Recursive and Recursively Enumerable Languages

COMP 3331

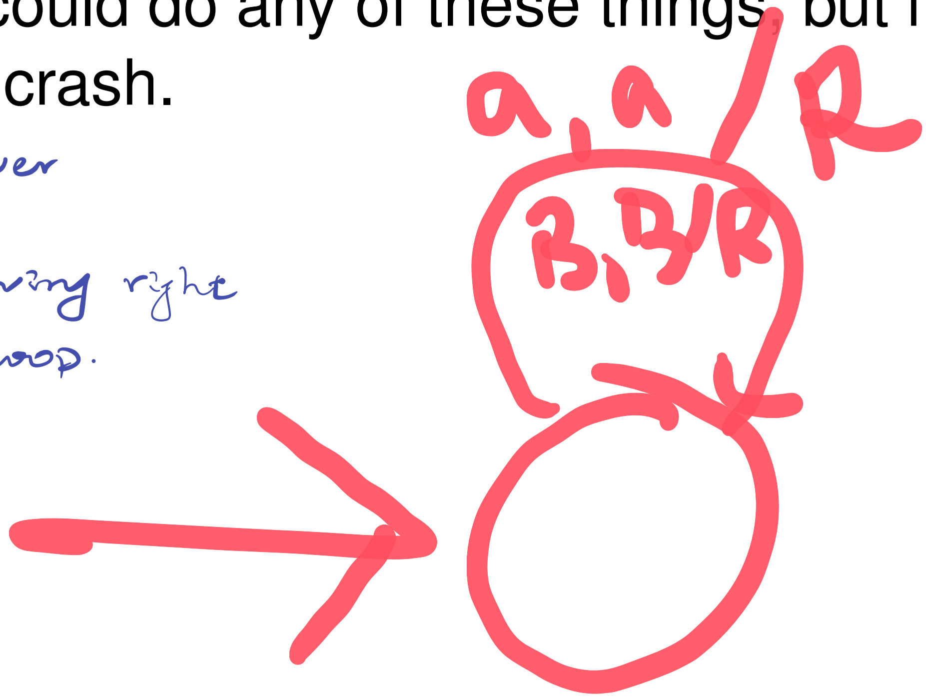
Outline

- Acceptance and Recognition by TMs.
- Recursive and Recursively Enumerable Languages.
- Closure Properties.

TMs that never halt

- With a DFA,NFA or PDA, one of three possibilities always occurred when reading an input word:
  - We arrive at a final state (empty stack) and accept.
  - We arrive at a non-final state (non-empty stack) and reject.
  - There is no further transition, and the devices “crashes”.
- A TM could do any of these things, but it could also never halt or crash.

this machine never crash or halt
it just keep moving right
like an infinite loop.



← for any letter.

Acceptance and Recognition

Recall that a language L is **accepted** by a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ if

final state start

$$L = L(M) = \{w \in \Sigma^* \;:\; \exists x_1, x_2 \in \Gamma^*, q_f \in F q_0 w \vdash_M^* x_1 q_f x_2\}.$$

We say that a language L is **recognized** by a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ if

(a) $L = L(M)$.

(b) For every word $w \notin L$, M eventually halts and rejects w.

Σ^* would either accept or halt/crash.

Recursive and Recursively Enumerable

- A language L is **recursive** if there is a TM M such that L is **recognized** by M. the one have to stop.
- A language L is **recursively enumerable** (r.e.) if there is a TM M such that L is **accepted** by M. a more generable one.

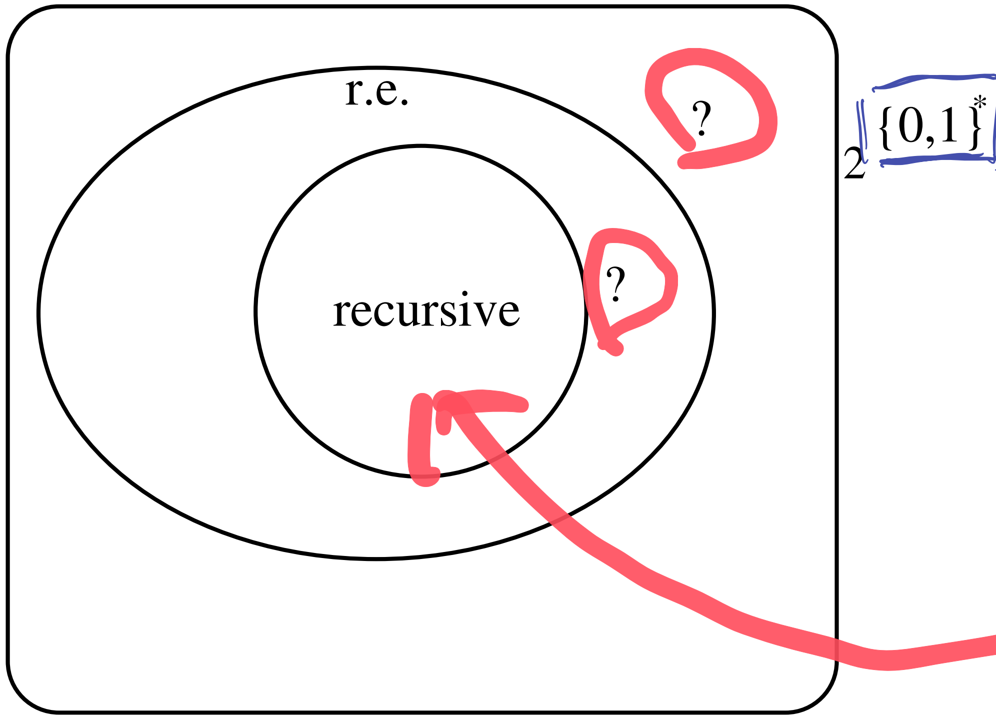
Every recursive language is a recursively enumerable language.

