## Closure Properties for Turing Machines (9.1, 9.2)

Recall, an input x on TM M. M can

- Halt and accept x, $x \in L(M)$
- Halt and reject x, $x \notin L(M)$
- Crash, $x \notin L(M)$
- Run forever, $x \notin L(M)$

This defines 2 different classes of languages:

TM M **accepts** language L if L = L(M).

- M accepts x if and only if $x \in L$
- May loop forever

TM M **decides** language L if L = L(M) and if $x \notin L$, M rejects or crashes on x.

- M always stops
- No infinite looping

A language is **recursive (or decidable)** if there exists a TM M that decides L.

A language is **recursively enumerable** if there exists a TM M that accepts L.

If L is *recursive* then L is also *recursively enumerable*.

- A TM that *decides* L also *accepts* L.

If L is recursive then the complement L' is also recursive.
TM for L': Run x on M (the TM that decides L)

- If M accepts x then reject x.
- If M rejects or crashes, then accept x.

**Union and Intersection**

**Recursive**

If $L_1$ is recursive and $L_2$ is recursive then $L_1 \cup L_2$ and $L_1 \cap L_2$ are also recursive.
Use a multitape TM:

- Copy input to tape 2 and tape 3
- Execute $M_1$ on tape 2 and $M_2$ on tape 3 (neither will run forever; i.e. we get a result)
- They will decide whether x is in $L_1$ and/or $L_2$
- Test if both $M_1$ and $M_2$ accepted (intersection)

- Test if one of $M_1$ and $M_2$ accepted (union)

If $L_1$ and $L_2$ are recursive then the difference $L_1 - L_2 = L_1 \cap L_2'$ is recursive.

**Recursively Enumerable**

If $L_1$ and $L_2$ are recursively enumerable then $L_1 \cup L_2$ and $L_1 \cap L_2$ are recursively enumerable.

- Similar to the recursive case but need to handle the case where $M_1$ and $M_2$ can run forever.
- Simulate $M_1$ and $M_2$ running simultaneously – alternate one step from each machine.

For example, union

- If either machine ever accepts then accept
- If either machine ever rejects or crashes then continue to work on the other machine.

If L is recursively enumerated and L' is recursively enumerable then L is recursive.

- Let M and M' be TMs that accept L and L', respectively.
- Run M and M' simultaneously.
- For any word x, it must be accepted by one of M or M'
- So, either M or M' will halt and accept
- If M halts and accepts then halt and accept
- If M' halts and accepts then halt and reject

The TM that runs M and M' simultaneously always halts and accepts or rejects so it decides L and L is recursive.

If L is recursively enumerable and L is not recursive then L' is not recursively enumerable.

**Some Interesting Languages**

E = { e(T) | T is a TM }

- Create a TM to check if e(T) is a valid encoding of a TM

LSA = { e(T) | TM T accepts on input e(T) }

LNSA = { e(T) | T does not accept input e(T) }

LH = { e(T)Δz | TM accepts input z }