Exercise 1: Constructing Turing Machines

Construct a Turing Machine for each of the following languages.

(a) $L_1 = \{a^i b^j a^i b^j | i, j > 0\}$

(b) Language $L_2$ of all strings over alphabet $\{a, b\}$ with the same number of $a$’s and $b$’s.

Remark: It is sufficient to give a detailed description of the Turing Machines. You do not need to give formal definitions.

Exercise 2: Semi-Decidable vs. Recursively Enumerable

Very often people in computer science use the terms *semi-decidable* and *recursively enumerable* equivalently. The following exercise shows in which way they actually are equivalent. We first recall the definition of both terms.

A language $L$ is *semi-decidable* if there is a Turing machine which accepts every $w \in L$ and does not accept any $w \notin L$ (this means the TM can either reject $w \notin L$ or simply not stop for $w \notin L$).

A language is *recursively enumerable* if there is a Turing machine which eventually outputs every word $w \in L$ and never outputs a word $w \notin L$.

(a) Show that any recursively enumerable language is semi-decidable.

(b) Show that any semi-decidable language is recursively enumerable.

Exercise 3: Decidability

1. The *special halting problem* is defined as

$$H_s = \{\langle M \rangle | \langle M \rangle \text{ encodes a TM and } M \text{ halts on } \langle M \rangle \}.$$ 

Show that $H_s$ is undecidable.

*Hint: Assume that $M$ is a TM which decides $H_s$ and then construct a TM which halts iff $M$ does not halt. Use this construction to find a contradiction.*

2. Show that $A = \{\langle R, S \rangle | R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\}$ is decidable.

3. Show that $EQ_{TM} = \{\langle M_1, M_2 \rangle | M_1, M_2 \text{ are Turing Machines and } L(M_1) = L(M_2)\}$ is undecidable.

*Hint: You may use that $EQ_{TM} = \{\langle M \rangle | M \text{ is a Turing Machine and } L(M) = \emptyset\}$ is undecidable.*