Examples of recursive languages:

- $L = \{a^n b^n c^n \ : \ n \geq 0\}$. not CFG, but it is recursive.
- $L = \{a^{n!} \ : \ n \geq 0\}$.

Examples of recursively-enumerable-but-not-recursive languages?

The Situation



$\{a^n b^n c^n : n \geq$

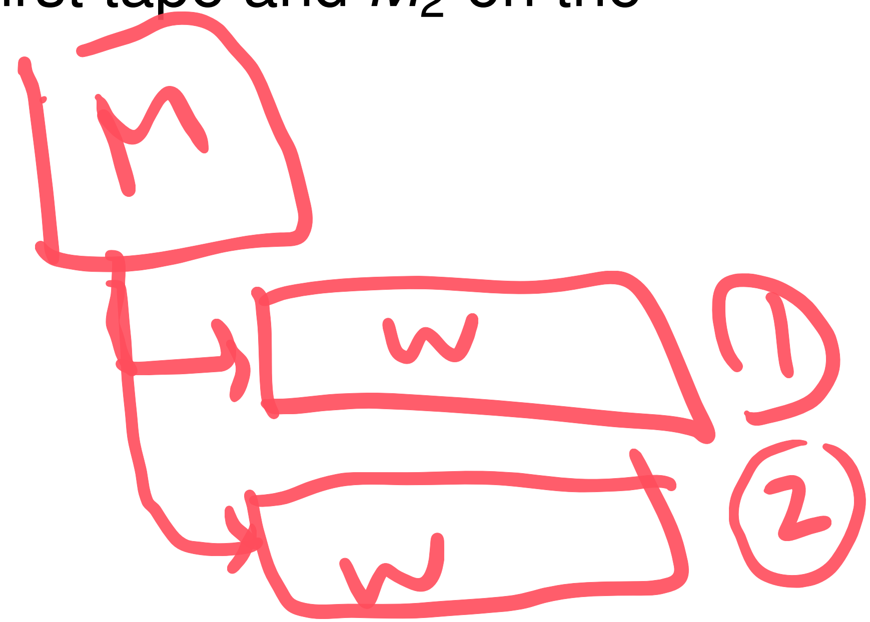
- Our goals: do there exist languages which are r.e. but not recursive?
- What about languages which are not r.e. ?

Recursive and r.e. Languages

Thm. The recursive (r.e.) languages are closed under union and intersection.

Proof. We show that the r.e. languages are closed under union. Let M_i be TMs for L_i, $i = 1, 2$. Then we design a 2-tape TM M such that M accepts $L_1 \cup L_2$:

- To begin, M copies the input from tape 1 to tape 2.
- M then simulates M_1 on the first tape and M_2 on the second tape **in parallel**.
- When does M accept?



Recursive and r.e. Languages

- M accepts according to the following rules:
  - (a) If M_1 or M_2 crashes, then M stops simulating that tape and continues.
  - (b) If the second machine crashes, then M stops and rejects.
  - (c) If M_1 or M_2 accepts, M stops and accepts.
  - (d) If M_1 and M_2 both do not halt, then M does not halt.
- In each of the cases (a)—(d), M does the right thing.

Closure under Complement

Thm. The recursive languages are closed under complement.

Proof. Let M be a TM. We add a new state q_f to M, and perform the following changes:

- Every accepting state in M is changed to a non-accepting state.
- If $(q, a) \in Q \times \Gamma$ is such that $\delta(q, a)$ is not defined, we set $\delta(q, a) = (q_f, a, S)$.
- q_f is the only final state of M.

M'

$w \in L(M)$

$q_0 w \vdash_M^* x_1 q x_2$

$q \in F$

$w \notin L(M)$

$q_0 w \vdash_M^* x_1 q x_2$

$q \notin F$

Closure under Complement

- The r.e. languages are not closed under complement (will show later).

Thm. If L is r.e. and $\overline{L}$ is r.e., then L is recursive.

- $L, \overline{L}$ are r.e.: let M_1, M_2 be TMs accepting L and $\overline{L}$.
- Let M be a 2-tape TM which simulates M_1 on tape 1 and M_2 on tape 2 **in parallel**.
- We claim that M can be made to recognize L. Why?

Closure Properties

The recursive and recursively enumerable languages are closed under:

- union, intersection.
- complement (recursive only).
- concatenation and Kleene closure.