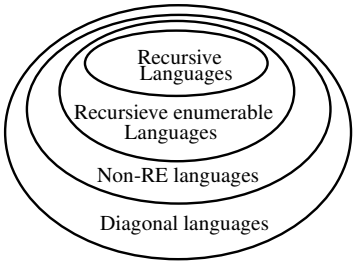

Figure 23.1: Relation between R and RE.

Theorem 23.6 *If a language L is recursive then there exists an enumeration procedure for it.*

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

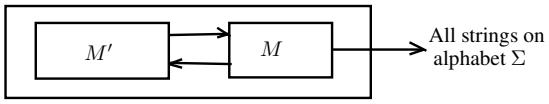

Figure 23.2: Enumeration of language.

The Enumeration procedure is as follows. Let M' generates string w. Next, the TM M checks, if $w \in L$ is yes, output w else ignore w. Let $L = \{a, ab, bb, aaa, \ldots\}$. The M' output $= \{a, b, aa, ab, ba, bb, aaa, \}$, $L(M) = \{a, ab, bb, aaa, \ldots\}$, and enumerated output is $a, ab, bb, aaa, \ldots$ ■