Problems Solved:

31	32	33	34	35

Name:

Matrikel-Nr.:

Problem 31. For a Turing machine M let $P(M)$ be the following property: *If M runs at least 1000 steps on a word w, then $w \in L(M)$.* Note that there is no statement about acceptance or non-acceptance if the machine runs less than 1000 steps.
In the following let M be a Turing machine that has the property $P(M)$.

1. Is there a Turing machine E with $P(E)$ such that $\varepsilon \in L(E)$.
2. Is there a Turing machine E with $P(E)$ such that $\varepsilon \notin L(E)$.
3. Is the property of $L(M)$ to contain the empty word, decidable?
4. Is $L(M)$ recursively enumerable?
5. Is the complement $\overline{L(M)}$ recursively enumerable?
6. Is $L(M)$ recursive?
7. Is $L(M)$ necessarily finite?
8. Is $L(M)$ necessarily infinite?

Justify your answers.

Solution of Problem 31:

1. Yes. Namely, a TM that first runs 1000 (dummy) steps, then goes into an accepting state and halts
2. Yes. Namely, a TM with the empty transition function (which accepts nothing).
3. Yes. Simulate (at most) 1000 steps of M on the empty word. If the empty word is not accepted up to this point, return not accepted, otherwise return accepted. That simulator decides the property in question.
4. Yes. $L(M)$ is always recursively enumerable by definition.
5. Yes. For any word w one can run such a simulation of M for 1000 steps and thus decide whether or not $w \in L(M)$.
6. Yes. Previous two answers combined.
7. No. Property $P(M)$ says nothing about the number of steps M really takes on a word. For example, a TM X that accepts every word has property $P(X)$. So $M = X$ is an example that violates finiteness of $L(M) = L(X)$.
8. No. A TM M that does not make a single step, does not accept anything and still has the property $P(M)$.

Problem 32. Which of the following problems are decidable? In each problem below, the input of the problem is the code $\langle M \rangle$ of a Turing machine M with input alphabet $\{0,1\}$.

(a) Does M have at least 4 states?

(b) Is $L(M) \subseteq \{0,1\}^*$?

(c) Is $L(M)$ recursive?

(d) Is $L(M)$ finite?

(e) Is $10101 \in L(M)$?

(f) Is $L(M)$ not recursively enumerable?

(g) Does there exist a word $w \in L(M)$ such that M does not halt on w?

Justify your answer.

Solution of Problem 32:

Question a is "syntactical" and easy to decide. Question b does always have the answer *yes* and is therefore decidable. Question f does always have the answer *no* and is therefore decidable. Question g does always have the answer *no* and is therefore decidable.

Questions c–e are undecidable by the Theorem of Rice. The properties of $L(M)$ are not *trivial*

Problem 33. Let $M_0, M_1, M_2, \dots$ be a list of all Turing machines with alphabet $\Sigma = \{0,1\}$ such that the function $i \mapsto \langle M_i \rangle$ is computable. Let $w_i := 10^i10^i1$ for all natural numbers i. Let $A := \{w_i \mid i \in \mathbb{N} \wedge w_i \in L(M_i)\}$ and $\overline{A} = \Sigma^* \setminus A$.

(a) Is $\overline{A}$ recursively enumerable? (Justify your answer.)

(b) Suppose there is an oracle X_{Delphi} that decides the Halting problem, i. e., you can give to X_{Delphi} the code $\langle M \rangle$ of a a Turing machine M and a word w and X_{Delphi} returns 1 (YES) or 0 (NO) depending on whether or not M halts on w.

Show that one can construct an Oracle-Turing machine T (which is allowed by a special extension to give some word $\langle M \rangle$ (a Turing machine code) and a word w to X_{Delphi} and gets back 1 or 0 depending on whether or not M halts on w) such that $L(T) = \overline{A}$.

(c) Does it follow from (a) and (b) that X_{Delphi} is not a Turing machine? Justify your answer. Note that you are not allowed to use the fact that the Halting problem is undecidable, but you must give a proof that only follows from (a) and (b).

Solution of Problem 33:

Let $W = \{w_i \mid i \in \mathbb{N}\} = \left\{10^i10^i1 \,\middle|\, i \in \mathbb{N}\right\}$, $W' = \Sigma^* \setminus W$.

(a) No. A is recursively enumerable. One can construct a Turing machine U that generates successively all natural numbers i, then computes the code $\langle M_i \rangle$ and the word w_i and simulates M_i on input w_i. If M_i accepts w_i, then U writes w_i to its output tape.

Let $A' = \{w_i \,|\, i \in \mathbb{N} \wedge w_i \notin L(M_i)\}$. Note that W' is recursive, since W is. Since, $\overline{A} = W' \cup A'$ and $W' \cap A' = \emptyset$, it suffices to show that A' is not recursively enumerable.

Indirect proof. Suppose there is a Turing machine V with $L(V) = A'$. Then $V = M_k$ for some natural number k. Consider the word w_k. If $w_k \in A$ then $w_k \in L(M_k) = L(V) = A'$ and, therefore, $w_k \notin A$. If $w_k \notin A$ then $w_k \in A' = L(V) = L(M_k)$. Therefore, $w_k \in A$ by definition of A. So the assumtion that A' is recursively enumerable is false. Therefore also $\overline{A}$ is not recursively enumerable.

