You will need 50% of all homework points to qualify for the exam. (That is, if you get at least 50%, your final grade will be the exam grade. And if you do not get 50%, you do not pass the course.)

You may hand in your solutions in person or by email. If you submit by email, either scan a handwritten solution or typeset your solution readably. I do not consider ASCII formulas readable.

When submitting, indicate your name and your matriculation number. On your first submission, please also indicate a password, this password will be needed for accessing the solutions and your points online.

Problem 1: Addition on a Turing machine

Construct a Turing machine that does the following: Given input $a|b$ (where $a$ and $b$ are unsigned integers in binary encoding and $|$ is a special symbol), it outputs the sum of $a$ and $b$ (in binary encoding).

Explicitly give the set $\Gamma$ of symbols (containing at least $\delta, \square, 0, 1, |$), the set of states $Q$, and the transition function $\delta$.

You do not need to prove that your TM does add correctly, but please add sufficient comments to make the workings understandable.

Notes: Input $a|b$ means that the input tape contains $\delta a | b \square \square \ldots$. You can choose whether your integers are encoded lsb-first or msb-first, but make your choice explicit and stick to it.

Solution. We encode integers lsb-first.

Set of states: $Q = \{q_{\text{start}}, q_{\text{rewind}}, q_{\text{add}}, q_{\text{carry}}, q_{\text{halt}}\}$.
Set of symbols: $\Gamma = \{\delta, \square, 0, 1, |\}$.
Number of tapes: $k = 2$. (Input and output tape.)
Transition functions $\delta$: See Figure 1.

Problem 2: NP-problems

Out of the following five problems, three are in $\text{NP}$. The other two are not (or at least, science cannot currently show that they are).

- Identify those three.
<table>
<thead>
<tr>
<th>$q_{in}$</th>
<th>$\gamma_{1_{in}}$</th>
<th>$\gamma_{2_{in}}$</th>
<th>$q_{out}$</th>
<th>$\gamma_{2_{out}}$</th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_{start}$</td>
<td>$\rhd$</td>
<td>*</td>
<td>$q_{start}$</td>
<td>$\rhd$</td>
<td>$R$</td>
<td>$R$</td>
<td>Move to start of $a$. Also mark the first cell of the output tape with $\rhd$ for later rewinding.</td>
</tr>
<tr>
<td>$q_{start}$</td>
<td>$x \in {0, 1}$</td>
<td>*</td>
<td>$q_{start}$</td>
<td>$x$</td>
<td>$R$</td>
<td>$R$</td>
<td>Copy $a$ onto output tape.</td>
</tr>
<tr>
<td>$q_{rewind}$</td>
<td>*</td>
<td>$x \in {0, 1}$</td>
<td>$q_{rewind}$</td>
<td>$x$</td>
<td>$S$</td>
<td>$L$</td>
<td>After copy, input head is at start of $b$. Output head starts rewinding.</td>
</tr>
<tr>
<td>$q_{rewind}$</td>
<td>*</td>
<td>$\rhd$</td>
<td>$q_{add}$</td>
<td>$\Box$</td>
<td>$S$</td>
<td>$R$</td>
<td>Rewinding done. Removing the marker $\rhd$ from output tape. Input head is now at start of $b$, output head at start of $a$. Adding can begin.</td>
</tr>
<tr>
<td>$q_{add}$</td>
<td>0 or $\Box$</td>
<td>0</td>
<td>$q_{add}$</td>
<td>0</td>
<td>$R$</td>
<td>$R$</td>
<td>Add current bit of $a$ and $b$. Case: both are 0. Thus: write 0 to output. Move to next bit. $\Box$ is interpreted as 0 as long as we are still adding.</td>
</tr>
<tr>
<td>$q_{add}$</td>
<td>0 or $\Box$</td>
<td>1</td>
<td>$q_{add}$</td>
<td>1</td>
<td>$R$</td>
<td>$R$</td>
<td>Like in the step before.</td>
</tr>
<tr>
<td>$q_{add}$</td>
<td>1 or $\Box$</td>
<td>0</td>
<td>$q_{add}$</td>
<td>1</td>
<td>$R$</td>
<td>$R$</td>
<td>Case: $0+1$.</td>
</tr>
<tr>
<td>$q_{add}$</td>
<td>1</td>
<td>1</td>
<td>$q_{carry}$</td>
<td>0</td>
<td>$R$</td>
<td>$R$</td>
<td>Case: $1+0$. Thus, the added bit is 0, and we have to add a carry bit in the next step. Thus we go into state $q_{carry}$.</td>
</tr>
<tr>
<td>$q_{carry}$</td>
<td>0 or $\Box$</td>
<td>0 or $\Box$</td>
<td>$q_{add}$</td>
<td>1</td>
<td>$R$</td>
<td>$R$</td>
<td>Case $0+0$ with carry. Thus output is 1. No carry for next step.</td>
</tr>
<tr>
<td>$q_{carry}$</td>
<td>0 or $\Box$</td>
<td>1</td>
<td>$q_{carry}$</td>
<td>0</td>
<td>$R$</td>
<td>$R$</td>
<td>Case $0+1$ with carry. Output is 0, need carry in next step.</td>
</tr>
<tr>
<td>$q_{carry}$</td>
<td>1</td>
<td>0 or $\Box$</td>
<td>$q_{carry}$</td>
<td>0</td>
<td>$R$</td>
<td>$R$</td>
<td>Case $1+0$ with carry. Output is 0, need carry in next step.</td>
</tr>
<tr>
<td>$q_{carry}$</td>
<td>1</td>
<td>1</td>
<td>$q_{carry}$</td>
<td>1</td>
<td>$R$</td>
<td>$R$</td>
<td>Case $1+1$ with carry. Output is 1, need carry in next step.</td>
</tr>
<tr>
<td>$q_{add}$</td>
<td>$\Box$</td>
<td>$\Box$</td>
<td>$q_{halt}$</td>
<td>$\Box$</td>
<td>$S$</td>
<td>$S$</td>
<td>Both $a$ and $b$ are finished. No carry. Addition complete.</td>
</tr>
</tbody>
</table>

