

Turing Machines-- Machines entering into infinite loop

Supplement Slides

Recognizing or accepting a language by a machine (automaton)

- Let us define ID for a DFA/NFA as
 - (q, x) where q is the current state and x is the remaining input.
- We say L is recognized/accepted by a machine M (may be a DFA/NFA/PDA/TM)
$$L = \{w \in \Sigma^* \mid id_0 \vdash^* id_f \text{ where } id_0 \text{ and } id_f \text{ are initial and accepting ids, respectively}\}$$

DFA/NFA

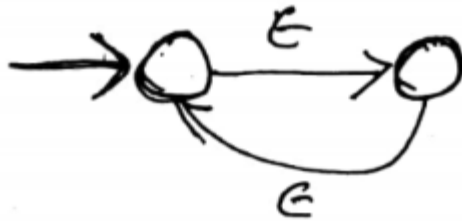
- $L = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (q_f, \epsilon) \text{ where } q_f \text{ is a final state}\}$
- Input has to be exhausted. { The remaining string should become ϵ }
- This is the same criterion even for NFAs.

DFA

- A DFA **never** enters in to an infinite loop.
 - Since there are no ϵ transitions, and
 - There is exactly one choice at every stage of the computation.
 - The input has to exhaust progressively (one character at each step) and should become ϵ
 - At this stage if the state is one of final, the input string is accepted, else rejected.

NFA

- A NFA can enter in to an infinite loop.



- For any given input this NFA enters in to an infinite loop.
- The language recognized by this machine is ϕ
- The language ϕ is regular, because there is a NFA to recognize this.

For this language we can find NFA and DFA that always halts.

- NFA

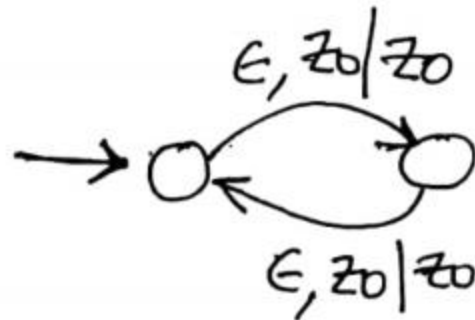


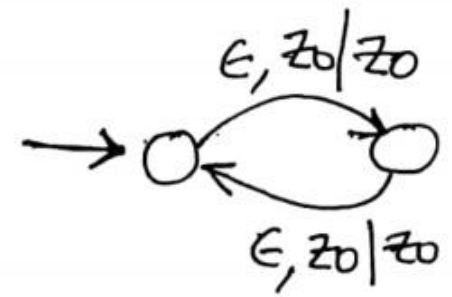
- There is a DFA (existence of either NFA or DFA is enough) also to accept ϕ . {DFA always halts}.



PDA by final state and empty stack

- For a PDA to recognize a language by final state, $L = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \epsilon, \alpha)\}$ where q_f is a final state and $\alpha \in \Gamma^*$
- A PDA can also enter in to an infinite loop





- The language accepted by this PDA is also ϕ .
- we can find a PDA which recognizes the language ϕ without entering in to an infinite loop.

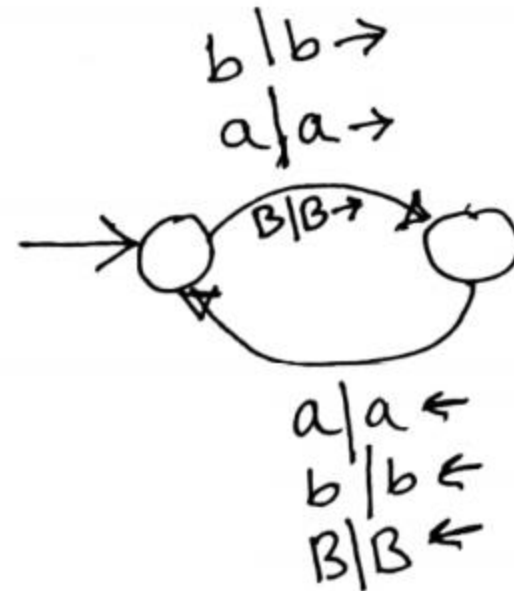


- Have you noted, both these PDAs are indeed DPDAs.

- For PDA by empty stack also similar arguments can be given.

TM

- For the given DTM also the language recognized is ϕ .



