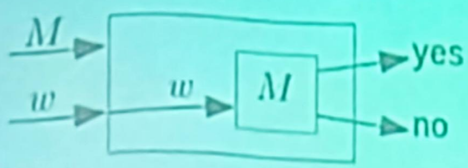
Universal Turing Machine

There exists a TM which takes $\langle M, w\rangle$ as input and simulates TM M on input w (step by step)

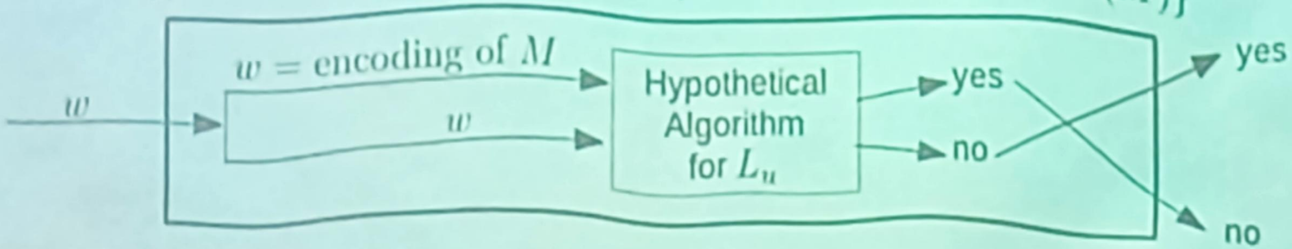


$L_u = \{x \mid x \text{ encodes } (M, w) \text{ and } w \in L(M)\}$ is r.e. but not recursive

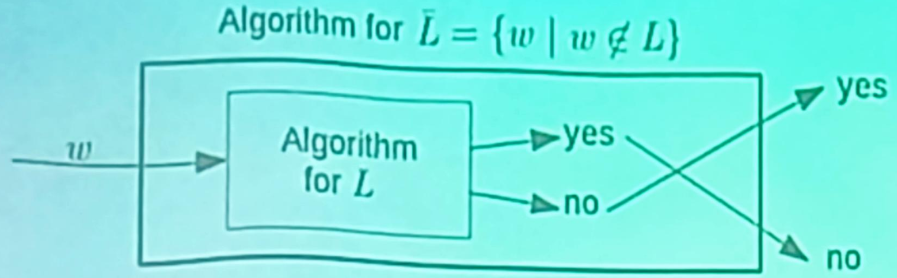
L_u is the language accepted by universal TM

L_u is recursive implies L_d is r.e. (contradiction)

Hypothetical Algorithm for $L_d = \{w \mid w \text{ encodes TM } M, w \notin L(M)\}$

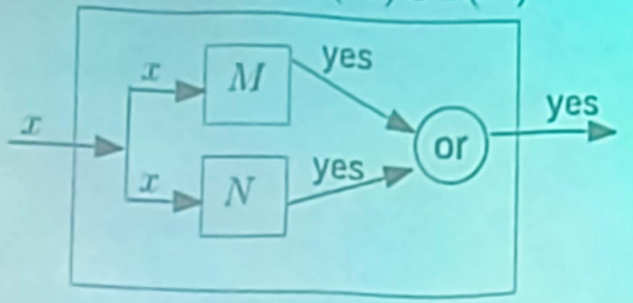


Theorem *If L is recursive then $\bar{L}$ is recursive*



Parallel Simulation of two TMs

TM to accept $L(M) \cup L(N)$



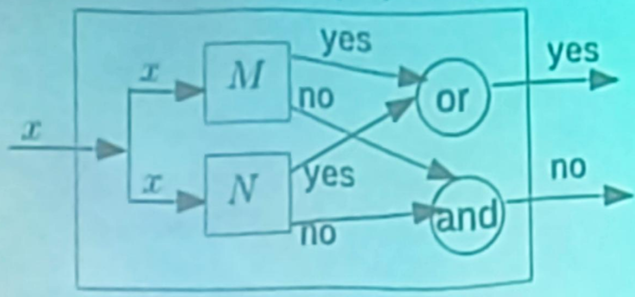
Theorem *If L_1, L_2 are r.e. then $L_1 \cup L_2$ is r.e.*