Figure 1: Transition function $\delta$ of the addition Turing machine. $q_{in}, \gamma_{1_{in}}, \gamma_{2_{in}}$ is the input to $\delta$, consisting of state $q_{in}$, symbol $\gamma_{1_{in}}$ read on tape 1, symbol $\gamma_{2_{in}}$ read on tape 2. $q_{out}, \gamma_{2_{out}}, m_1, m_2$ is the output of $\delta$, consisting of the new state $q_{out}$, the symbol $\gamma_{2_{out}}$ written to tape 2 (tape 1 is the read-only input tape), and the movements $m_1, m_2$ of the heads on tapes 1 and 2 ($R$=right, $L$=left, $S$=stay). In the input fields, $*$ stands for ‘does not matter’ (wildcard). Only the part of the value table of $\delta$ that is relevant for the execution is described.
• Show that they are in \textbf{NP}. That is, say what the Turing machine \( M \) from the
definition of the class \textbf{NP} does, say what \( u \) is. You do not need to “program” \( M \), it
is sufficient to say what \( M \) does.\footnote{E.g., like “\( M(x, w) \) outputs 1 iff \(|x| = |w|\)”.
}\footnote{In Arora-Barak, \textbf{SAT} is defined somewhat differently, namely as \textbf{SAT} := \{ \( B \) : \( B \) is a satisfiable Boolean formula in conjunctive normal form (CNF)\}.}

• For the remaining two, explain why you cannot show that they are in \textbf{NP}. (No
formal proof is needed. Just an explanation of the difficulties.)

(a) \textbf{SAT} := \{ \( B \) : \( B \) is a satisfiable Boolean formula\}\footnote{A Boolean formula is a formula
with variables \( x_1, x_2, x_3, \ldots \) and the operations \( \land, \lor, \neg \). A Boolean formula is satisfiable
if the variables \( x_1, x_2, \ldots \) can be assigned values true/false so that the formula
evaluates to true.} A Boolean formula is a formula

\begin{align*}
\exists y_1, \ldots, y_m \in \mathbb{Z}. & p_1(y_1, \ldots, y_m) = \cdots = p_n(y_1, \ldots, y_m) = 0.
\end{align*}

Here \( p_1, \ldots, p_n \) are polynomials in \( m \) variables with integer coefficients. (Both \( n, m \)
can vary.)

