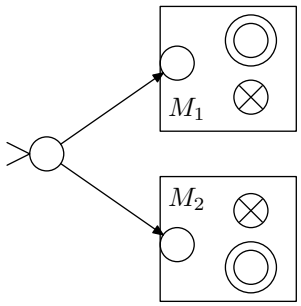
Solutions to Supplementary Problems

S9.1 Prove that the class of semi-decidable languages is closed under unions and intersections. Why can't we prove that the class is closed under complementation in a similar way as in the case of decidable languages, i.e. by simply interchanging the accepting and rejecting states of the respective Turing machines?

Solution. Let L_1 and L_2 be semi-decidable languages. Then, there exist Turing machines M_1 and M_2 such that $\mathscr{L}(M_1) = L_1$ and $\mathscr{L}(M_2) = L_2$. We assume that the two machines have disjoint sets of states. We now present machines $M_\cup$ and $M_\cap$ that recognise 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:

$M_\cup$:

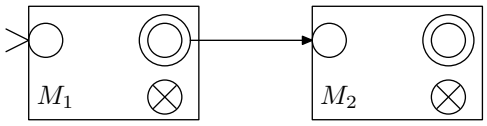


The machine is nondeterministic and it initially chooses whether it simulates M_1 or M_2. Since nondeterministic and deterministic Turing machines have the same expressive power (Theorem 8.3 on the lecture slides) and $M_\cup$ recognises $L_1 \cup L_2$, $L_1 \cup L_2$ is semi-decidable.

As an alternative construction, we could also have built the machine $M_\cup$ so that it runs both M_1 and M_2 in parallel, and stops and accepts whenever one of them stops and accepts.

Intersection: The machine $M_\cap$ is formed as follows:

$M_\cap$:



Given input x, machine $M_\cap$ first performs the computation $M_1(x)$. If that 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, $L_1 \cap L_2$ is semi-decidable. (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. This is easy to arrange if we use a two-tape machine that first backups the input on its second tape.)

Complementation: When a language L is semi-decidable but not necessarily decidable, then a Turing machine M that recognises L can reject an input string x in two different ways:

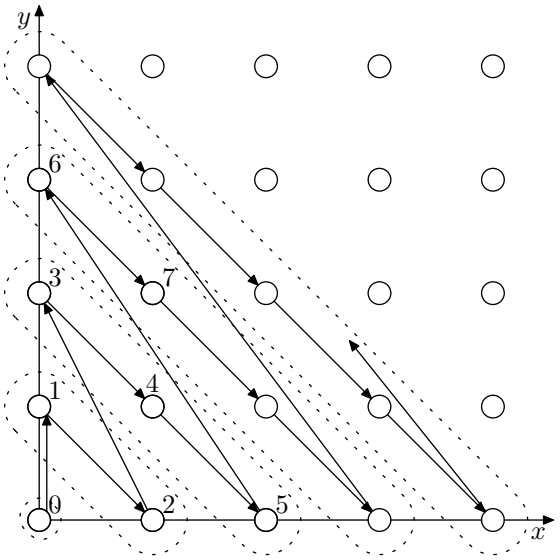
1. M halts in state q_{rej}; or
2. M does not halt at all.

Suppose we construct a machine $\overline{M}$ where the accepting and rejecting states of M are switched. Now, if M does not halt on a given string x, then also $\overline{M}$ does not halt on x. This means that for such an x, $x \notin \mathscr{L}(M)$ but also $x \notin \mathscr{L}(\overline{M})$. But $x \in \overline{\mathscr{L}(M)}$. So $\mathscr{L}(\overline{M}) \neq \overline{\mathscr{L}(M)}$. In fact, if a language L is semi-decidable but not decidable, then the complement language $\overline{L}$ is not semi-decidable. (Corollary 9.4 on the lecture slides.)

S9.2 Prove that the Cartesian product $\mathbb{N} \times \mathbb{N}$ is countably infinite. (*Hint:* Think of the pairs $(m,n) \in \mathbb{N} \times \mathbb{N}$ as embedded in the Euclidean (x,y) plane $\mathbb{R}^2$. Enumerate the pairs by diagonals parallel to the line $y = -x$.) Conclude from this result that also the set $\mathbb{Q}$ of rational numbers is countably infinite.

Solution. A set S is countably infinite, if there exists a bijective mapping $f : \mathbb{N} \to S$. Intuitively speaking, all members of the set S can be assigned an unambiguous running number.

The members $(x,y) \in \mathbb{N} \times \mathbb{N}$ of the set $\mathbb{N} \times \mathbb{N}$ can be assigned a number as shown in the following figure.



