The RECURSION THEOREM: Code is data.

P: program $\qquad$ ⟨P⟩: text (code) of the program

We assume two new primitives:

Obtain ⟨P⟩ : allows a program access to its own source code

Run ⟨P⟩ on x: allows a program to call itself.

EXAMPLE I: P_1

```
Obtain <P1>;
Output <P1>
```

This is a simple self-reproducing program.

EXAMPLE II: P_2:

```
if w = ε then output 0
else { obtain <P2>;
       Run <P2> on tail(w);
       If P2(tail(w)) returns n then
          return (n+1); }
```

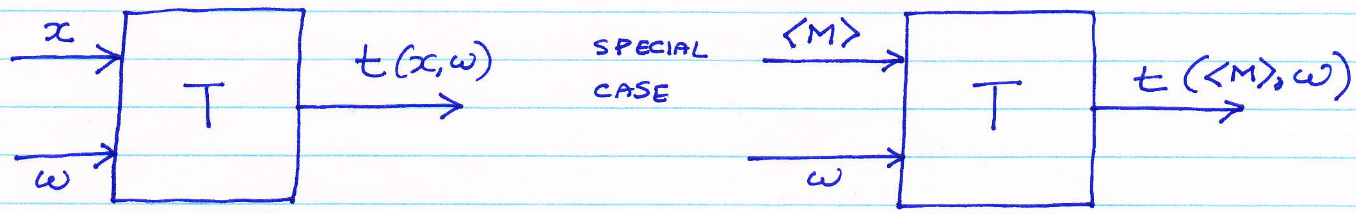
This program is a recursive program to compute the length of its input:

```
fun length [] = 0
  | length (x::xs) = 1 + length (xs)
```

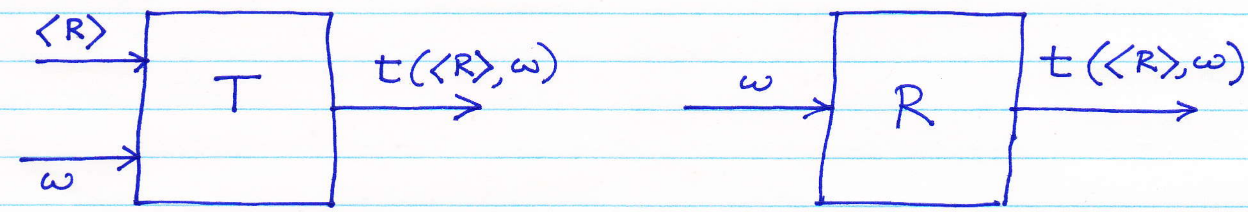
The recursion theorem says: This can be simulated by ordinary Turing machines.

THM: Let T be a TM that computes $t : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$. There is another TM R that computes $r : \Sigma^* \longrightarrow \Sigma^*$, where $\forall w \in \Sigma^*$ $r(w) = t(\langle R \rangle, w)$.

REMARK: r & t may be partial functions.



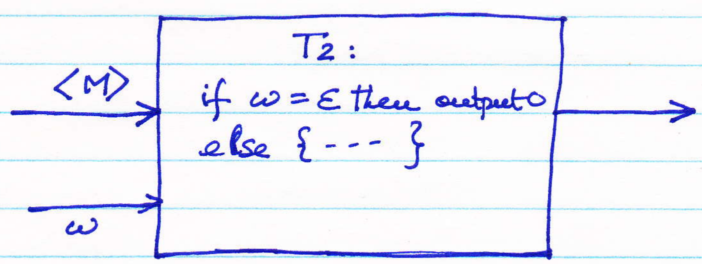
How does R behave?



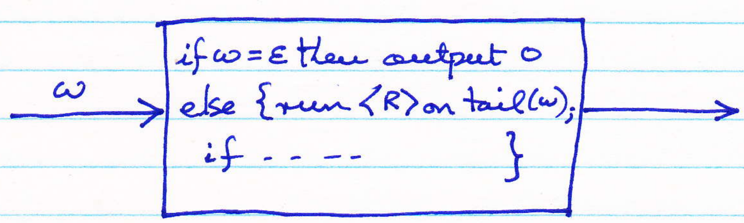
R behaves like T with its first argument fixed to be its own source code.

