CS 154

Advanced Computability: Oracles, Self-Reference, Foundations

Next Wednesday (2/17)

Your Midterm: IN CLASS

Today: instead of a new homework,

you'll get an optional (not graded!) practice midterm

Solutions to practice midterm will come out during the weekend. Same with all remaining HW solutions.

When you see the practice midterm…

DON'T PANIC!

Practice midterm will be harder than midterm

Next Wednesday (2/17)

Your Midterm: IN CLASS

Today: instead of a new homework,

you'll get an optional (not graded!) practice midterm

FAQ: What is fair game for the midterm?

Everything BEFORE this lecture (Lectures 1-10)

FAQ: Can I bring notes?

Yes, one single-sided sheet of notes, letter paper

Rice's Theorem (Restated)

Suppose L is a language that satisfies two conditions:

1. **(Nontrivial) There are TMs M_{YES} and M_{NO}, where $M_{YES} \in L$ and $M_{NO} \notin L$**
2. **(Semantic) For all TMs M_1 and M_2 such that $L(M_1) = L(M_2)$, $M_1 \in L$ if and only if $M_2 \in L$**

Then, L is undecidable.

Recognizability via Logic

Def. A decidable predicate R(x,y) is a proposition about the input strings x and y, such that some TM M implements R. That is,

for all x, y, R(x,y) is TRUE $\Rightarrow$ M(x,y) accepts
R(x,y) is FALSE $\Rightarrow$ M(x,y) rejects

Can think of R as a function from $\Sigma^* \times \Sigma^* \to \{T,F\}$

EXAMPLES: R(x,y) = "xy has at most 100 zeroes"
R(N,y) = "TM N halts on y in at most 99 steps"

Theorem: A language $A \subseteq \Sigma^*$ is *recognizable* if and only if there is a decidable predicate $R(x, y)$ such that:

$$A = \{ x \mid \exists y \in \Sigma^* \ \ R(x, y) \}$$

Proof: (1) If $A = \{ x \mid \exists y\, R(x,y) \}$ then A is recognizable

Define the TM M(x): For all strings $y \in \Sigma^*$,
If R(x,y) is true, *accept*.
Then, M accepts exactly those x s.t. $\exists y\, R(x,y)$ is true

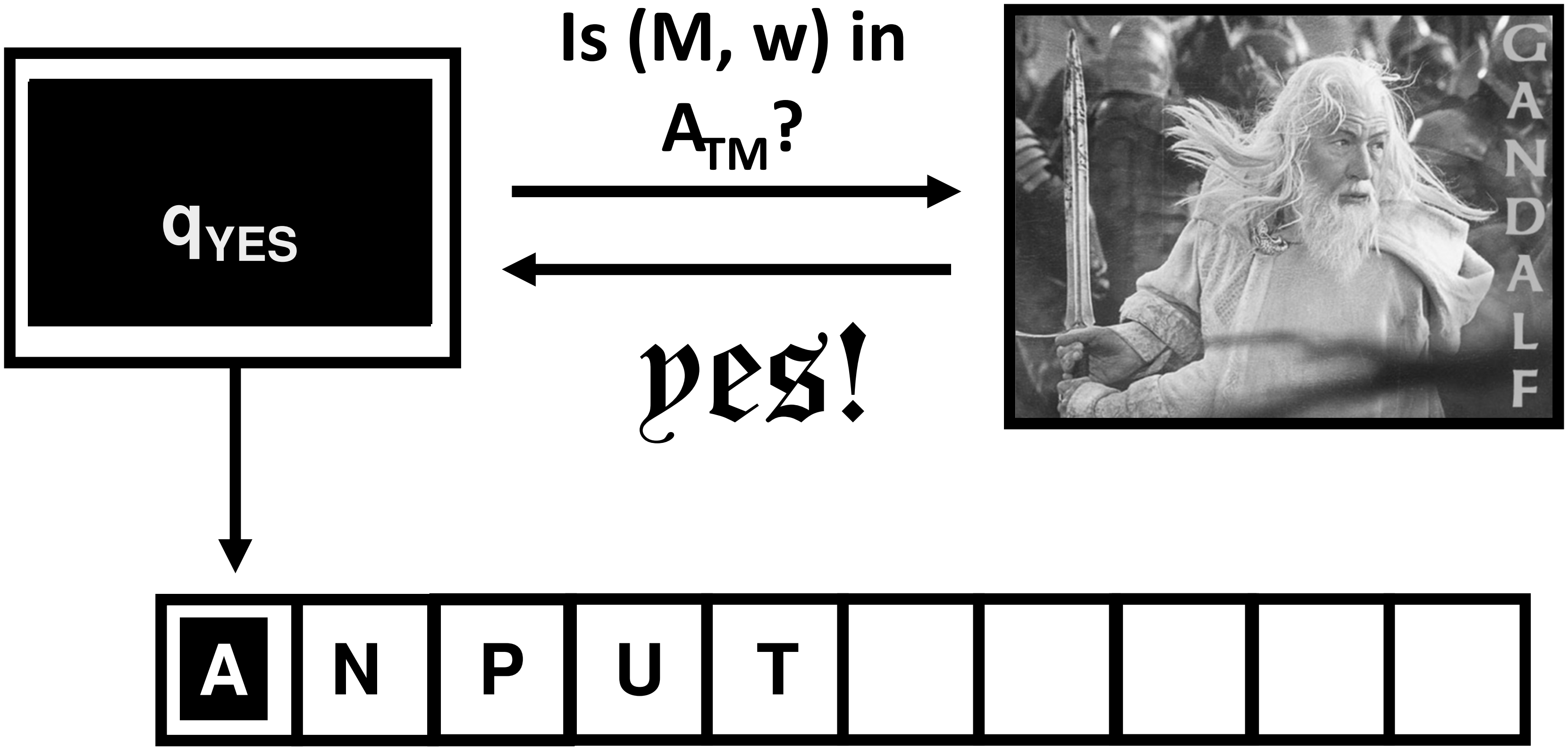
(2) If A is recognizable, then $A = \{ x \mid \exists y\, R(x,y) \}$

