Self-Reference

Lemma: For any string \( w \), we can design a Turing machine \( P_w \) that prints \( w \) and then halts.

Reworded: There is a computable function from strings to strings, \( q : \Sigma^* \rightarrow \Sigma^* \), so that \( q(w) \) is the description of a Turing machine \( P_w \) that prints \( w \) and then halts.

TM \( Q \) computes \( q(w) \):
On input string \( w \):
1. Construct the following TM \( P_w \):
   “On any input
   1. Erase input
   2. Write \( w \) on tape
   3. Halt.”
2. Output \( \langle P_w \rangle \)

Constructing a Self-Printing Turing Machine

- We compose \( SELF \) of two parts, \( A \) and \( B \), so \( SELF \) prints out \( \langle SELF \rangle = \langle AB \rangle \)
- \( A \) should print \( B \); just take \( A = P_{\langle B \rangle} \).
  
  so need description of \( B \).
- Can’t define \( B \) based on \( A \) in the same way (circular)
- But can have \( B \) compute \( A \) from the output of \( A \! \)!
- \( \langle B \rangle \) was left on tape when \( A \) finished
- so \( B \) can obtain its own description \( \langle B \rangle \)
- \( B \) can compute \( q(\langle B \rangle) = A \)
- \( B \) combines \( A \) and \( B \) and writes \( \langle AB \rangle = \langle SELF \rangle \) on tape
**Self-Printing Turing Machine: Definition**

\[ B = \text{Given input } \langle M \rangle \text{ describing TM (fragment) } M: \]

1. Compute \( q(\langle M \rangle) \)
2. Combine result with \( \langle M \rangle \) into a TM
3. Print description of this TM and halt

\[ A = P_B(\langle \rangle) \]

**Figure 6.2**

Schematic of \( SELF \), a TM that prints its own description

Control starts at \( A \), who produces \( B \) and passes control.

**Recursion Theorem**

\[ A \rightarrow B \rightarrow T \]

\[ (= P_{(BT)}) \]

Control for \( R \)

Construction in three parts:

- \( T \) is the given TM.
- \( A \) is \( q(\langle BT \rangle) \)
  - we want to preserve the input \( w \)
  - change \( q \) to append TM description to tape content
  - after running \( A \), we have \( w(\langle BT \rangle) \) on tape

- \( B \): read tape, apply \( q \), get \( \langle A \rangle \)
  - combine \( A, B, T \) into single machine
  - encode this with \( w \) on tape: \( \langle R, w \rangle \), pass control to \( T \)

**Minimal Turing Machines**

**Def.** Say \( M \) is minimal if there is no equivalent TM with a shorter description than \( \langle M \rangle \) (number of symbols).

Let \( MIN_TM = \{ \langle M \rangle | M \text{ is a minimal TM} \} \).

**Theorem:** \( MIN_TM \) is not Turing-recognizable.

Proof: by contradiction. Assuming an enumerator \( E \), construct \( C \):

\( C = \text{“On input } w: \)

1. Obtain own description \( \langle C \rangle \) via recursion theorem
2. Run enumerator \( E \) until getting a machine \( D \) longer than \( C \)
3. Simulate \( D \) on \( w \)

\( D \) exists, since \( MIN_TM \) is infinite.

Since \( C \) simulates \( D \), they are equivalent.

But \( C \) is shorter than \( D \), so \( D \) should not be in \( E \)'s list.

**Revisiting: \( A_{TM} \) is undecidable**

Assume decider \( H(\langle M, w \rangle) \).

Previous proof: Construct \( Test(\langle M \rangle) = \neg H(\langle M, \langle M \rangle \rangle) \)

Run \( Test(\langle Test \rangle) = \neg H(\langle Test, \langle Test \rangle \rangle) = \neg Test(\langle Test \rangle) \), contradiction

New proof: Construct the following TM \( B \):

On input \( w \):

1. Obtain own description \( \langle B \rangle \) via recursion theorem
2. Run \( H \) on input \( \langle B, w \rangle \)
3. Do the opposite of \( H \) (accept/reject)

\( B \) does the opposite of what \( H \) says it does: contradiction

**Fixed Point Version of Recursion Theorem**

**Def.** A fixed point (fixpoint) of a function \( f \) is a value \( x \) that is unchanged by applying the function: \( f(x) = x \)

For any transformation on Turing machines, there is a TM unchanged by that transformation. Formally:

**Theorem:** Let \( t : \Sigma^* \rightarrow \Sigma^* \) be a computable function.
Then there is a TM \( F \) for which \( t(\langle F \rangle) \) is equivalent to \( F \).

\( F = \text{“On input } w: \)

1. Obtain own description \( \langle F \rangle \) via recursion theorem
2. Compute \( t(\langle F \rangle) \) to obtain a TM \( G \)
3. Simulate \( G \) on \( w \).

Since \( F \) simulates \( G \), they are equivalent!