RECURSION THEOREM

21 February 2024 07:23 PM

SELF-REFERENCE

Let's begin by making a Turing machine that ignores its input and prints out a copy of its own description. We call this machine $SELF$. To help describe $SELF$, we need the following lemma.

LEMMA 6.1

There is a computable function $q\colon \Sigma^* \longrightarrow \Sigma^*$, where if w is any string, $q(w)$ is the description of a Turing machine P_w that prints out w and then halts.

PROOF Once we understand the statement of this lemma, the proof is easy. Obviously, we can take any string w and construct from it a Turing machine that has w built into a table so that the machine can simply output w when started. The following TM Q computes $q(w)$.

Q = "On input string w:
1. Construct the following Turing machine P_w.
   P_w = "On any input:
   1. Erase input.
   2. Write w on the tape.
   3. Halt."
2. Output $\langle P_w \rangle$."

The Turing machine $SELF$ is in two parts, A and B. We think of A and B as being two separate procedures that go together to make up $SELF$. We want $SELF$ to print out $\langle SELF \rangle = \langle AB \rangle$.

Part A runs first and upon completion passes control to B. The job of A is to print out a description of B, and conversely the job of B is to print out a description of A. The result is the desired description of $SELF$. The jobs are

For A we use the machine $P_{\langle B \rangle}$, described by $q(\langle B \rangle)$, which is the result of applying the function q to $\langle B \rangle$. Thus part A is a Turing machine that prints out $\langle B \rangle$. Our description of A depends on having a description of B. So we can't complete the description of A until we construct B.

defined in terms of B. That would be a *circular* definition of an object in terms of itself, a logical transgression. Instead, we define B so that it prints A by using a different strategy: B *computes* A from the output that A produces.

$A = P_{\langle B \rangle}$, and

B = "On input $\langle M \rangle$, where M is a portion of a TM:
1. Compute $q(\langle M \rangle)$.
2. Combine the result with $\langle M \rangle$ to make a complete TM.
3. Print the description of this TM and halt."

If we now run $SELF$ we observe the following behavior.

1. First A runs. It prints $\langle B \rangle$ on the tape.
2. B starts. It looks at the tape and finds its input, $\langle B \rangle$.
3. B calculates $q(\langle B \rangle) = \langle A \rangle$ and combines that with $\langle B \rangle$ into a TM description, $\langle SELF \rangle$.
4. B prints this description and halts.

THEOREM 6.3

Recursion theorem Let T be a Turing machine that computes a function $t\colon \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$. There is a Turing machine R that computes a function $r\colon \Sigma^* \longrightarrow \Sigma^*$, where for every w,

$$r(w) = t(\langle R \rangle, w).$$

Theorem: For any TM T there is a TM R where for all w R on input w operates in the same way as T on input $\langle w, R \rangle$.

PROOF The proof is similar to the construction of $SELF$. We construct a TM R in three parts, A, B, and T, where T is given by the statement of the theorem; a schematic diagram is presented in the following figure.

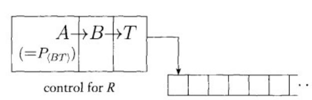
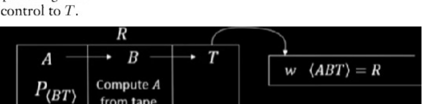


FIGURE 6.4
Schematic of R

Here, A is the Turing machine $P_{\langle BT \rangle}$ described by $q(\langle BT \rangle)$. To preserve the input w, we redesign q so that $P_{\langle BT \rangle}$ writes its output following any string preexisting on the tape. After A runs, the tape contains $w\langle BT \rangle$.

Again, B is a procedure that examines its tape and applies q to its contents. The result is $\langle A \rangle$. Then B combines A, B, and T into a single machine and obtains its description $\langle ABT \rangle = \langle R \rangle$. Finally, it encodes that description together with w, places the resulting string $\langle R, w \rangle$ on the tape, and passes control to T.

Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = P_{\langle BT \rangle}$
B = "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w, R \rangle$."



$SELF$ = "On any input:
1. Obtain, via the recursion theorem, own description $\langle SELF \rangle$.
2. Print $\langle SELF \rangle$."

THEOREM 6.5

A_{TM} is undecidable.

