8 Turing Machines

8.1 Construct Turing machines for solving each of the following problems:

a) Accept the language $\{0^m 1^n \mid 0 \leq m \leq n\}$.

b) Accept the language $\{x \in \{a, b\}^* \mid x$ contains the same number of a's and b's$\}$.

c) If the string 1^n is placed on the tape, the TM generates the string $(01)^n$.

d) If the string $1^m \sharp 1^n$ is placed on the tape, the TM generates the string 1^{mn} (i.e. multiplication).

8.2 For the following statements, indicate whether they are true or false and give a justification:

a) The intersection of two recursive languages is recursive.

b) The union of two r.e. (recursively enumerable) languages is r.e..

c) If the complement $\sim L$ of a r.e. language L is recursive, then L is recursive.

d) Let L_1 and L_2 be r.e. languages. Then $L_1 L_2$ is r.e..

8.2 a) Assume that we have two recursive languages L_1, L_2. We show that $L_1 \cap L_2$ is recursive.

L_1, L_2 are defined by some total TM's (Turing machines), $L_1 = L(M_1)$ and $L_2 = L(M_2)$. We can construct a TM M that, on an input string x, behaves in the following way:

- M simulates step-by-step the computation of M_1 on input x until M_1 halts,
- if M_1 rejects x then M halts and rejects,
- otherwise M simulates the computation of M_2 on input x,
- if M_2 accepts x then M halts and accepts,
- otherwise M halts and rejects.

We need x available when starting simulation of M_2; for this purpose a copy of x can be kept at the beginning of the tape. Obviously, x is accepted by M iff it is accepted by M_1 and by M_2. Thus $L(M) = L(M_1) \cap L(M_2)$. Notice that if M_1 and M_2 are total TM's then M is total, thus $L(M)$ is recursive.

b) Assume that we have two r.e. languages L_1, L_2 defined by two TM's; $L_1 = L(M_1)$ and $L_2 = L(M_2)$. We show that $L_1 \cup L_2$ is r.e..

We can construct a TM M that simulates the computations of M_1 and M_2 on the same input string "in parallel", performing one step of M_1, one step of M_2, and so on. M can be a two tape TM, one tape would be the tape of M_1, the other of M_2. If one of the simulated machines accepts, then M accepts. Clearly, $L(M) = L(M_1) \cup L(M_2)$.

Notice that we cannot first perform the computation of M_1 and then of M_2; the first computation may not halt and thus the computation of M_2 may never begin.

c) Construct a total TM for L out of TM's for L and $\sim L$. The construction is similar to the previous one.

d) A TM M for L_1L_2 would first split the input string x into two ($x = y_iz_i$) in all $|x| + 1$ possible ways ($i = 1, \ldots, |x| + 1$). Then for each of them M checks whether $y_i \in L_1$ and $z_i \in L_2$. The checks have to be done in parallel.[1] M accepts when some pair of checks succeeds. (This implies that $x \in L_1L_2$. Conversely, if $x \in L_1L_2$ then at least one pair of checks succeeds.)

We cannot use $2(|x| + 1)$ tapes, as the number of tapes has to be fixed (independent of x). So we split a tape into $2(|x| + 1)$ sections, each section would play the role of the tape of a TM performing one of the checks. If the TM needs more tape cells than assigned, the tape section is enlarged by shifting all the tape sections that are to the right of the given one. The second tape of M would be used for bookkeeping (which section is dealt with now, etc.).

[1] More precisely, for a given splitting $x = x_1x_2$ the two checks for $x_1 \in L_1$ and $x_2 \in L_2$ can be performed sequentially.