Suppose TM M recognizes A.
Let R(x,y) be TRUE iff M accepts x in $|y|$ steps
Then, M accepts x $\Leftrightarrow \exists y\, R(x,y)$

Computability With Oracles

***We do not condone smoking. Don't do it. It's bad. Kthxbye**

Oracle Turing Machines



INFINITE TAPE

Now leaving reality for a moment….

Oracle Turing Machines

An oracle Turing machine M is equipped with a set $B \subseteq \Gamma^*$ to which a TM M may ask membership queries on a special "oracle tape"
[Formally, M enters a special state $q_?$]

and the TM receives a query answer in one step
[Formally, the transition function on $q_?$ is defined in terms of the *entire oracle tape*:
if the string y written on the oracle tape is in B,
then state $q_?$ is changed to q_{YES}, otherwise q_{NO}]

This notion makes sense even if B is not decidable!

How to Think about Oracles?

Think in terms of Turing Machine pseudocode!

An oracle Turing machine M with oracle B $\subseteq \Gamma^*$ lets you include the following kind of branching instructions:

"if (z in B) then <do something>
else <do something else>"

where z is some string defined earlier in pseudocode. By definition, the oracle TM can always check the condition (z in B) in one step

This notion makes sense even if B is not decidable!

Definition: A is recognizable with B if there is an *oracle TM M with oracle B* that recognizes A

Definition: A is decidable with B if there is an *oracle TM M with oracle B* that decides A

Language A "Turing-Reduces" to B

$$\mathbf{A} \leq_{\mathrm{T}} \mathbf{B}$$

A_{TM} is decidable with $HALT_{TM}$ ($A_{TM} \leq_T HALT_{TM}$)

We can decide if M accepts w using an ORACLE for the Halting Problem:

On input (M,w),
If (M,w) is in $HALT_{TM}$ then
run M(w) and output its answer.
else REJECT.

$HALT_{TM}$ is decidable with A_{TM} ($HALT_{TM} \leq_T A_{TM}$)

On input (M,w), decide if M halts on w as follows:

1. **If (M,w) is in A_{TM} then ACCEPT**
2. **Else, switch the accept and reject states of M to get a machine M′. If (M',w) is in A_{TM} then ACCEPT**
3. **REJECT**

$\leq_T$ versus $\leq_m$

Theorem: If $A \leq_m B$ then $A \leq_T B$

Proof (Sketch):

If $A \leq_m B$ then there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

To decide A on the string w,
just compute f(w) and "call the oracle" for B

Theorem: $\neg HALT_{TM} \leq_T HALT_{TM}$

Theorem: $\neg HALT_{TM} \not\leq_m HALT_{TM}$ ***Why?***

Limitations on Oracle TMs!

The following problem cannot be decided by any TM with an oracle for the Halting Problem:

SUPERHALT = { (M,x) | M, with an oracle for the Halting Problem, halts on x}

We can use the proof by diagonalization!

Assume H (with HALT oracle) decides SUPERHALT

Define D(X) := "if H(X,X) (with HALT oracle) accepts then LOOP, else ACCEPT."

(D uses a HALT oracle to simulate H)

But D(D) halts ⇔ H(D,D) accepts ⇔ D(D) loops…
(by assumption) (by def of D)

Limits on Oracle TMs

"Theorem" There is an *infinite hierarchy* of unsolvable problems!

