4. **Problem:** Prove that the class of recursively enumerable languages is closed under unions and intersections. Why cannot one prove that the class is closed under complements in a similar way as in the case of recursive languages, i.e. simply by interchanging the accepting and rejecting states of the respective Turing machines?

**Solutions:** Let \( L_1 \) and \( L_2 \) be recursively enumerable languages \((L_1, L_2 \in RE)\). Then, there exists Turing machines \( M_1 \) and \( M_2 \) such that \( L(M_1) = L_1 \) and \( L(M_2) = L_2 \). We can now form machines \( M_\cup \) and \( M_\cap \) that recognize the languages \( L_1 \cup L_2 \) and \( L_1 \cap L_2 \).

**Union:** The Turing machine \( M_\cup \) is formed by composing the machines \( M_1 \) and \( M_2 \) as follows:

\[
M_\cup
\]

The machine is nondeterministic and it initially chooses whether it simulates \( M_1 \) or \( M_2 \). Since nondeterministic and deterministic Turing machines have the same expressive power and \( M_\cup \) recognizes \( L_1 \cup L_2 \), \( L_1 \cup L_2 \in RE \).

**Intersection:** The machine \( M_\cap \) is formed as follows:

\[
M_\cap
\]

Given input \( x \), machine \( M_\cap \) first simulates the computation \( M_1(x) \). If it halts (in the accepting state \( q_{acc} \)), \( M_\cap \) continues with computation \( M_2(x) \). If both computations accept, \( x \in L_1 \cap L_2 \) and \( M_\cap \) accepts. Thus, \( L_1 \cap L_2 \in RE \). (To be precise, \( M_\cap \) has to store the input \( x \) somewhere so that it can be given to \( M_2 \) after \( M_1 \) has finished.)

**Complementation:** When a language \( L \) belongs to class \( RE - R \) (= recursively enumerable but not recursive), then a Turing machine \( M \) that recognizes it can reject a word \( x \) in two different ways:

(a) \( M \) halts in state \( q_{rej} \); or
(b) \( M \) does not halt at all.

We now construct a machine \( \overline{M} \) where the accepting and rejecting states are switched. Now also \( \overline{M} \) rejects \( x \) if the computation does not halt, so \( x \notin L(M) \cup L(\overline{M}) \), so \( L(\overline{M}) \neq \overline{L} \).

It can be proved (exercises 1d and 2b) that if \( L \in RE - R \), then \( \overline{L} \notin RE \).

5. **Problem:**

\(^1\) Note that we again suppose that the machines have disjoint sets of states.
(a) Prove that any decision problem that has only finitely many possible inputs is decidable.

(b) Prove that the problem “Does the decimal expansion of \( \pi \) contain 100 consecutive zeros?” is decidable. What does this result tell you about (i) the decimal expansion of \( \pi \), (ii) the notions of decidability and undecidability?

Solution

(a) If a decision problem has a finite number of possible inputs, it is possible to construct a Turing machine that encodes all inputs and the correct answers into its states. Thus, a problem is always decidable if there are only a finite number of inputs.

(b) The decision problem “Does the decimal expansion of \( \pi \) contain 100 consecutive zeros?” has only one input, namely, \( \pi \) so by the preceding point it is decidable.

However, it is not easy to come up with the actual algorithm to compute the answer. The natural way would be to generate the digits of \( \pi \) one by one and check whether there are 100 consecutive zeroes. Unfortunately this only semi-decides the problem; if the answer is “no”, the machine never halts.

The problem has only one answer: either \( \pi \) has 100 consecutive zeroes or it does not have. Thus, one of the following Turing machines decides it:

<table>
<thead>
<tr>
<th></th>
<th>&quot;Yes&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_y )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>&quot;No&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_n )</td>
<td></td>
</tr>
</tbody>
</table>

The machine \( M_y \) accepts the input and \( M_n \) rejects it. Unfortunately, we cannot know which one of the machines is the correct one (though \( M_y \) seems to be more probable). As we see, the concept of decidability is a very weak one. A problem may be decidable even if we cannot solve it in practice, for example, if we do not have enough computational resources.