Church-Turing thesis

$\forall$ problem X $\exists$ algorithm for X

$\Updownarrow$ $\exists$ DTM for (deciding) X

Chapter 4 Decidability

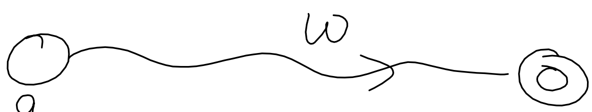
$$A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA and } w \in L(B) \}$$

or $\langle B \rangle, \langle w \rangle$

ex $w = a_3 a_7 a_1$

$\langle w \rangle = (a_3)(a_7)(a_1)$

Thus $\langle B, w \rangle \in A_{DFA} \iff B$ is a DFA and $w \in L(B)$



$q_1 \leftarrow$ in unisal state list

A_{DFA} is decidable

<u>Algorithm</u> (in words) DTM M_1

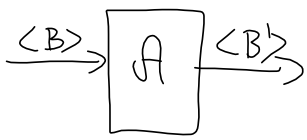
M_1 : on input $\langle B, w \rangle = \langle B \rangle \langle w \rangle$

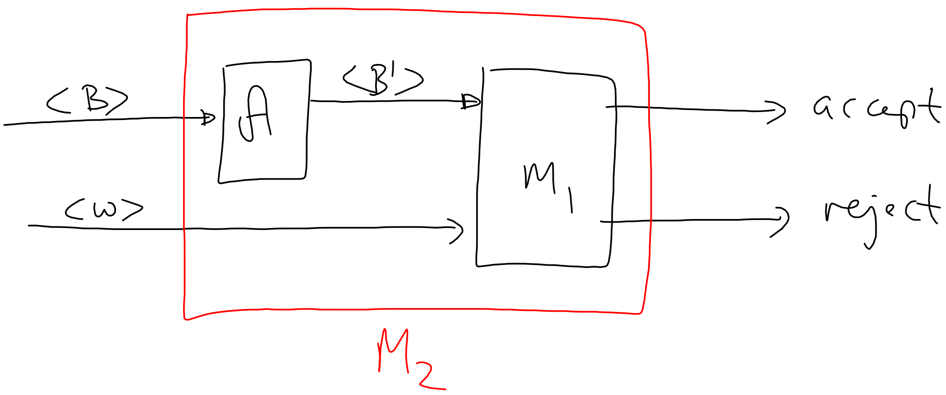
1. Check wheter B is a DFA
   if not, then reject $\langle B, w \rangle$.
2. Simulate B on w
3. If B ends in an accept state,
   then accept $\langle B, w \rangle$
   else reject $\langle B, w \rangle$

$A_{NFA} = \{\langle B, w\rangle \mid B \text{ is an NFA and } w \in L(B)\}$

Thm A_{NFA} is decidable

Let $\mathcal{A}$ be an algorithm for converting an NFA to an equivalent DFA B'





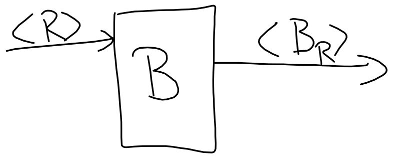
M_2 decides A_{NFA}

(If $\langle B\rangle$ is not coding an NFA, we assume that $\mathcal{A}$ returns $\langle B'\rangle$ which is not a DFA)

$$A_{REX} = \{ \langle R \rangle \langle w \rangle \mid R \text{ is a regular expression} \wedge w \in L(R) \}$$

