Simulation of a non-deterministic TM:

- starts with input on tape 1, others empty.

Repeat: (i) write down (sequentially) the next branch of computation on tape 2.

(ii) If the branch written on tape 2 is a valid one, copy input (w) from tape 1 to tape 3, and carry out the computation. If it goes to q_{accept}, stop (accept).

Theorem: The set of turing decidable (or recursive) languages is closed under

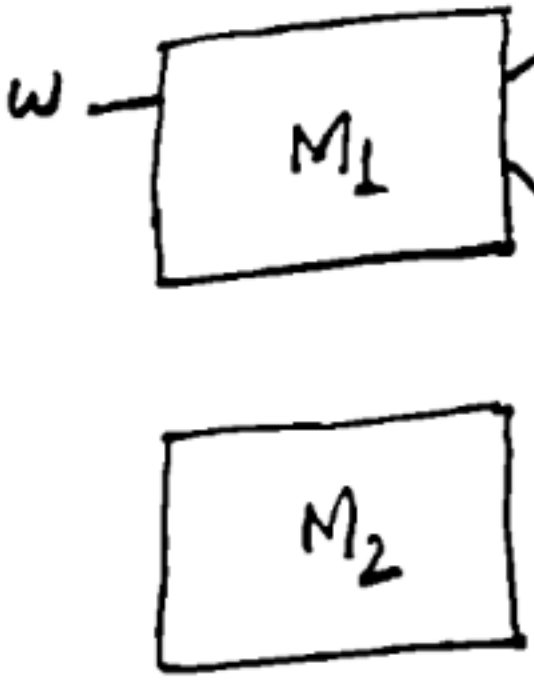
a) Union

b) Concatenation

c) Star operation

d) complementation

e) Intersection



. M_1 decides A.

. M_2 decides B.

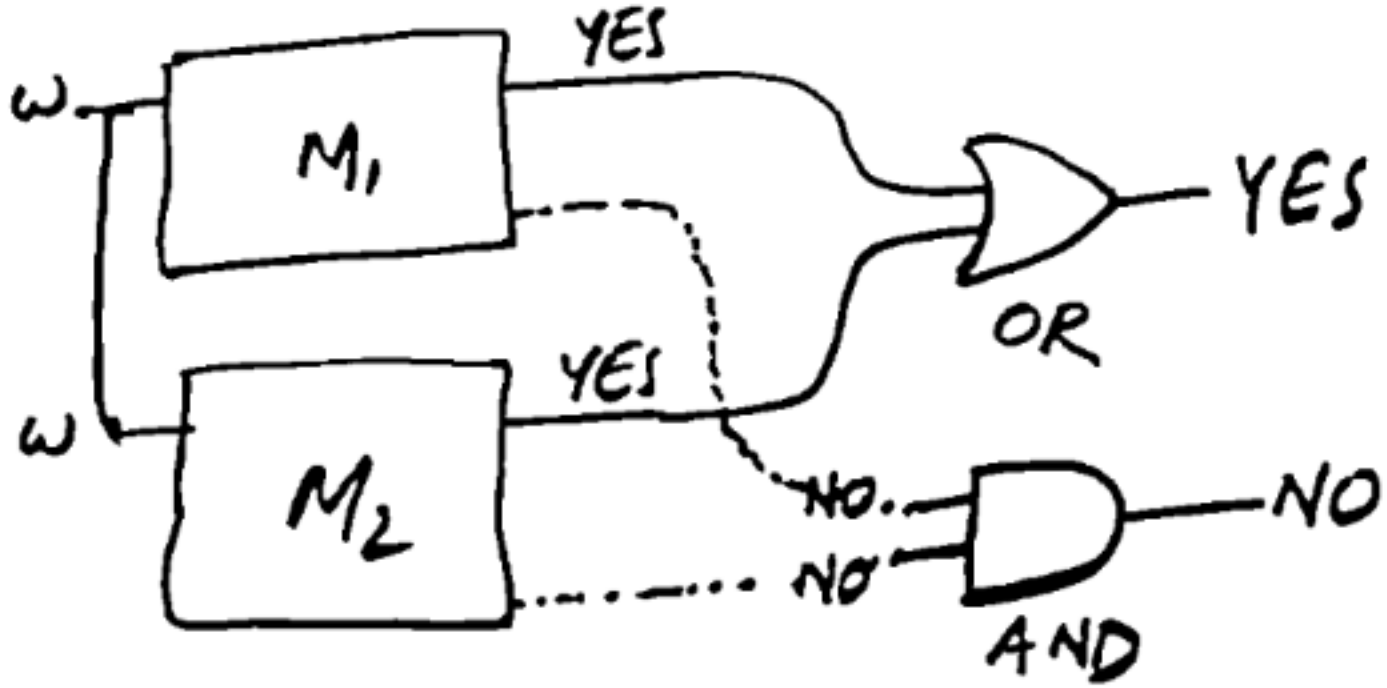
i.e., On an input w,

M_1 says yes if $w \in A$

says NO if $w \notin A$.