M, N are TMs which accept L_1, L_2 respectively

```
ID_M = Initial ID of M on input x
ID_N = Initial ID of N on input x
while true
{   If ID_M reaches a final state of M then return yes
    If ID_N reaches a final state of N then return yes

    If M does not halt in ID_M then
    {  Compute ID_M ⊢ ID_M_next ; ID_M := ID_M_next}
    If N does not halt in ID_N then
    {  Compute ID_N ⊢ ID_N_next ; ID_N := ID_N_next}
}
```

Parallel Simulation of two TMs

Algorithm for $L(M) \cup L(N)$



Theorem *If L_1, L_2 are recursive then $L_1 \cup L_2$ is recursive*

M, N are Algorithms which decide L_1, L_2 respectively

```
ID_M = Initial ID of M on input x
ID_N = Initial ID of N on input x
while true
{  If ID_M reaches a final state of M then return yes
   If ID_N reaches a final state of N then return yes

   If M does not halt in ID_M then
   {  Compute ID_M ⊢ ID_M_next ; ID_M := ID_M_next}
   If N does not halt in ID_N then
   {  Compute ID_N ⊢ ID_N_next ; ID_N := ID_N_next}

   If both M and N halt in nonfinal state then return no
}
```

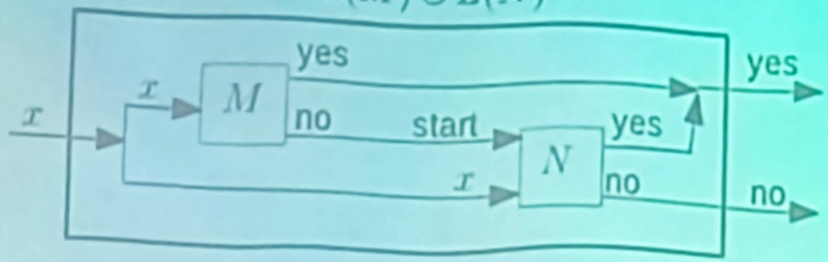
Sequential Simulation of two TMs

(Alternate Proof)

Theorem *If L_1, L_2 are recursive then $L_1 \cup L_2$ is recursive*

M, N are Algorithms which decide L_1, L_2 respectively

Algorithm for $L(M) \cup L(N)$

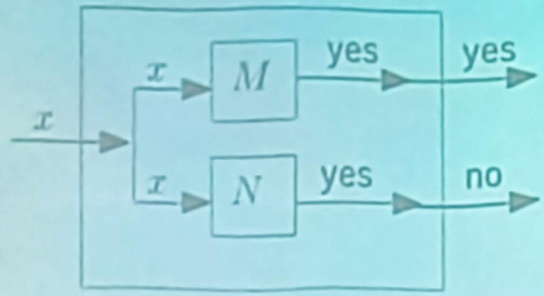



```
ID_M = Initial ID of M on input x
ID_N = Initial ID of N on input x

while ID_M is not a halting configuration
(  Compute ID_M ⊢ ID_M_next ; ID_M := ID_M_next)
If ID_M reaches a final state of M then return yes

while ID_N is not a halting configuration
(  Compute ID_N ⊢ ID_N_next ; ID_N := ID_N_next)
If ID_N reaches a final state of N then return yes

return no
```

Parallel Simulation of two TMs

Theorem *If $L, \bar{L}$ are both r.e. then $L, \bar{L}$ are both recursive.*

Algorithm for L



$L = L(M)$, $\bar{L} = L(N)$
M, N may not always halt

L is recursive. There exists an algo for L.
If L is recursive then $\bar{L}$ is recursive.

```
ID_M = Initial ID of M on input x
ID_N = Initial ID of N on input x
while true
{  If ID_M reaches a final state of M then return yes
   If ID_N reaches a final state of N then return no

   If M does not halt in ID_M then
   {  Compute ID_M ⊢ ID_M_next ; ID_M := ID_M_next}
   If N does not halt in ID_N then
   {  Compute ID_N ⊢ ID_N_next ; ID_N := ID_N_next}
}
```