Let us analyze the set-up for recursive program P_2:

$T_2(\langle M\rangle, \omega)$: if $\omega = \varepsilon$ then output 0
else {run $\langle M\rangle$ on tail(ω);
if output of M(tail(ω)) = n
then output n+1 }



We want to feed T_2 its own source code but the types don't quite match. The recursion them says
$\exists R$ s.t. run $R(\omega) = r(\omega) = t(\langle R\rangle, \omega) =$ run T_2 on $\langle R\rangle, \omega)$



The output is just what we expect from P_2

Ordinary recursive programs can be coded with Turing machines.

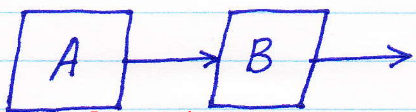
PROOF OF SPECIAL CASE OF THE RECURSION THEOREM namely P_1.

__Lemma__ [6.1 in SIPSER] There is a __total__ computable function

$q : \Sigma^* \to \Sigma^*$ such that, for any string w, $q(w)$ is the description of a TM that outputs w & halts no matter what its input is: call this P_w. $q(w) = \langle P_w \rangle$.

__PROOF__ Straightforward.

Now, back to our special case:



B reads output of A
We write $A; B$

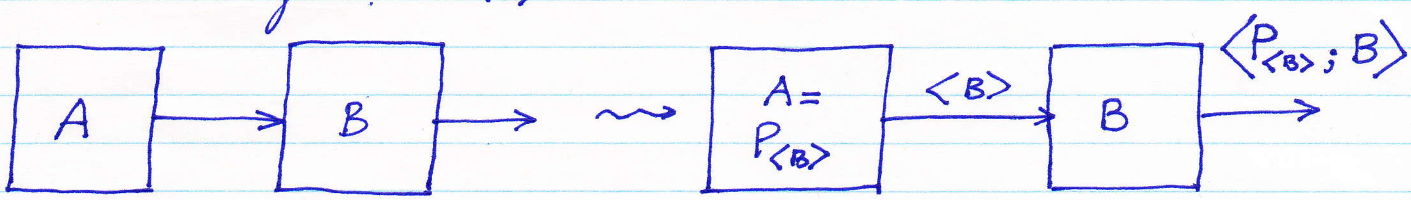
B expects the output to be a TM description $\langle M \rangle$

B outputs $\langle P_{\langle M \rangle}; M \rangle$: easy use q to produce $\langle P_{\langle M \rangle} \rangle$

& package this with $\langle M \rangle$.

So we can describe B; its description is $\langle B \rangle$.

A is just $P_{\langle B \rangle}$

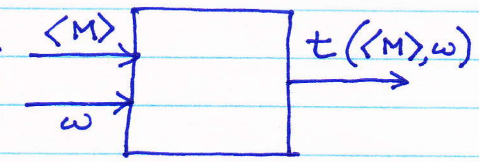


So the output is $\langle P_{\langle B \rangle}; B \rangle = \langle A; B \rangle$!!

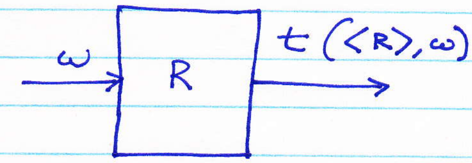
We have our self-reproducing program.

Recall the statement of the theorem:

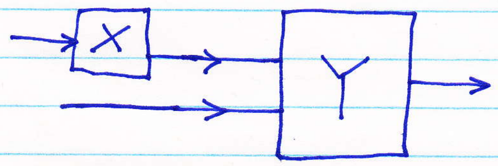
$\forall T$ s.t.



$\exists R$ s.t.