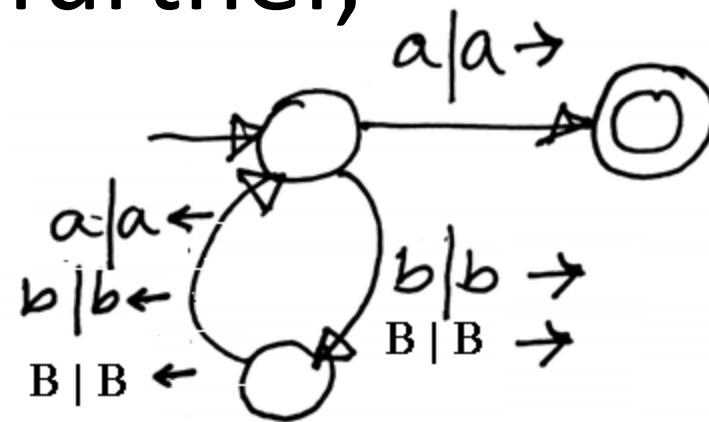
- Again there is a DTM that

Recognizes this language without entering in to infinite loop, which is

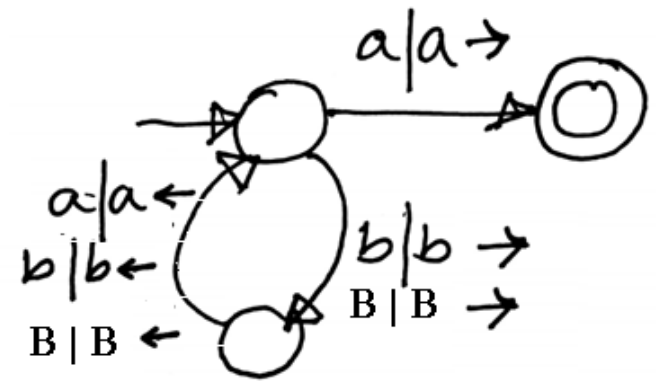


- So, $\phi \in RE$
- In fact $\phi \in R$ also.

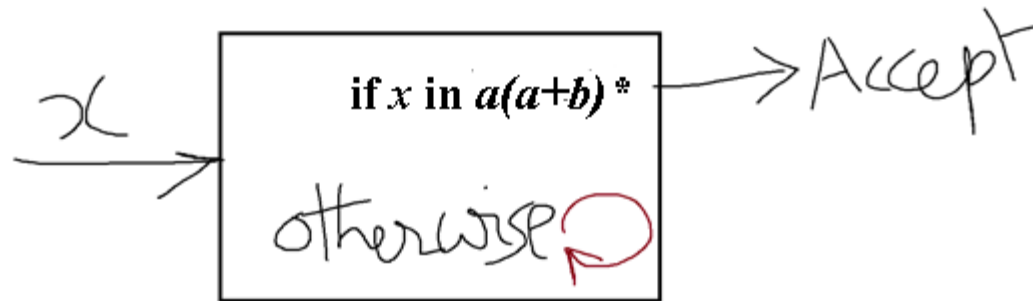
To elucidate further,



- The given DTM recognizes the language $a(a + b)^*$, but enters in to an infinite loop for all other inputs.
- So, the language recognized by this DTM is $a(a + b)^*$
- This is in RE.



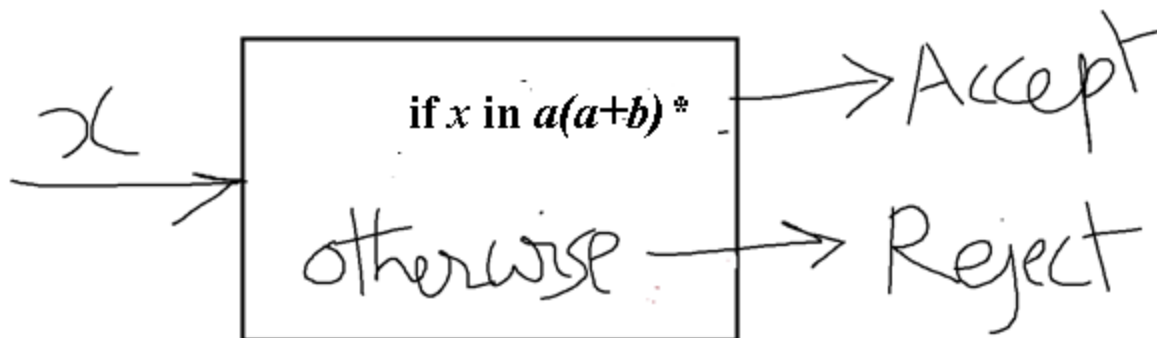
- Behaviour of this machine is



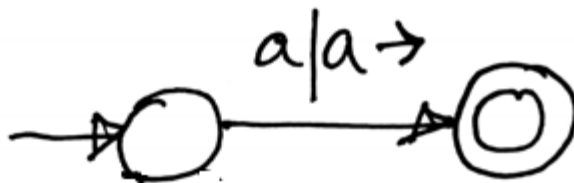
- so, the language $a(a + b)^* \in RE$

Can we say $a(a + b)^*$ is in R ?

