E 2.1

Exercise (Example 8.2 from textbook)

Construct a Turing Machine accepting the language

$\{0^n 1^n \mid n \geq 1\}$

solution

The idea is that the TM M that we construct reads the leftmost 0, turns it into X, and moves right until it reaches a 1, that is turned into Y. Then the head moves left again to the leftmost 0 (on the right to a X), and starts again until all 0's and 1's are turned into X's and Y's respectively.
If the input is not in $0^* 1^*$, M will fail to find a move and it won't accept. If M changes the last 0 and the last 1 in the same round, it will go into the final state and accept.

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, X, Y, b\}$ (b denotes blank symbol)

q_0 : start state

$F = \{q_4\}$

In q_0 is the state in which M is when the head precedes the leftmost 0. In state q_1, M moves right skipping 0's and Y's until it gets to a 1. In state q_2, M moves left while skipping Y's and 0's again, until it gets to a X and goes again in q_0.

E 2.2

Starting from q_0, if a Y is read instead of a 0, M goes in q_3 and moves right: if a 1 is found, then there are more 1's than 0's; if a $\not{b}$ is read, then the initial string is accepted (transition to q_4).

	0	1	X	Y	$\not{b}$
q_0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q_1	$(q_1, 0, R)$	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	$(q_2, 0, L)$	—	(q_0, X, R)	(q_2, Y, L)	—
q_3	—	—	—	(q_3, Y, R)	$(q_4, \not{b}, R)$
q_4	—	—	—	—	—

Exercise

Show the computation of the TM above when the input string is:

(a) 00

(b) 000111

solution

(a) $q_0 00 \vdash X q_1 0 \vdash X 0 q_1$

and the TM halts

(b) $q_0 000111 \vdash X q_1 00111 \vdash X0 q_1 0111 \vdash$
$X00 q_1 111 \vdash X0 q_2 0Y11 \vdash X q_2 00Y11 \vdash q_2 X00Y11 \vdash$
$X q_0 00Y11 \vdash XX q_1 0Y11 \vdash XX0 q_1 Y11 \vdash XX0Y q_1 11 \vdash$
$XX0 q_2 YY1 \vdash XX q_2 0YY1 \vdash X q_2 X0YY1 \vdash XX q_0 0YY1 \vdash$
$XXX q_1 YY1 \vdash XXXY q_1 Y1 \vdash XXXYY q_1 1 \vdash XXXY q_2 YY \vdash$
$XXX q_2 YYY \vdash XX q_2 X YYY \vdash XXX q_0 YYY \vdash XXXY q_3 YY \vdash$
$XXXYY q_3 Y \vdash XXXYYY q_3 \not{b} \vdash XXXYYY\not{b} q_4 \not{b}$

E 2.3

Exercise (8.1.1 from textbook)

Give a reduction from the hello-world problem to the following problem:

> given a program P and an input I, does the program ever produce any output?

solution

We modify P by making it print its output on some array A, capable of storing 12 characters. When A is full, P checks whether it stores "hello world": if it does, P prints (on the output, not on the array) some character (like @); if not, it does not print anything.

So the modified program prints some output if and only if P prints "hello, world"; if we are able to determine whether a program produces any output, we can solve the hello-world problem. This ends our reduction.

E 2.4

Exercise (8.2.3 from textbook):

Design a Turing Machine that takes as input a number N in binary and turns it into $N+1$ (in binary); the number N is preceded by the symbol \$, which may be destroyed during the computation. For example, \$111 is turned into 1000; \$1001 is turned into \$1010.

solution

The idea is to toggle the rightmost digit, and, from right to left, all consecutive 1's until we get to the first 0 (which is also toggled). If there is no 0 to be toggled, a 1 is added on the left of the first digit (i.e., in place of the \$).

We need three states, where only q_2 is the final state; we briefly describe what the TM does in the different states.

q_0 : the TM goes right until it reaches b, after the rightmost digit. When b is reached, the TM goes into q_1.

q_1 : goes left toggling all 1's and the first 0 (from right); when 0 or \$ is reached, the symbol is turned into 1.

q_2 : final state; the TM does nothing

	\$	0	1	b
q_0	$(q_0, \$, R)$	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, b, L)
q_1	$(q_2, 1, L)$	$(q_2, 1, L)$	$(q_1, 0, L)$	—
q_2	—	—	—	—

E 2.5

Exercise (8.22 from textbook)

Design Turing machines accepting the following languages:

$$\{ w \in \{0,1\}^* \mid w \text{ has an equal number of 0's and 1's} \}$$

Solution

The idea is that the head of our TM M moves back and forth on the tape, "deleting" one 0 for each 1; if there are no 0's and 1's in the end, the string is accepted.
When in state q_1, M has found a 1 and looks for a 0; in state q_2 is the other way around.
Note than the head never moves left of any X, so that there are never unmatched 0's and 1's on the left of an X.
From initial state q_0, M picks up a 0 or a 1 and turns it into X. The only final state is q_4. In state q_3 M moves head left looking for the rightmost X.