PROOF We assume that Turing machine H decides A_{TM}, for the purposes of obtaining a contradiction. We construct the following machine B.

B = "On input w:
1. Obtain, via the recursion theorem, own description $\langle B \rangle$.
2. Run H on input $\langle B, w \rangle$.
3. Do the opposite of what H says. That is, *accept* if H rejects and

THEOREM 6.5

A_{TM} is undecidable.

PROOF We assume that Turing machine H decides A_{TM}, for the purposes of obtaining a contradiction. We construct the following machine B.

B = "On input w:
1. Obtain, via the recursion theorem, own description $\langle B \rangle$.
2. Run H on input $\langle B, w \rangle$.
3. Do the opposite of what H says. That is, *accept* if H rejects and *reject* if H accepts."

Running B on input w does the opposite of what H declares it does. Therefore H cannot be deciding A_{TM}. Done!

$A = P_{\langle BT \rangle}$

B = "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w, R \rangle$."

$P_{\langle BT \rangle}$	Compute A from tape	

DEFINITION 6.6

If M is a Turing machine, then we say that the ***length*** of the description $\langle M \rangle$ of M is the number of symbols in the string describing M. Say that M is ***minimal*** if there is no Turing machine equivalent to M that has a shorter description. Let

$$MIN_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a minimal TM}\}.$$

THEOREM 6.7

MIN_{TM} is not Turing-recognizable.

PROOF Assume that some TM E enumerates MIN_{TM} and obtain a contradiction. We construct the following TM C.

C = "On input w:
1. Obtain, via the recursion theorem, own description $\langle C \rangle$.
2. Run the enumerator E until a machine D appears with a longer description than that of C.
3. Simulate D on input w."

Because MIN_{TM} is infinite, E's list must contain a TM with a longer description than C's description. Therefore step 2 of C eventually terminates with some TM D that is longer than C. Then C simulates D and so is equivalent to it. Because C is shorter than D and is equivalent to it, D cannot be minimal. But D appears on the list that E produces. Thus we have a contradiction.

THEOREM 6.8

Let $t\colon \Sigma^* \longrightarrow \Sigma^*$ be a computable function. Then there is a Turing machine F for which $t(\langle F \rangle)$ describes a Turing machine equivalent to F. Here we'll assume that if a string isn't a proper Turing machine encoding, it describes a Turing machine that always rejects immediately.

Theorem: For any computable function $f\colon \Sigma^* \to \Sigma^*$, there is a TM R such that $L(R) = L(S)$ where $f(\langle R \rangle) = \langle S \rangle$.

PROOF Let F be the following Turing machine.

F = "On input w:
1. Obtain, via the recursion theorem, own description $\langle F \rangle$.
2. Compute $t(\langle F \rangle)$ to obtain the description of a TM G.
3. Simulate G on w."

Clearly, $\langle F \rangle$ and $t(\langle F \rangle) = \langle G \rangle$ describe equivalent Turing machines because F simulates G.

Check-in 11.1

Implementations of the Recursion Theorem have two parts, a Template and an Action. In the TM and English implementations, which is the Action part?

(a) A and the upper phrase
(b) A and the lower phrase
(c) B and the upper phrase
(d) B and the lower phrase.

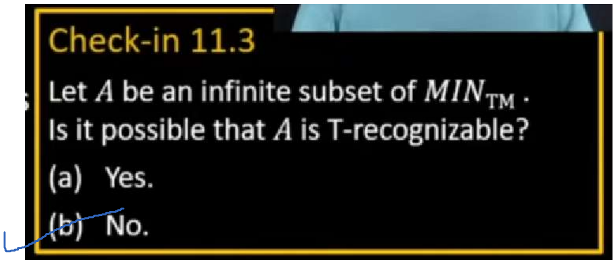
Write the following twice, the second time in quotes → action
"Write the following twice, the second time in quotes"
Write the following twice, the second time in quotes
"Write the following twice, the second time in quotes"

Check-in 11.2

Can we use the Recursion Theorem to design a TM T where $L(T) = \{\langle T \rangle\}$?

(a) Yes.
(b) No.

MIT OCW



You can have lang which are not turing recog but have infinite turing recog subsets