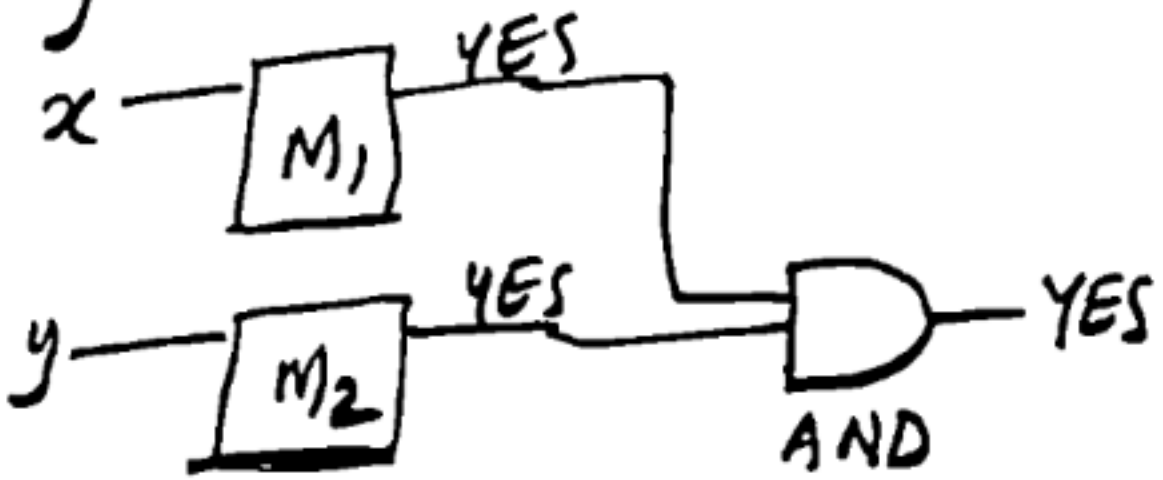
Similarly for M_2 and B.

Given M_1 and M_2, can you design M for $A \cup B$?

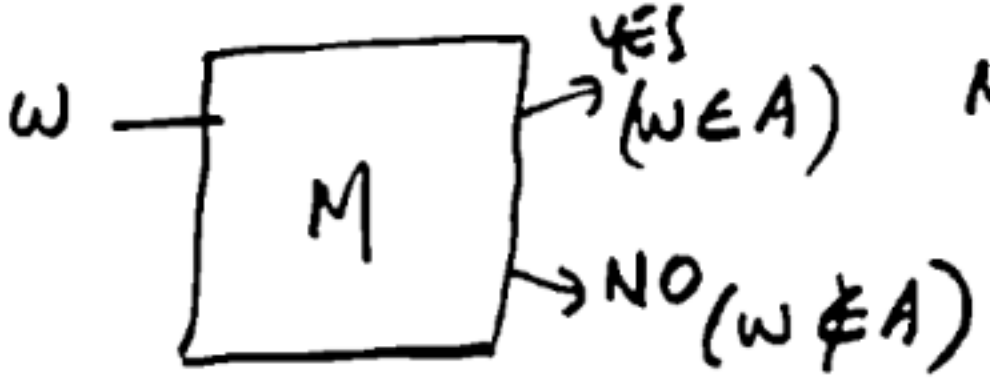


M

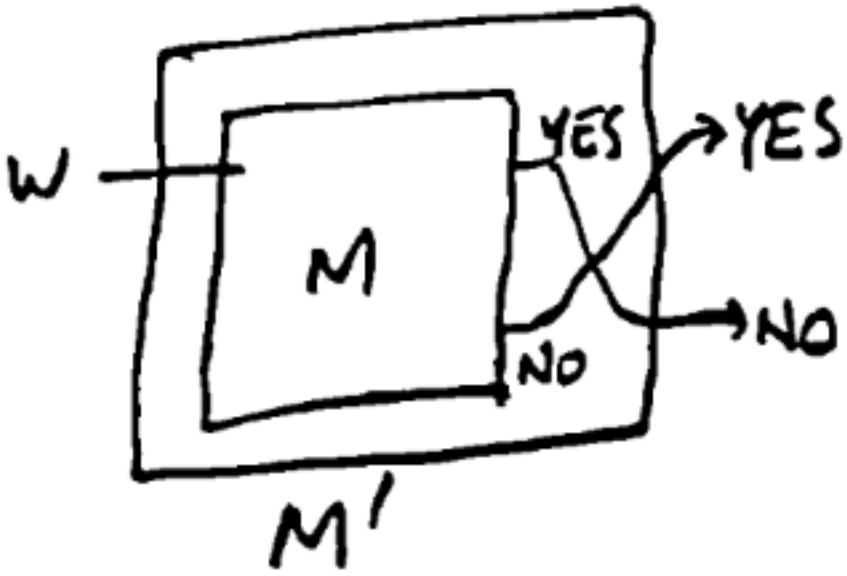
- Intersection - similar.
- Concatenation : Given an input string w, Non-deterministically "guess" a splitting point, to get $w = xy$.



• Complementation.



M decides A.



