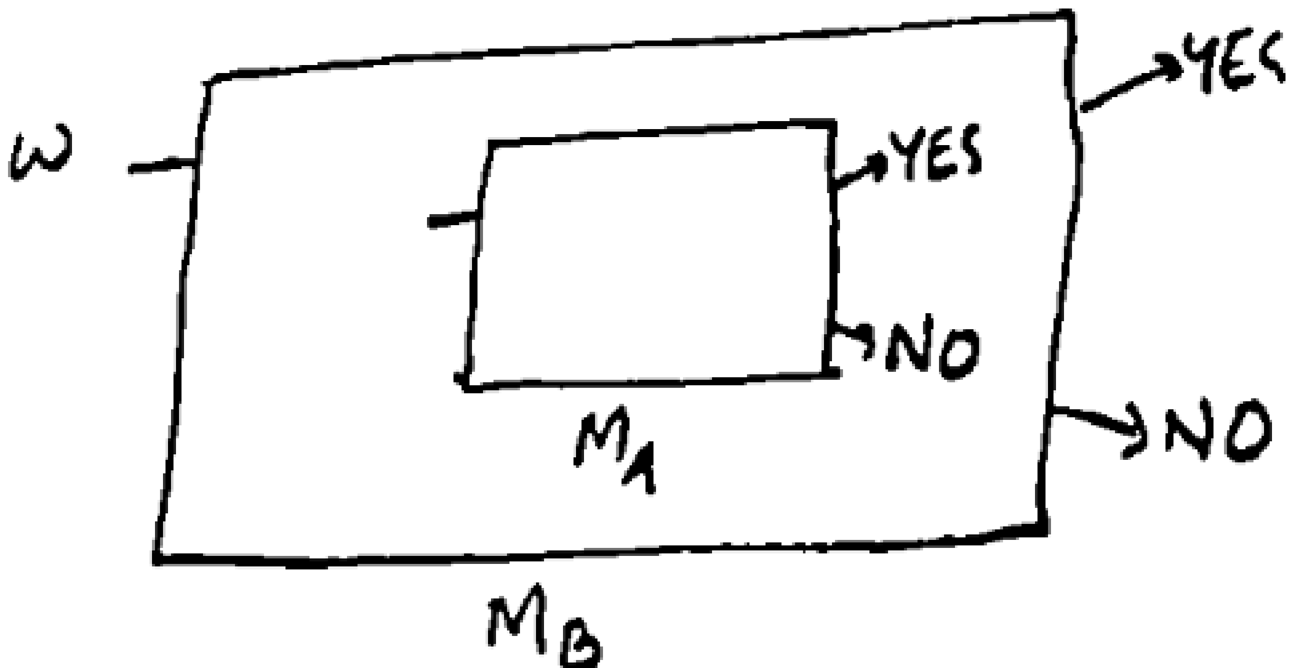
- **Reducibility:**
  - We have been 'reducing' one problem to the other already.
  - If problem A can be solved, B can also be solved (using a computer).
  - If A is decidable, B is also decidable.

- These are called 'reductions'.
- We reduce one problem to the other.

For example, "if A is decidable, then so is B".



- We use a TM that decides A
  to design a TM that decides B.
- i.e, if A was solvable, then we can solve B.

Or, we reduce the problem of solving B
to solving problem A.

We say "B reduces to A".

- Useful in proving that some problems are undecidable (as we already did) and it is also useful in the next module, in showing results about the complexity (or hardness) of solving a problem.

It can be used in different ways:

① If A can be solved, then B can also be solved. (simple)

② If A can be solved easily, then B can also be solved easily.

(we use this in complexity).

• Reductions that we already did:

(i) We showed: If L_{TM} was decidable,
then L_d is decidable.

i.e., we reduced L_d to L_{TM};
to show that L_{TM} is undecidable.

(Because we know L_d is not decidable).

(ii) To show that Halting problem is undecidable,

we reduced L_{TM} to L_{HALT}.

ie, we gave a design for M_{TM} (that decides L_{TM}), using a machine M_{HALT} that decides L_{HALT}.