- Yes.



- We need a TM which behaves like above



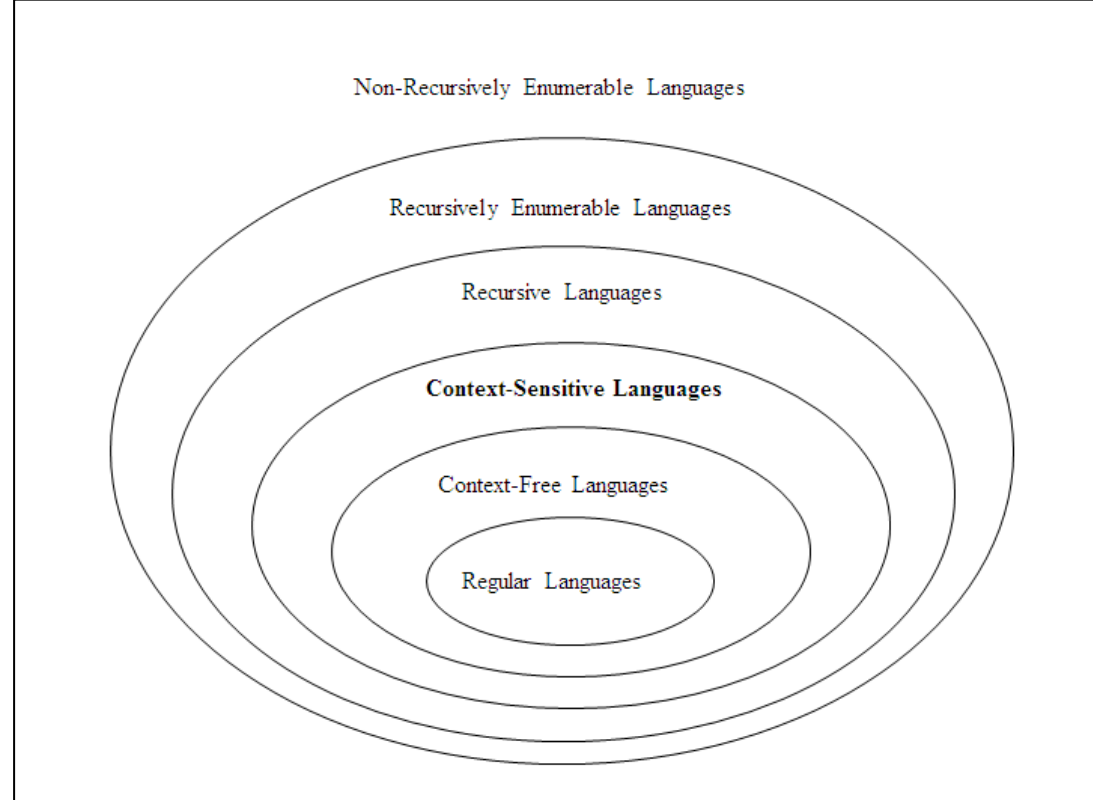
- Note, existence of one such TM is enough.

- Are there languages which are in RE but not in R ?
- Yes.
- Finding an $L \in RE - R$ is a **challenging** task.



Finding an $L \in RE - R$ is a **challenging** task

- That is, for any string in L , the machine should halt in a final state.
- There is a string which is not in L , for which every possible TM (which accepts L) enters in to an infinite loop.
- There exists such languages !!
- This one important aspect of the theory of computation.
- This, indeed points at limitations of computing machines.



- This diagram is saying that regular, context free and some other languages are recursive.
- That is, all these languages are Turing Decidable.

- We can say, for a regular language there exists a DFA (and also a NFA) that decides the language.
- Similarly for a CFL, a PDA that decides the language always exists.
- So, we do not need powerful TMs to decide regular and CFLs.