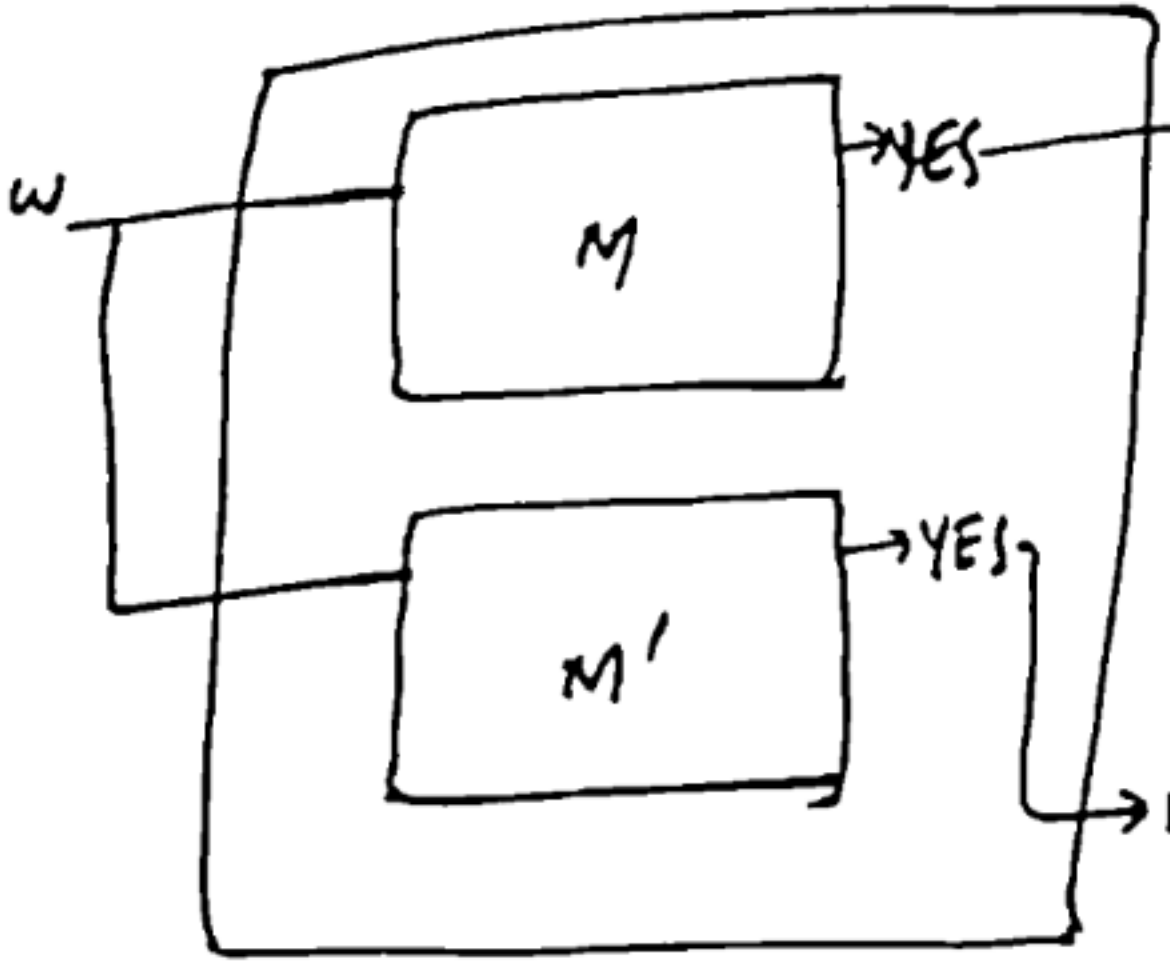
M' decides $\bar{A}$.

- Except for complementation, everything else works for turing recognizable languages also. They are closed under
  a) Union
  b) Concatenation
  c) Star
  d) Intersection.

. Theorem (or an important fact):

If language A and its complement $\bar{A}$
are both turing recognizable,
then A is turing decidable.

(Proof Easy).



D

. D decides A.

. M recognizes A,
M' recognizes $\bar{A}$.

ie, M says YES
if and only if
$w \in A$.

. M' says YES
if and only if $w \in \bar{A}$.
(i.e, $w \notin A$).

- If A is decidable, so is $\bar{A}$.
- If A & $\bar{A}$ are recognizable, both are decidable also.
- If A is recognizable and NOT decidable, what can we say about $\bar{A}$?

• $\bar{A}$ is not turing recognizable.
(otherwise A would be decidable).

• Similarly, if A is not turing recognizable, what can we say about $\bar{A}$?

- $\bar{A}$ is also not turing recognizable

OR

- $\bar{A}$ is recognizable, but not decidable.

- Next we are trying to define a language that is NOT turing recognizable.
- For this, we 'number' (order) the input-strings as well as all turing machines.
- Assume the input alphabet is $\{0,1\}$.
  What is a good ordering of input strings?

$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\ldots$

this is a good ordering in the sense that given a string, we can easily find the 'position' of that string (or index) in this order.

index(w) = no. of strings with length less than the length of w + decimal value of $w + 1$ = 2^{length} + value of w.