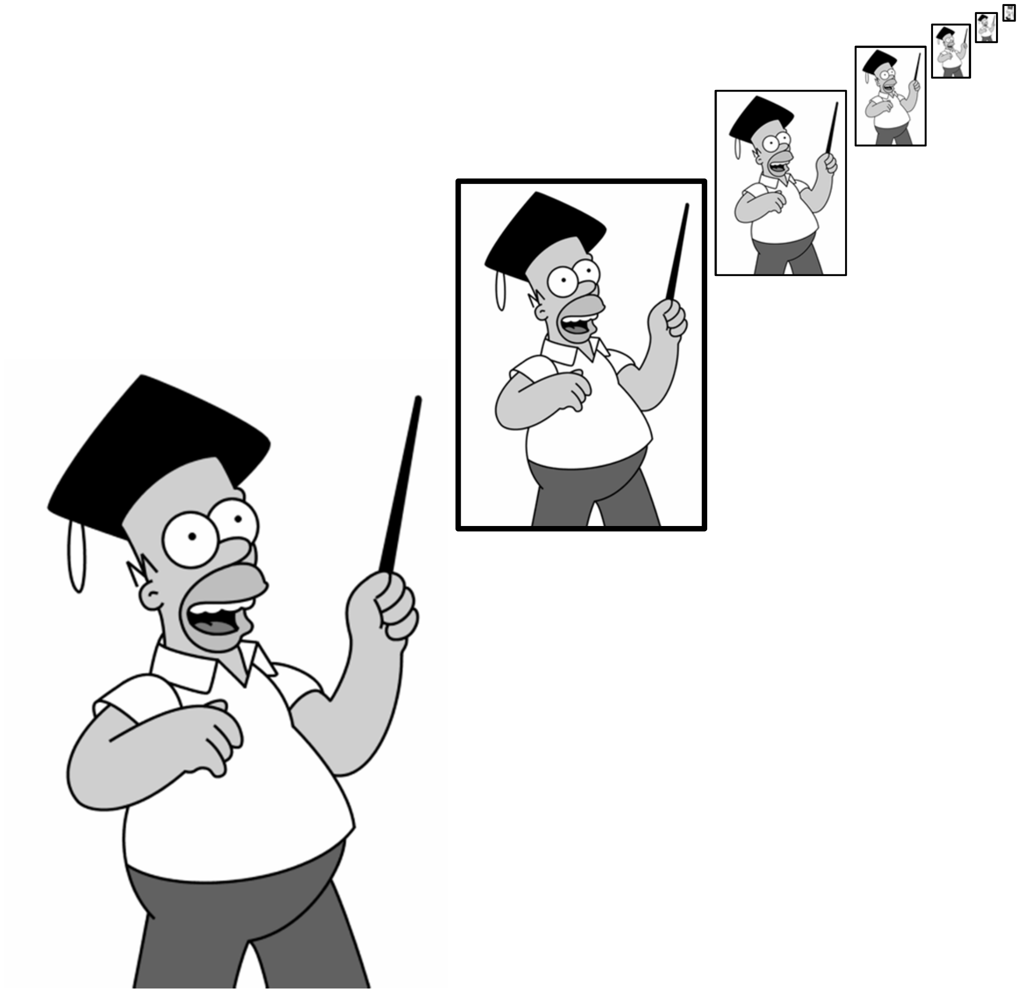
Given ANY oracle O, there is always a <u>harder</u> problem that cannot be decided with that oracle O

SUPERHALT^0 = HALT = { (M,x) | M halts on x}.

SUPERHALT^1 = { (M,x) | M, with an oracle for HALT_{TM}, halts on x}

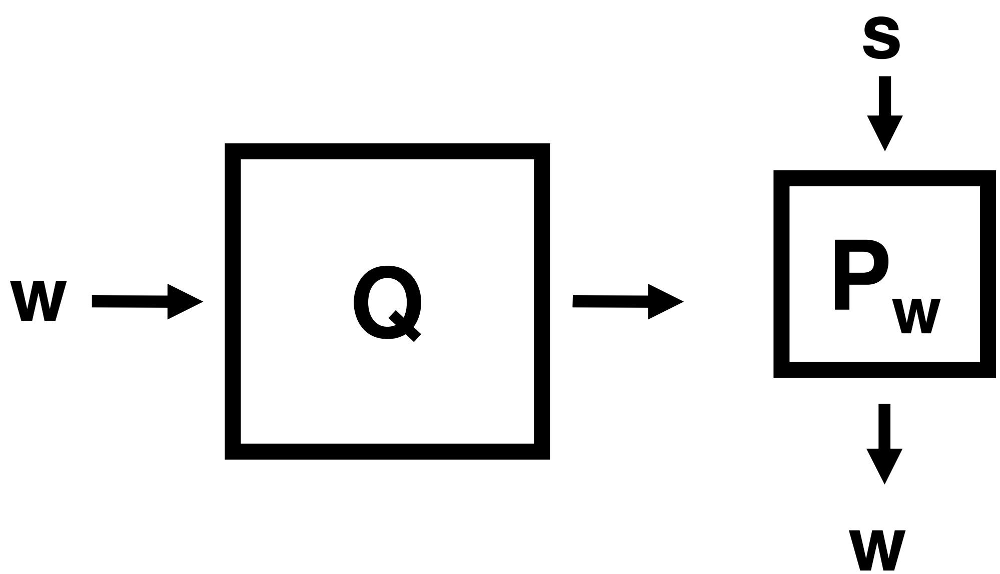
SUPERHALT^n = { (M,x) | M, with an oracle for SUPERHALT^{n-1}, halts on x}