	0	1	b	X	Y
q_0	(q_2, X, R)	(q_1, X, R)	(q_4, b, R)	—	(q_0, Y, R)
q_1	(q_3, Y, L)	$(q_1, 1, R)$	—	—	(q_1, Y, R)
q_2	$(q_2, 0, R)$	(q_3, Y, L)	—	—	(q_2, Y, R)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	—	(q_0, X, R)	(q_3, Y, L)
q_4	—	—	—	—	—

<u>Exercise</u> (8.1.1(a) from MHU)

Show that the following problem is undecidable, by giving a reduction from the Hello-World problem:

Given a program P and an input x, does P eventually halt when it is given x as input?

(Note: this problem is called the Halting problem)

<u>Solution:</u>

We have to construct a reduction Red from the HWP to the HP.

Red is a program that:

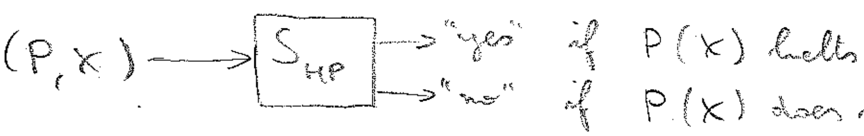
- takes as input an instance (Q, y) of the HWP, and
- produces as output an instance (P, x) of the HP

such that

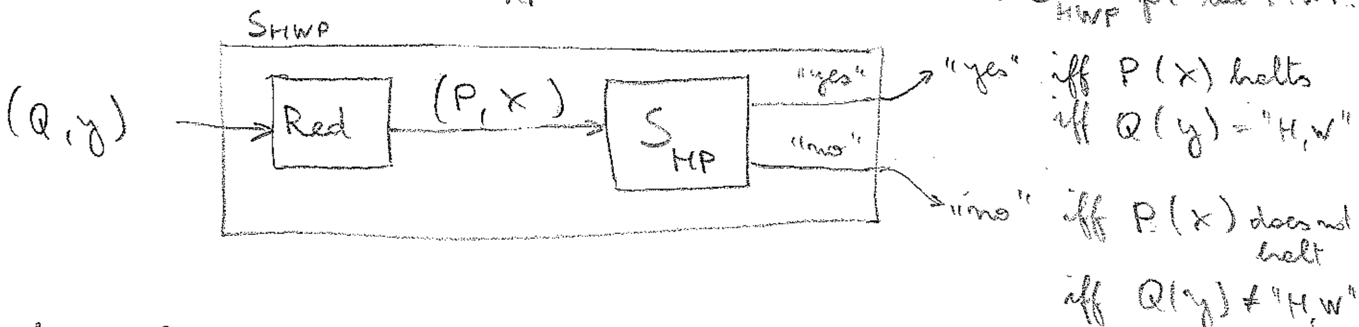
$$Q(y) = \text{"Hello, World"} \quad \text{iff} \quad P(x) \text{ eventually halts.}$$

$$(\text{i.e., } HWP(Q, y) = \text{"yes"} \quad \text{iff} \quad HP(P, x) = \text{"yes"})$$

Using Red, we can show that the HP is undecidable.
Indeed, assume the HP is decidable, and let S_{HP} be a solver for the HP, i.e.



We use Red and S_{HP} to construct a solver S_{HWP} for the HWP:



Since S_{HWP} does not exist, also S_{HP} cannot exist.

OPTIONAL

We show now how to construct Red by describing what it does:

$$(Q, y) \longrightarrow \boxed{\text{Red}} \longrightarrow (P, x)$$

Red leaves y unchanged, i.e. $x = y$

Red performs on Q the following modifications:

1) It makes sure that Q never halts
(e.g. by inserting `while (true) { };` at the end of `main()` and before every `return;` in `main()`)

2) It modifies the `println()` method so that it stores the printed characters in an array, and then checks whether the array contains "Hello, World". If yes, Q halts.

The resulting program is P.

Note that Red, which computes P from Q can be written in Java.