- More undecidability proofs, using this 'reduction' technique:

- Defⁿ: $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM},\ L(M) = \phi\}$

<u>Theorem:</u> E_{TM} is undecidable.

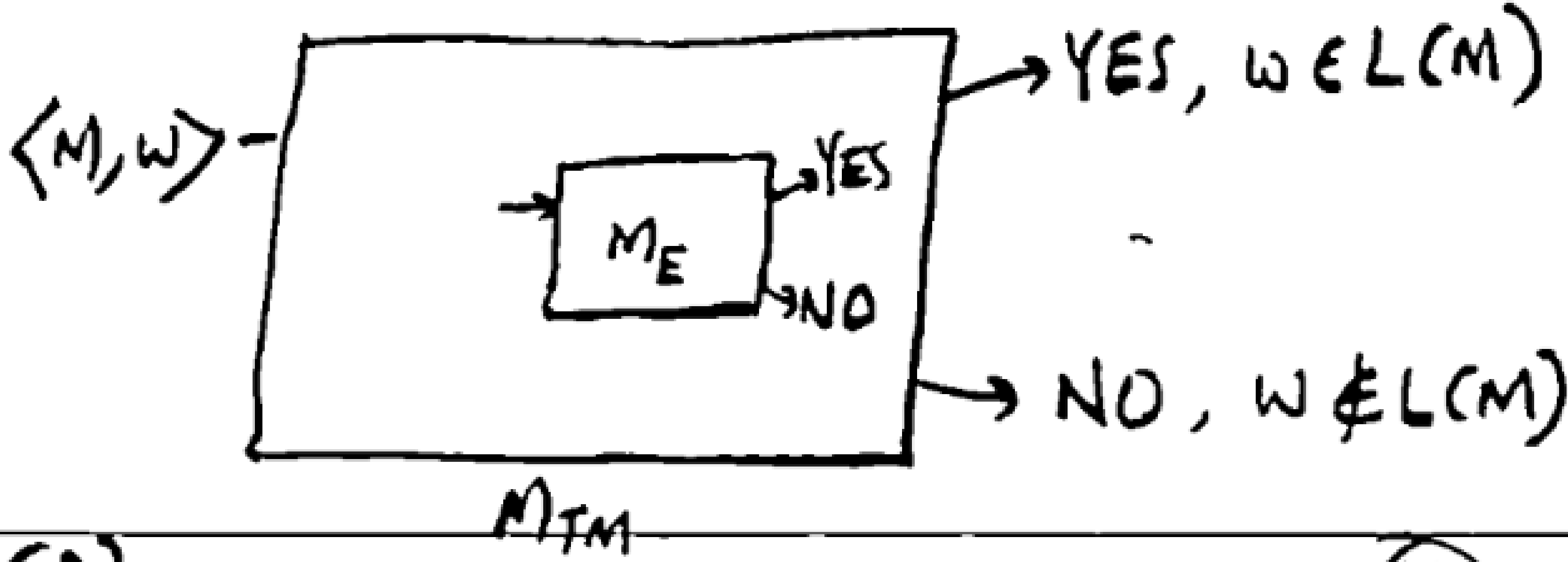
Proof idea:

If E_{TM} is decidable, then some problem B (that is known to be undecidable) is decidable.

i.e., We have to reduce B to E_{TM}.

What is a good choice for B?

- We choose L_{TM} as our B.
- Now, we have to design an M_{TM} (that decides L_{TM}), with the help of a TM that decides E_{TM}.



- Questions: - What should be the input to M_E?
  - What is the algorithm that this M_{TM} uses?

<u>Idea:</u> we can use the given M and w to design a machine M_w:

It does this: On input x,
- if $x \neq w$, reject.
- Else, run M on w, accept is M accepts.

(SUDEEP)

Now, the algorithm for M_{TM} is as follows:

Input: $\langle M, w \rangle$

- Construct an M_w using M & w;
- Give M_w as input to M_E
- If M_E says NO, accept.
  if M_E says YES, reject.