Self-Reference and the Recursion Theorem



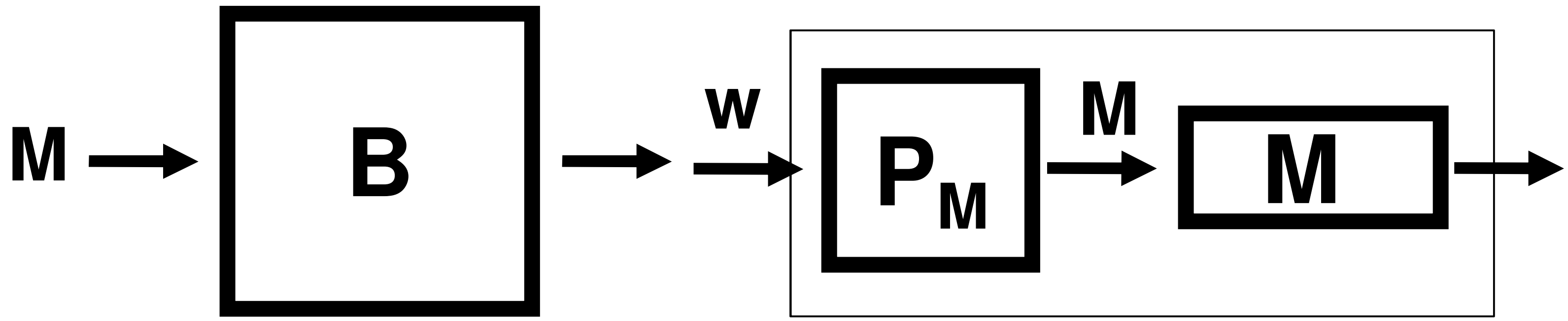
Lemma: There is a computable function $q : \Sigma^* \rightarrow \Sigma^*$ such that for every string w, q(w) is the *description* of a TM P_w that on every input, prints out w and then accepts

"Proof" Define a TM Q:

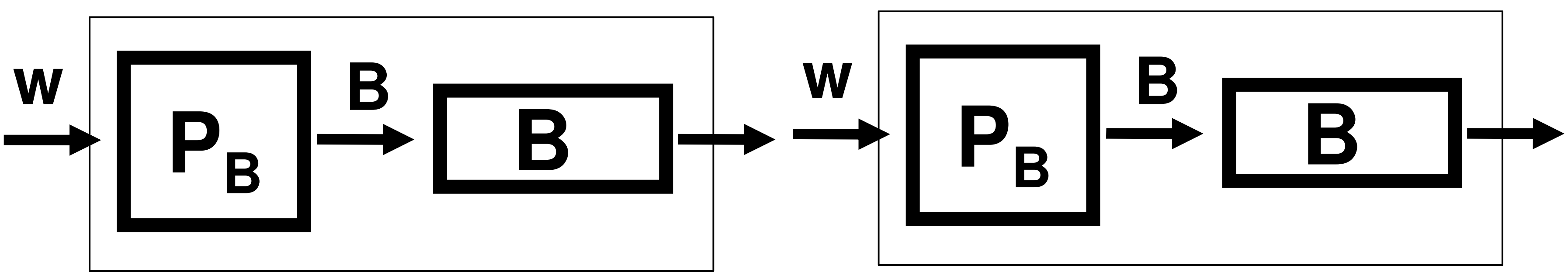


Theorem: There is a Self-Printing TM

Proof: First define a TM B which does this:



Now consider the TM that looks like this:



No explicit self-reference here! **QED**

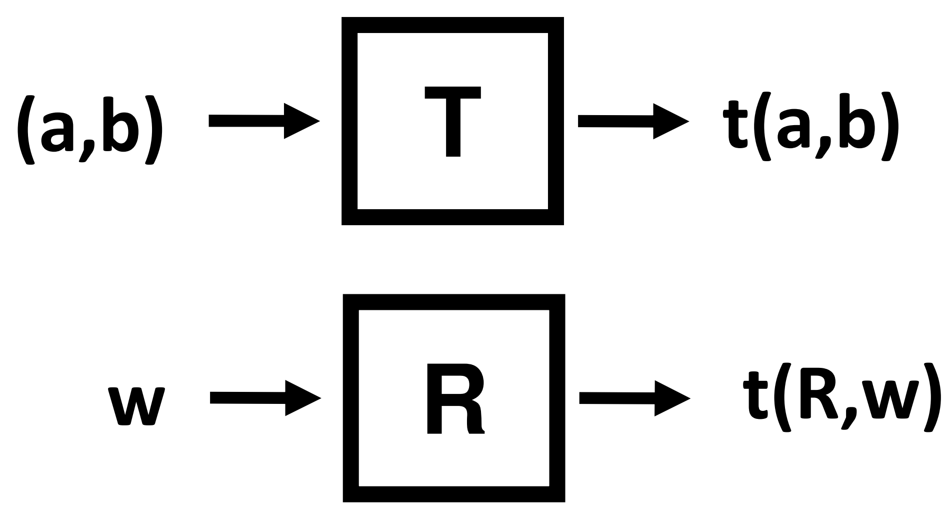
The Recursion Theorem

Theorem: For every TM T computing a function

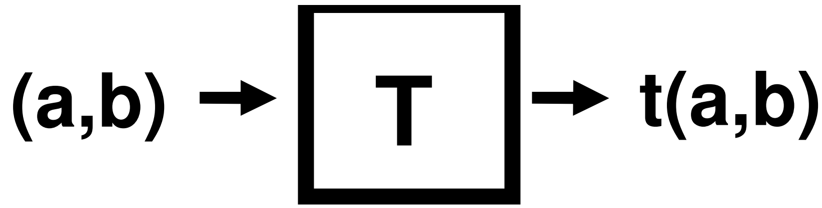
$$t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

there is a Turing machine R computing a function $r : \Sigma^* \rightarrow \Sigma^*$, such that for every string w,

