- Some details that we skipped earlier:
  - How to encode a move, and
  - How to give a number for a TM.

Input alphabet: $\{0,1\}$

Fact: Any TM that runs on this input has an equivalent TM that uses only 3 symbols: $\{0, 1, \sqcup\}$

- Now, the tape symbols can be numbered:

$$\begin{array}{ccc} 0 & 1 & \sqcup \\ (X_1) & (X_2) & (X_3) \end{array}$$

- States can be numbered too. 'Special' states q_{accept} and q_{reject} at 1 and 2.

$$q_1 \quad q_2 \quad q_3 \quad \cdots\cdots \quad q_n$$

- Directions: Left (D_1), Right (D_2)

A move $\delta(q_i, X_j) = (q_k, X_l, D_m)$

can be encoded as

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Now, to encode the machine:

$1\ \text{move}_1\ 11\ \text{move}_2\ 11 \ldots 11\ \text{move}_{last}\ 1$

- Some more decidable languages

* Given a DFA A, Is the language $L(A)$ empty?

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

<u>Thm:</u> E_{DFA} is decidable.

Algorithm:

1. - Mark the start state of A
2. Repeat until no new states get marked:
   - Mark all states to which there is an arrow (move) from an already marked state.
3. If no final (accept) state marked, answer YES. else NO.

- Given a regular expression R and a string w, Does R generate w?

Language:

$$L_{REX} = \{ \langle R, w \rangle \mid \text{Reg exp } R \text{ generates } w \}$$

- What would be the algorithm?

EQ_{DFA} : $\{ \langle A, B \rangle \mid$ DFAs A and B are equivalent. Or, $L(A) = L(B) \}$

EQ_{DFA} is also decidable.

Proof Idea: Construct C, such that C accepts strings accepted by A or B, not both.

Rest of the proof? Use E_{DFA}.

- Run the input string w on A and B
- A accepts and B rejects $\rightarrow$ accept
  B accepts and A rejects $\rightarrow$ accept
- Else do not accept.

This is our C.

- Now, just check if $L(C)$ is EMPTY.

$L(C) = \phi$ means $L(A) = L(B)$.

L_{HALT} : $\{ \langle M, w \rangle$: Machine M halts on input string $w \}$

'halts' means accepts or rejects.

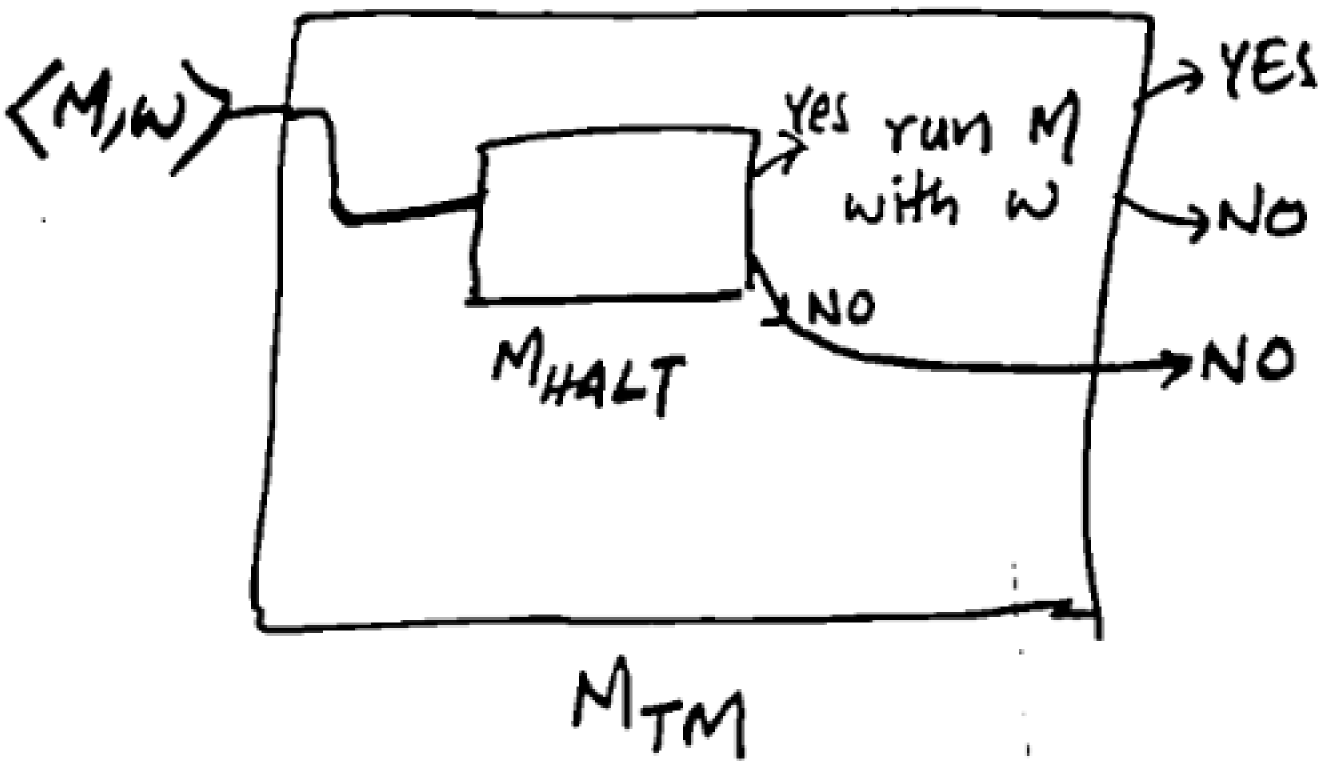
Qn: Given M and input string w,
Does M halt on w?

(HALTING PROBLEM)

Thm: HALTING PROBLEM is undecidable.

Or, there is no TM M_{HALT} that can answer this YES/NO question correctly.

- Proof Idea: If there was such an M_{HALT}, then L_{TM} is decidable.



- Step 1: Give ⟨M,W⟩ as input to M_{HALT}. If it says NO, answer NO.

(It is correct, 'NO' from M_{HALT} means M is going to loop forever on w)

- Step 2: On YES, we are guaranteed that M will halt on w.
  So run M with w as input.
  YES → YES, NO → NO.