<u>Thm</u> A_{REX} is decidable

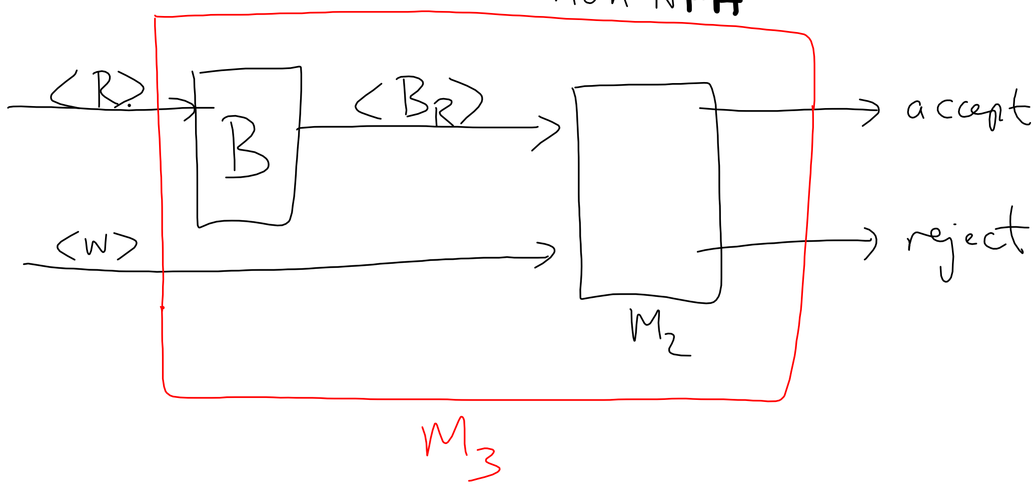
<u>P</u>:



B checks whether R is a legal reg. expr.

If 'Yes' then **B** generates B_R an NFA with $L(B_R) = L(R)$

If no then B returns a coding $\langle B_R \rangle$ of a fixed non-N**FA**

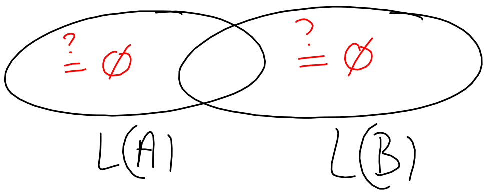


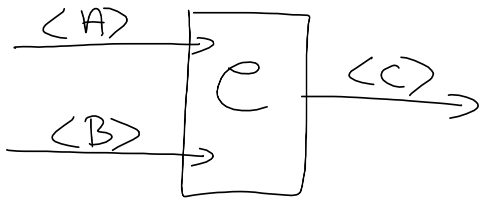
$E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$

easily seen to be decidable by DTM M_4

$EQ_{DFA} = \{ \langle A \rangle \langle B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$

Thm EQ_{DFA} is decidable.



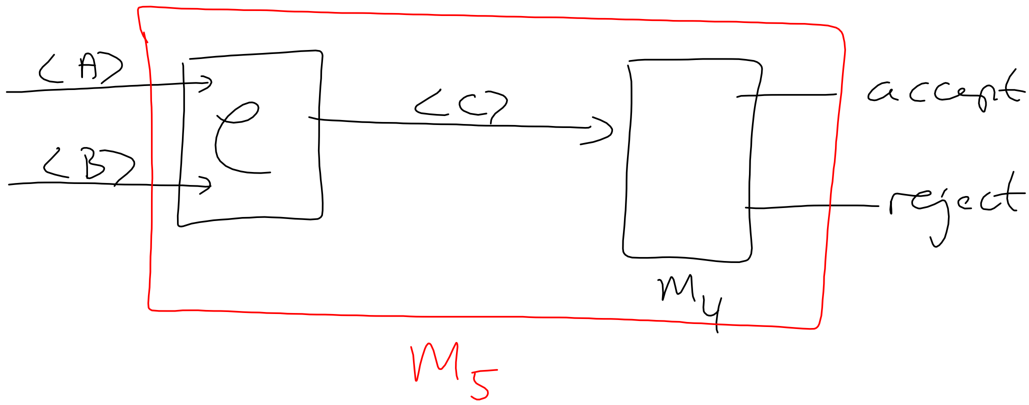


if A and B are DFAs then

$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(L(B) \cap \overline{L(A)}\right)$

Else $\langle C \rangle$ DFA with $L(C) = \Sigma^*$

$L(C) = \emptyset \iff \langle A \rangle \langle B \rangle \in EQ_{DFA}$


<A>
<B>
C
<C>
accept
reject
M4
M5

$A_{CFG} = \{\langle G\rangle\langle w\rangle \mid G \text{ is a CFG} \wedge w \in L(G)\}$

Thm 4.7 A_{CFG} is decidable (by DTM M_{CFG})

"easy" to check whether $\langle G\rangle$ codes a CFG.

convert G to G' equivalent CFG in chomsky normal form.

