Recursive and R.e. Languages

We examine the languages of TMs that do and don't halt.

Recursive and Recursively Enumerable

A language is ***recursively enumerable*** (***r.e.***) if it is the set of strings accepted by some TM.

A language is ***recursive*** if it is the set of strings accepted by some TM that halts on *every* input.

For example, any regular language is recursive.

Closure Facts

Fact. *(a) The set of r.e. languages is closed under union and intersection.*

(b) The set of recursive languages is closed under union and intersection.

We prove (a) for union.

Say languages L_1 and L_2 are r.e., accepted by TMs M_1 and M_2. A TM for $L_1 \cup L_2$ simply runs M_1 and M_2 *in parallel*. The input is in the union exactly when at least one machine halts and accepts.

Recursive, R.e., and Complements

Theorem. *A language is recursive if and only if both it and its complement are r.e.*

- If L is recursive, then so is its complement (interchange states h_a and h_r).
- Assume both L and $\bar{L}$ are r.e.; that is, they have TMs. Then run the two TMs in *parallel*. At least one will halt, and that gives us the answer.

A Printer Turing Machine

A ***printer-TM*** is TM with an added printer-tape. The printer-TM writes strings on the printer-tape (separated by Δ); once written, a string is not altered.

Theorem. *A language is r.e. if and only if some printer-TM outputs precisely those strings.*

The proof is in two constructions. . .

From Printer-TM to Standard TM

Armed with printer-TM M for language L, we build standard TM N.

On input x, TM N runs M and monitors M; if N ever finds x on the printer-tape, then N accepts. So N accepts strings in L, and does not halt otherwise.

From Standard TM to Printer-TM

Armed with standard TM N for L, we build printer-TM M.

The idea is to run N on every possible string in parallel—an infinite number of tasks!—and print out those it accepts.

Infinite Parallelism

The printer-TM works M in rounds.

In round i, M starting from scratch, generates the first i strings lexicographically (in dictionary order), runs N on each for i steps, and outputs any string that is accepted.

Eventually, every string in $L(N)$ will be generated and N run for long enough, and will appear in the output.

Still to Come

We will show later that there are many r.e. languages that are not recursive, and many languages that are not even r.e.

Practice

Show that the set of recursive languages is closed under reversal.

Solution to Practice

The questions asks one to show that, if L is recursive, then so is $L^R = \{ x^R : x \in L \}$.

If L is decided by TM M, then L^R can be decided by a TM that simply reverses the input and then calls M.

Summary

Recursive languages are accepted by TMs that always halt; r.e. languages are accepted by TMs. These two families are closed under intersection and union. If a language is recursive, then so is its complement; if both a language and its complement are r.e., then the language is recursive. There is a connection with printer-TMs.