(b) \textbf{PALIN} := \{ \( x \) : \( x \) is a palindrome\}.

(c) \textbf{EQ}_1 := \{ (p_1, \ldots, p_n) : \exists x_1, \ldots, x_m \in \mathbb{Z}. p_1(x_1, \ldots, x_m) = \cdots = p_n(x_1, \ldots, x_m) = 0 \}.

Here \( p_1, \ldots, p_n \) are polynomials in \( m \) variables with integer coefficients. (Both \( n, m \)
can vary.)

(d) \textbf{EQ}_2 := \{ (p_1, \ldots, p_n, b_1, \ldots, b_n) : \exists x_1, \ldots, x_m \in \mathbb{Z} \text{ s.t. } p_1(x_1, \ldots, x_m) = \cdots = p_n(x_1, \ldots, x_m) = 0 \land |x_1| \leq b_1, \ldots, |x_m| \leq b_m \}.

Here \( p_1, \ldots, p_n \) are polynomials in \( m \) variables with integer coefficients. (Both \( n, m \)
can vary.)

(e) \textbf{co-SAT} := \{ \( B \) : \( B \) is not a satisfiable Boolean formula\}.

Solution.

• \textbf{SAT} \in \textbf{NP}. For a satisfiable formula \( B \), the certificate \( u \) is an assignment of
variables such that \( B(u) = 1 \). The length of \( u \) is one bit per variable, and each
variable needs to occur in \( B \), hence the length of \( u \) is polynomially-bounded in the
length of \( B \). \( M(B, u) \) computes \( B(u) \).

• \textbf{PALIN} \in \textbf{NP}. Actually, we already know that \textbf{PALIN} \in \textbf{P} \subseteq \textbf{NP}. But we can also
show it directly: \( u \) is the empty word, and \( M(x, u) \) outputs 1 if \( x \) is a palindrome.

• \textbf{EQ}_1 \notin \textbf{NP}. One might first think that it is in \textbf{NP} due to the following argument:
\( u \) is the list of integers that are assigned to \( x_1, \ldots, x_n \). Then \( M((p_1, \ldots, p_n), u) \) just
checks whether \( p_1(x_1, \ldots, x_m) = \cdots = p_n(x_1, \ldots, x_m) = 0 \). However, in general
it could be that a solution \( x_1, \ldots, x_n \) contains extremely large integers. Thus the
length of the certificate \( u \) will not necessarily be polynomially bounded in the length
of \( (p_1, \ldots, p_n) \). But a polynomially-bounded certificate is required for showing that
\textbf{EQ}_1 \notin \textbf{NP}. In fact, it is known that \textbf{EQ}_1 is undecidable (like the Halting Problem).

Equations like in the definition of \textbf{EQ}_1 are called Diophantine equations, solving
them was Hilbert’s tenth problem, and Matiyasevich’s theorem that even whether a
single Diophantine equation has a solution is undecidable.
• **EQ₂ ∈ NP.** Here the certificate \( u \) consists of the solution \( x_1, \ldots, x_m \). And \( M((p_1, \ldots, p_n, b_1, \ldots, b_m), u) \) checks whether \( p_1(x_1, \ldots, x_m) = \cdots = p_n(x_1, \ldots, x_m) = 0 \) and \( |x_1| \leq b_1, \ldots, |x_m| \leq b_m \). Note that in this case, the length of \( u \) is polynomially-bounded because each \( x_i \) is at most as long as the corresponding \( b_i \).

• **co-SAT** is not known to be in **NP.** In fact, it is easy to give a certificate that \( B \) is a satisfiable Boolean formula, but to show that \( B \) is not satisfiable, one needs to find a certificate that shows that for all inputs, \( B \) evaluates to 0. It is not clear how one would do that. It is widely believed that **coSAT \( \not\in \) NP** (but there is no proof of that fact).