Check all possible derivations of length $2|w|-1$
if one of these is w, answer 'Yes'
else answer 'no'

$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG} \wedge L(G) = \emptyset \}$

<u>Thm E_{CFG} is decidable</u>

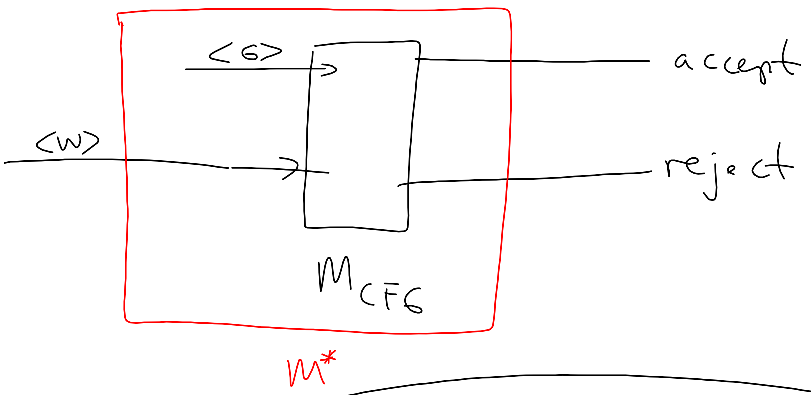
P: we want to check whether $\exists \beta \in \Sigma^*$ (string of terminals)
s.t $S \overset{*}{\Rightarrow} \beta$

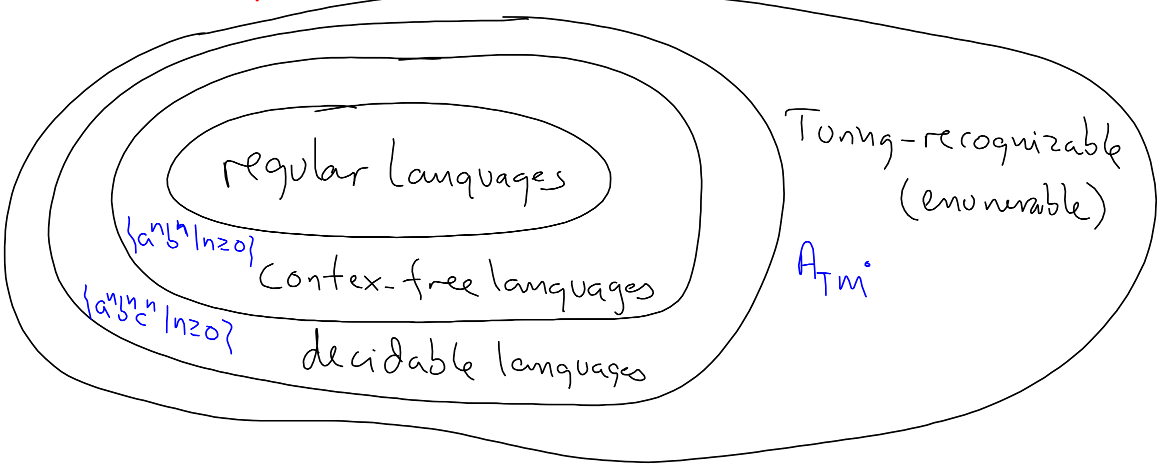
idea: terminal symbols are marked
if $A \to u_1 u_2 \cdots u_n$ when all u_i marked
then mark A
repeat this until no change
if S is ever marked 'reject'
else 'accept'.

$S \to AB \mid CD$
$A \to AA$
$B \to BC$
$C \to c$
$D \to AB \mid d$

Thm 4.9 every CFL is decidable

P: We need an algorithm which given a CFL A (represented by a CFG G with $L(G)=A$) and $w \in \Sigma^*$, decides whether $w \in A$



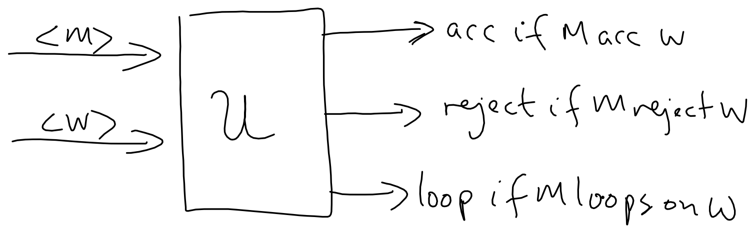


$A_{TM} = \{ \langle M \rangle \langle w \rangle \mid M$ is a (Deterministic) TM and $w \in L(M)$