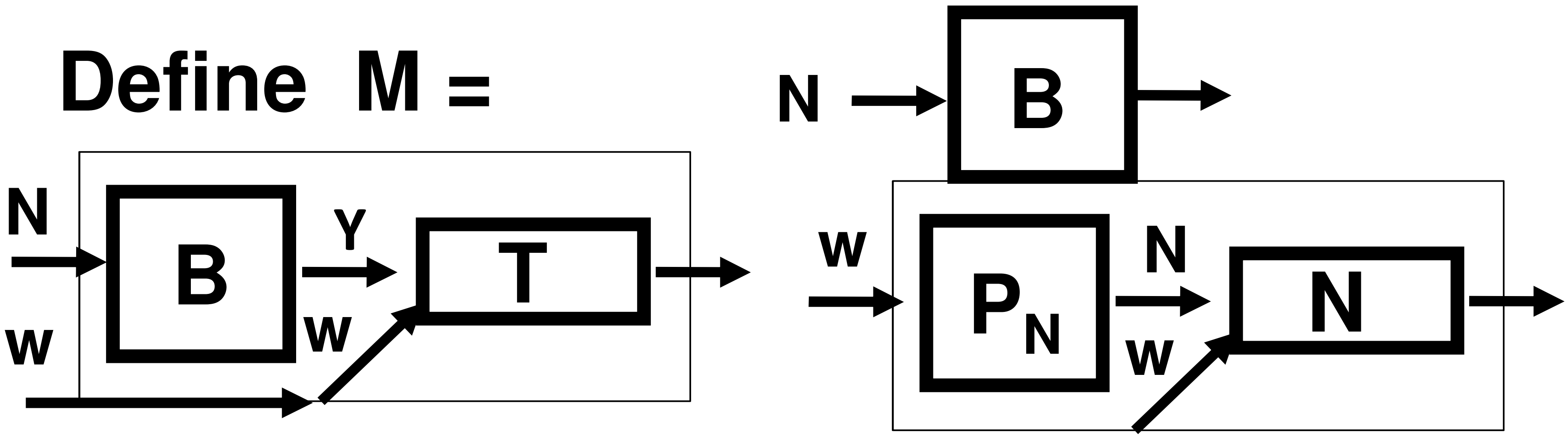
$$r(w) = t(R, w)$$



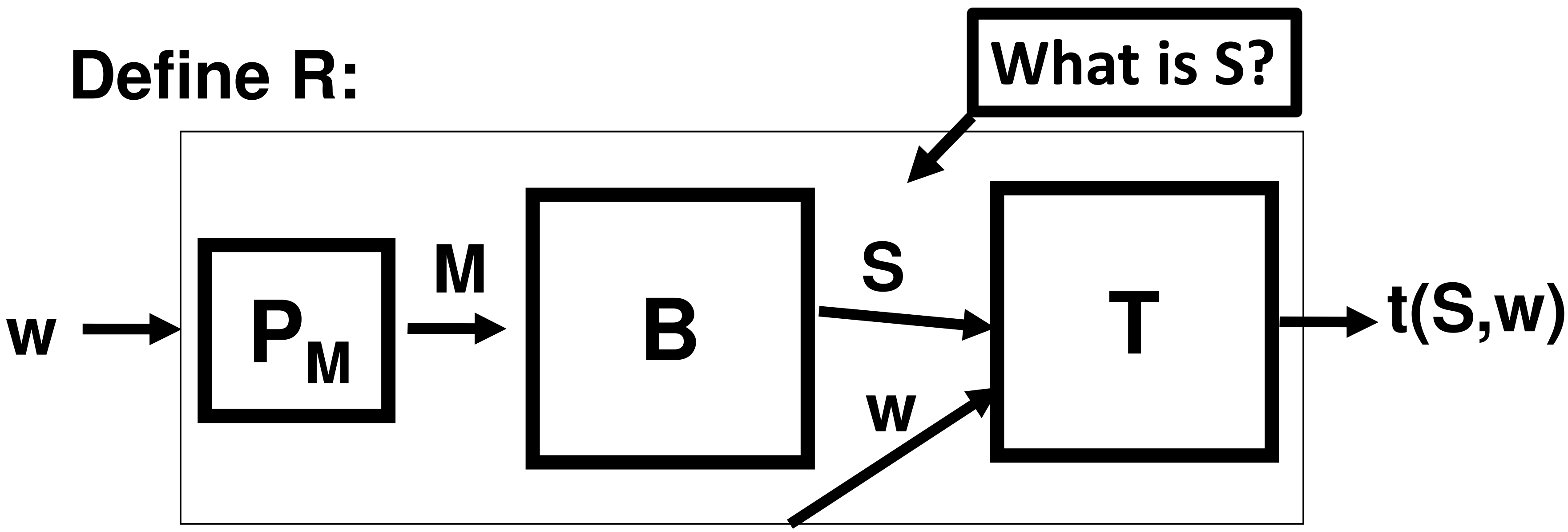
Proof:



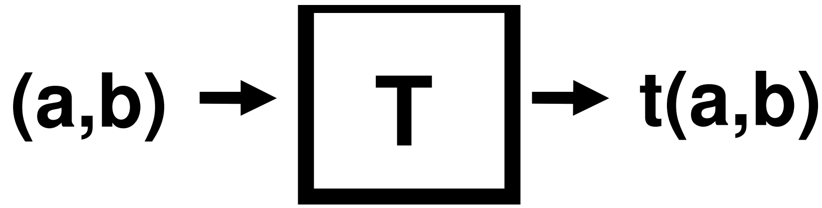
Define M =



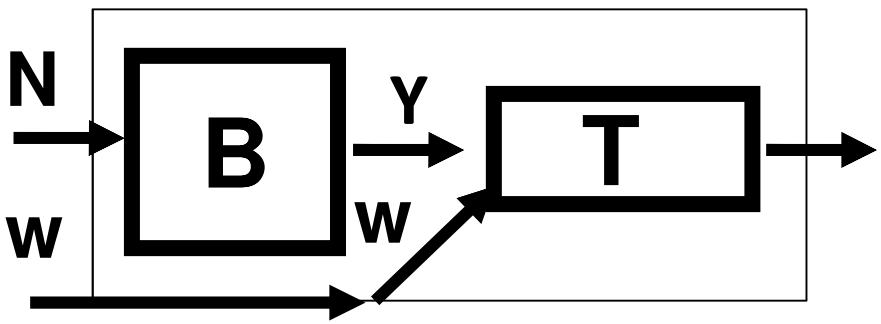
Define R:

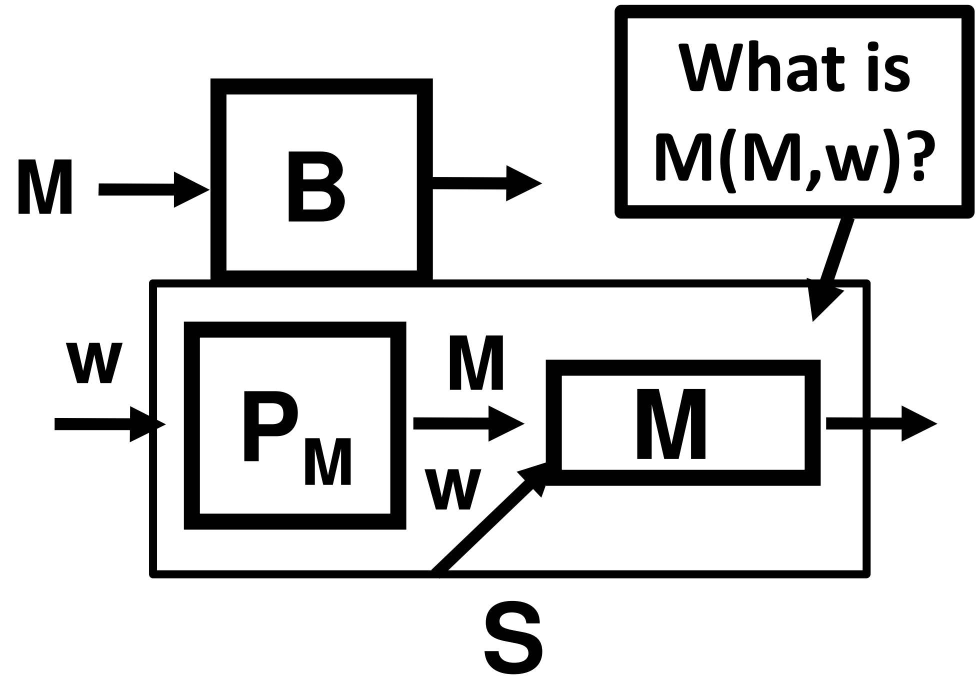


Proof:

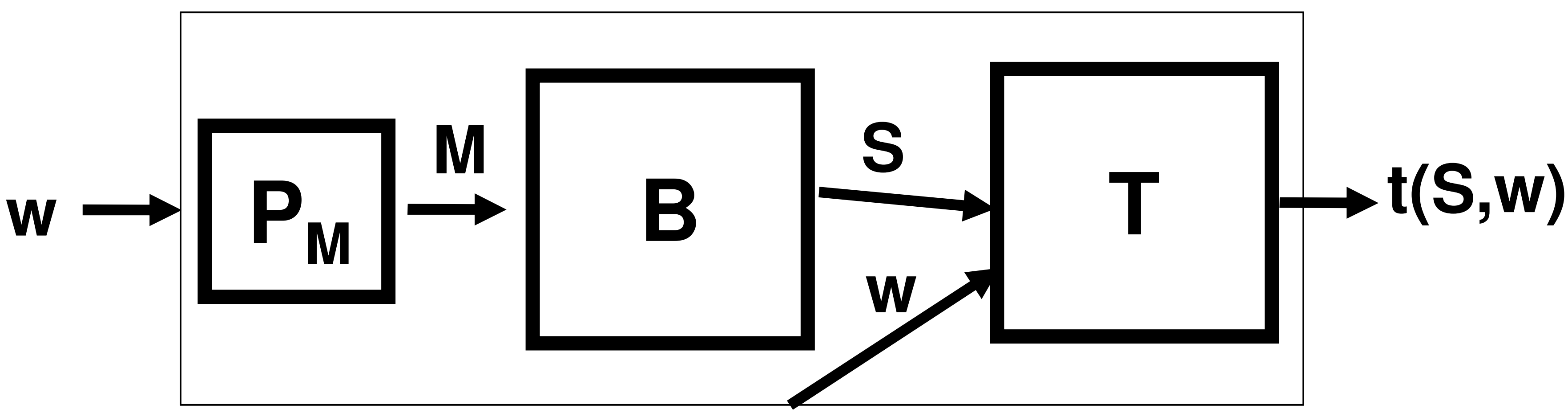


Define M =

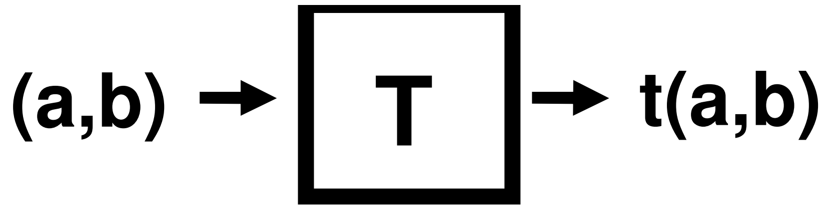


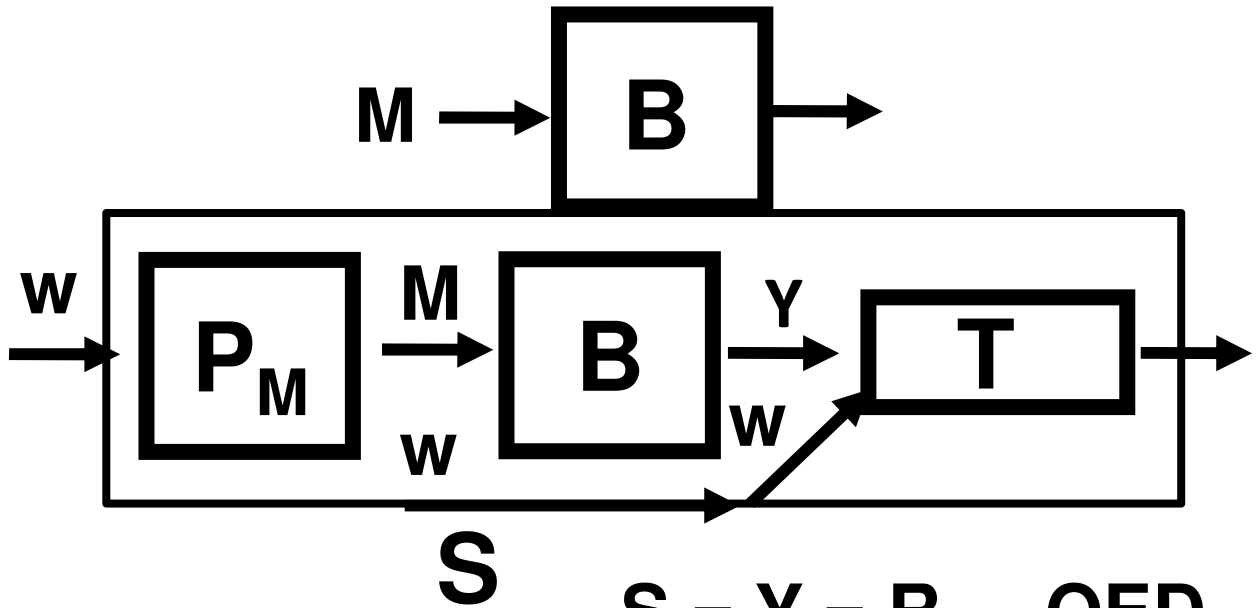


Define R:



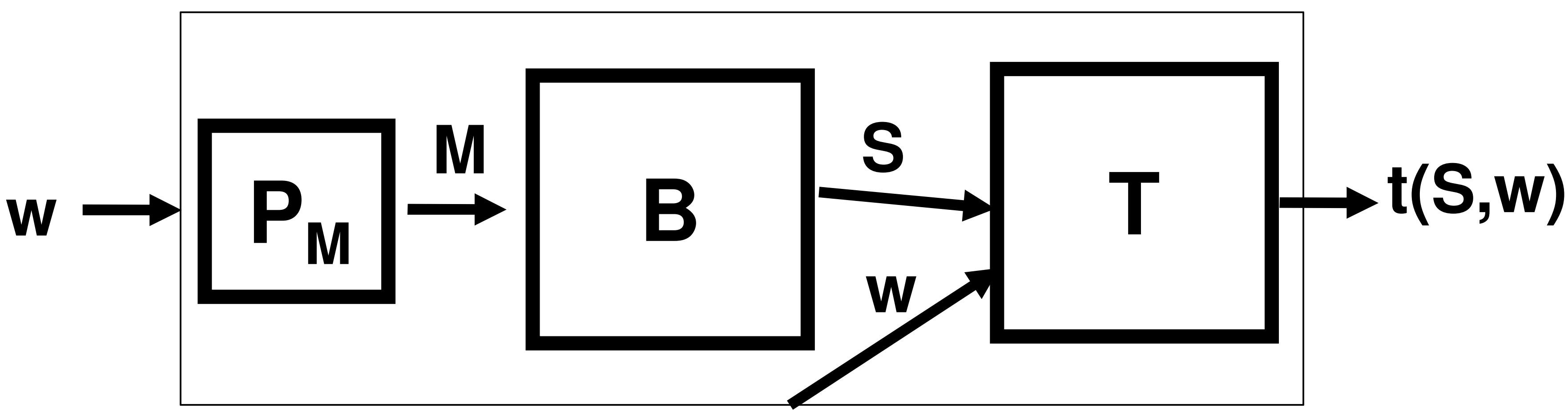
Proof:





S = Y = R. QED

Define R:



For every computable t, there is a computable r such that r(w) = t(R,w) where *R is a description of r*

Suppose we can design a TM T of the form:
"On input (x,w), do bla bla with x,
do bla bla bla with w, etc. etc."
We can then find a TM R with the *behavior*:
"On input w, do bla bla with (a description of R),
do bla bla bla with w, etc. etc."

We can use the operation:
"Obtain your own description"
in Turing machine pseudocode!

Theorem: A_{TM} is undecidable

Proof (using the recursion theorem)

Assume H decides A_{TM}

Construct machine B such that on input w:

1. Obtains its own description B
2. Runs H on (B, w) and flips the output

Running B on input w always does the opposite of what H says it should!

A formalization of “free will” paradoxes!

No single machine can predict behavior of all others