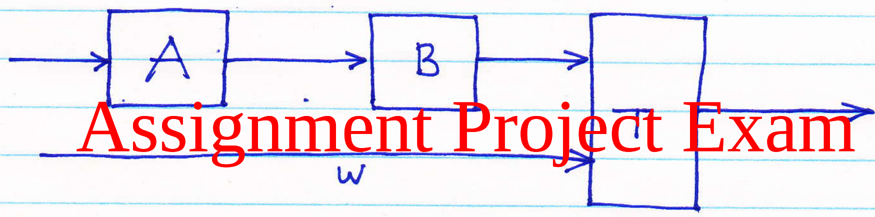


Small variation of the previous proof



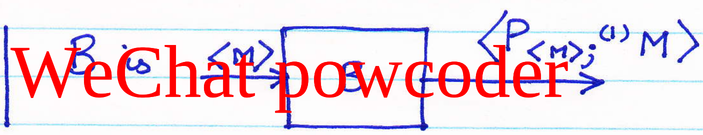
$X ;^{(1)} Y$ notation for the picture

CONSTRUCTION OF R:

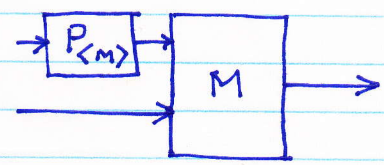


$$(A;B);^{(1)} T = A;^{(1)} (B;^{(1)} T)$$

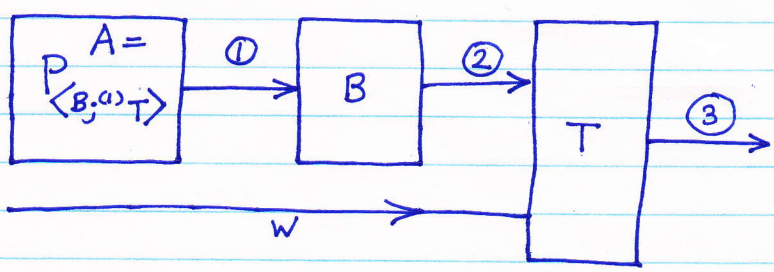
A is $P_{\langle B;^{(1)} T\rangle}$ | B:



The output of B is a <u>description of</u>



R is $(A;B);^{(1)} T$



① OUTPUT OF A $= \langle B;^{(1)} T\rangle$

② OUTPUT OF B $= \langle P_{\langle B;^{(1)} T\rangle} ;^{(1)} (B;^{(1)} T)\rangle = \langle (P_{\langle B;^{(1)} T\rangle} ; B);^{(1)} T\rangle$

$= \langle (A;B);^{(1)} T\rangle = \langle R\rangle$

③ OUTPUT IS $= t(\langle R\rangle, \omega)$

EXAMPLE

$MIN_{TM} = \{\langle M \rangle \mid$ No TM with a shorter encoding recognizes the same language $\}$

Suppose MIN_{TM} is CE so there is an enumerator E.

DEFINE R (USING THE RECURSION THM) as follows:

- Obtain $\langle R \rangle$
- Run E producing $\langle M_1 \rangle, \langle M_2 \rangle, \langle M_3 \rangle, \cdots$ until you find M_i s.t. $|\langle M_i \rangle| > |\langle R \rangle|$.
- Run M_i on w & do whatever M_i does.

Now $L(R) = L(M_i)$ but $|\langle R \rangle| < |\langle M_i \rangle|$ so M_i is not minimal $\otimes$.

Thus MIN_{TM} is not even C.E.

The essence of recursion is fixed point theory. The recursion theorem is a fixed point theorem and can be proved in the context of partial computable functions.

If $G(\cdot,\cdot)$ is a Gödel universal function & $\sigma : \mathbb{N} \to \mathbb{N}$ is a total computable function then there is some n s.t

$$\forall x \in \mathbb{N} \qquad G(n, x) = G(\sigma(n), x)$$

i.e. n & $\sigma(n)$ define the same function.

The proof is not any harder that the proof of the recursion theorem in these notes. I have put a latexed version of it on the course web site.

Notice one consequence; if we think of $\sigma : \Sigma^* \to \Sigma^*$ then given any prog. language & any string mangling function there is some code that gets to other code with exactly the same behaviour!