(b) T takes a word w and checks if it is in W. If it is not then $w \in \overline{A}$ and T accepts w. If $w \in W$ then $w = w_i$ for some i. T constructs this i and computes the Turing machine code $\langle M_i \rangle$. Then T gives $\langle M_i \rangle$ and w to the oracle X_{Delphi}. If the answer is 0 then T accepts the word w. Otherwise, T simulates M_i on w. Since M_i halts on w, T can read the final state of M_i. If that state is a non-accepting state then T accepts w, otherwise it does not accept w.

(c) Yes. Assume X_{Delphi} is a Turing machine. By (b) we can conclude that $\overline{A}$ is recursively enumerable. However, that contradicts (a). So the assumption must be false.

Problem 34. Show that the Acceptance Problem is reducible to the restricted Halting problem. First explain clearly which Turing machine you have to construct to prove this statement and then give a reasonably detailed description of this construction.

Solution of Problem 34:

Assume the restriced halting problem is decidable. There exists a TM M_H such that M_H accepts input c, iff c is the code of a TM M and M halts on ε.

Then we can define a TM M_A that accepts input (c, w) iff c is the code of a TM M that accepts the word w.

M_A works as follows. If c is not a well-formed Turing machine code, then M_A does not accept its input. Otherwise M_A slightly modifies c to the code c' of a TM M' which halts on ε iff M accepts w.

Thus c' is constructed such that

(a) M' first writes w on the tape,

(b) when M halts and does not accept w, then M' does not halt. In other words, for each non-accepting state and each letter σ where there is no transition in M, it adds a transition to a new state q_{loop} and this state "loops" for every letter to itself.

This modification of c to c' is clearly computable by a Turing machine.

M_A passes c' to M_H. If M_H accepts c' then M_A accepts (c, w). If M_H does not accept c' then M_A does not accepts (c, w).

In fact, this "proof" contains a gap. It might happen that M does not accept w, because M starts on w and first tries to move its head to the left. This situation is not covered by the above construction. But it is easy to remedy this by assuming that the tape is infinite in both directions or by introducing a new tape symbol X that is used as an end marker on the left. Thus, M' would have to write Xw on the tape then position it's head on the beginning of w. Furthermore, c has to be extended in such a way that whenever M' hits X it loops forever. Clearly, also this can be done by a Turing machine.

Problem 35. Let a language $L = L(T) \subseteq \{0,1\}^*$ be given by the code of a Turing machine $\langle T\rangle$. It is known that $\varepsilon \in L$.
Let S_0 be the set of Turing machines of the form $(Q, \{0, 1, X, \sqcup\}, \sqcup, \{0, 1\}, \delta, q_0, \emptyset)$.
Let S_1 be the set of Turing machines of the form $(Q, \{0, 1, X, \sqcup\}, \sqcup, \{0, 1\}, \delta, q_0, Q)$.
Is it decidable whether $L = L(M)$ and $M \in S_0$? That is: Is there a Turing machine D_0 such that it takes a word w as input and returns "yes" if $w = \langle M\rangle$ for a TM $M \in S_0$ with the property $L(M) = L$, and returns "no" otherwise?
What is the answer, if you replace S_0 by S_1? Justify your answers.

Solution of Problem 35:

The answer for S_0 is clearly, "decidable". D_0 simply always say "no". The reason that D_0 doesn't have to look at it's input is that no TM in S_0 accepts anything, i. e., $L(M) = \emptyset$ for every $M \in S_0$. Since $\varepsilon \in L \neq \emptyset = L(M)$, D_0 can always say "no".

For S_1 the answer is not that easy, because, there is no condition that says that $M \in S_1$ always halts. However, the problem can be solved using the Theorem of Rice.

Suppose X is a Turing machine. One can construct from X a TM M with $L(M) = L(X)$ and $M \in S_1$, i.e., there is a computable mapping f such that $f(\langle X\rangle) = \langle M\rangle$ (with M as just described).

The construction of M works as follows: M is identical to X, exept that if M halts in a non-accepting state, the transition function for M is modified such that it leads to a new (accepting) state and M never halts in that state. Then every non-accepting state can be turned into an accepting state (because M never halts in such a state). So M accepts the same langauge as X, but whenever X would halt in an accepting state, M does not halt (i.e., does not accept the word, even though it "loops" indefinitely between accepting states).

We have shown above that the condition $M \in S_1$ is not really a restriction, because if we find a TM X with $L(X) = L = L(T)$, we can also find a TM M with $L(M) = L$ and $M \in S_1$.

Our task is to find out whether $P_1 = \{\langle M\rangle \mid L(M) = L \wedge M \in S_1\}$ is decidable.

We prove that P_1 is undecidable by an indirect proof. Suppose P_1 is decidable. Then the set $P = \{\langle X\rangle \mid L(X) = L\}$ would also be decidable, because one can build a TM D_1 that takes $\langle X\rangle$ as input and simply decides whether or not $f(\langle X \rangle) \in P_1$. However, $P = P_S$ for the nontrivial Problem $S = \{L\}$ (see Lecture notes Def. 43). By the Theorem of Rice, P_S is undecidable and so the assumption must be wrong, i. e., P_1 must also be undecidable.