The idea is to arrange all pairs of numbers on diagonals parallel to the line $y = -x$ and enumerate the lines by member one at a time, starting from the shortest one. Here the enumeration can not be done parallel to the x-axis; when doing this all indices would be used to enumerate only the x-axis and no pair (x,y) where $y > 0$ would ever be reached.

The enumerating scheme above can be defined as follows:

$$f(x,y) = \sum_{r=0}^{x+y-1} (r+1) + x = \frac{(x+y+1)(x+y)}{2} + x$$

For an example, $f(3,1) = 13$, that is, the running number of pair $(3,1)$ is 13. The function $f(x,y)$ is a bijection; for every running number there exists an unambiguous pair of numbers. Given an index $k \in \mathbb{N}$, one can also compute the corresponding coordinate pair (x,y). It can be found on the line $x + y = r$, where r is the largest number satisfying the equation

$$\frac{(r+1)r}{2} \leq k.$$

The solution for this is $r = \left\lfloor \frac{-1+\sqrt{1+8k}}{2} \right\rfloor$, from which one obtains:

$$\begin{aligned} x &= (k - \frac{(r+1)r}{2}) \\ y &= r - (k - \frac{(r+1)r}{2}) \end{aligned}$$

Every nonnegative rational number $q \in \mathbb{Q}^+$ has a unique canonical, fully reduced representation as $q = \frac{m}{n}$, where $m, n \in \mathbb{N}$ and $n \neq 0$. (Let us agree that the canonical representation of 0 is $\frac{0}{1}$.) Thus, the set $\mathbb{Q}^+$ can be presented as, i.e. can be bijectively mapped to, the corresponding set of pairs $(m, n) \in \mathbb{N} \times \mathbb{N}$, which is a proper subset of the countably infinite set $\mathbb{N} \times \mathbb{N}$. By Problem H9.1, $\mathbb{Q}^+$ is thus either finite or countably infinite. We know that $\mathbb{Q}^+$ is not finite, because it contains the countably infinite set $\{0, 1, 2, \dots\} = \{\frac{0}{1}, \frac{1}{1}, \frac{2}{1}, \dots\}$. Therefore $\mathbb{Q}^+$ is countably infinite. By the same argument, the set $\mathbb{Q}^-$:

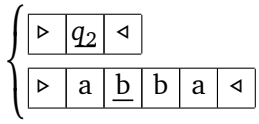
$$\mathbb{Q}^- = \{(-m, n) \mid (m, n) \in \mathbb{Q}^+\}$$

is countably infinite. Thus, the set $\mathbb{Q} = \mathbb{Q}^+ \cup \mathbb{Q}^-$ is the union of two countably infinite sets, and by Problem C9.1 it too is countably infinite.

S9.3 Show that Turing machines that have only *one* internal state in addition to their accepting and rejecting states are capable of recognising exactly the same languages as the standard machines with arbitrarily many states. You may assume that the simulating machines have multiple tapes and may also keep their tape heads stationary (direction code 'S') in a transition.

Solution. Let M be a Turing machine with n states. It is possible to simulate M with a single-state two-tape machine $\widehat{M}$ if we add a symbol q_i to the tape alphabet for each state q_i of M. The first tape of $\widehat{M}$ keeps track of the current state of the machine, while the second tape is used for the actual computation.

For example, the configuration $(q_2, a\underline{b}ba)$ of M would be represented as:



If M has a transition $\delta(q_i, a) = (q_j, b, \Delta)$, then $\widehat{M}$ has a transition $\delta(\hat{q}_0, q_i, a) = (\hat{q}_0, (q_j, S), (b, \Delta))$, where $\hat{q}_0$ is the sole non-halting state of $\widehat{M}$. The read/write head on the first tape stays stationary since otherwise the state information of the machine would be lost.

Formally the construction is as follows: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ be a standard-issue Turing machine where $Q = \{q_0, \dots, q_n\}$. We construct a 2-tape machine $\widehat{M} = (\{\hat{q}_0\}, \Sigma, \Gamma \cup \{q_0, \dots, q_n\}, \hat{\delta}, \hat{q}_0, \hat{q}_{\text{acc}}, \hat{q}_{\text{rej}})$, where

$$\begin{aligned}\hat{\delta} = &\{((\hat{q}_0, q_i, a), (\hat{q}_0, (q_j, S), (b, \Delta)) \mid \delta(q_i, a) = (q_j, b, \Delta), q_j \notin \{q_{\text{acc}}, q_{\text{rej}}\}\} \\ &\cup \{((\hat{q}_0, q_i, a), (\hat{q}, (q, S), (b, \Delta)) \mid \delta(q_i, a) = (q, b, \Delta), q \in \{q_{\text{acc}}, q_{\text{rej}}\}\} \ .\end{aligned}$$