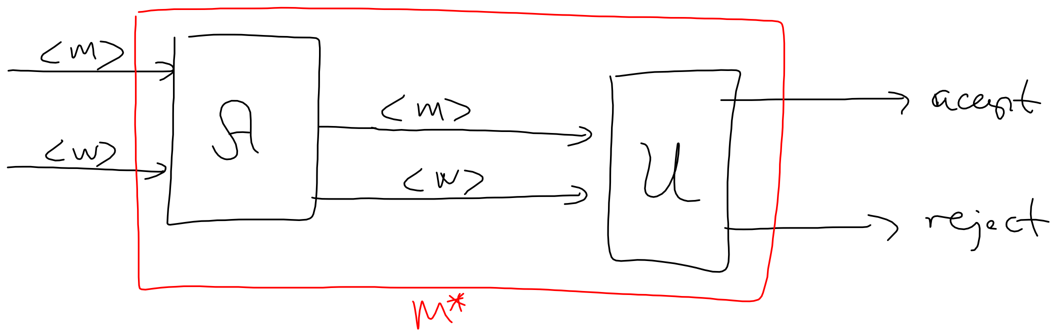
Thm A_{TM} is Turing-recognizable

part 1: check whether $\langle M \rangle$ codes a DTM ✓

if $\langle M \rangle$ ok :



alternatively



A : check whether $\langle M \rangle$ codes a DTM
if not 'loop'
else send $\langle M \rangle, \langle w \rangle$

M^* recognizes A_{TM}

Countable sets

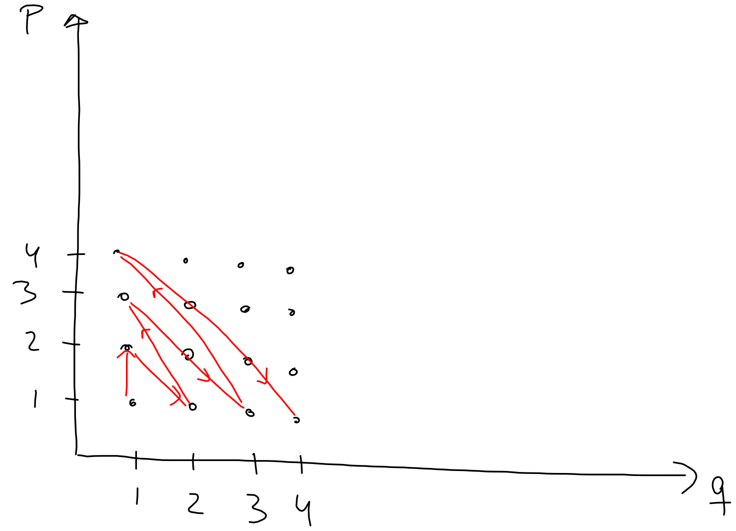
Def A set S is countable if either

1. S is finite, or
2. $\exists f: S \to \mathbb{N}$, s.t f is 1-1 and onto

Ex $S = \{2k \mid k \in \mathbb{N}\}$ $f: 2k \to k$

Ex 2 $S = \Sigma^*$ f orders strings in S lexicographically

Ex 3 rational numbers $(S = \mathbb{Q})$



Thm $\mathbb{R}$ is uncountable (Read yourselves)

Thm The set B of infinite binary strings is uncountable

P: suppose b_1, b_2, b_3 .... is a list of all infinite binary strings

define b^* s.t on position i in b^*

$$b^*(i) = \begin{cases} 1 & \text{if } b_i(i) = 0 \\ 0 & \text{if } b_i(i) = 1 \end{cases}$$

Suppose $b^* = b_j$

problem $b^*(j) = 1 - b_j(j) \neq b_j(j)$ ↯

<u>Observation</u> Every language L over an alphabet Σ is a subset of $P(\Sigma^*)$

and L can be decribed via infinite binary strings:

order Σ^* lexicographically

if jth string on this order is in L then $b_L(j)=1$

else $b_L(j)=0$

<u>Corollary</u> The set of all languages over a nontrivial alphabet is uncountable.