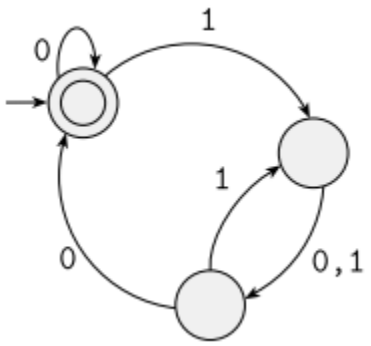
Capitolul 4

Exercitiu:

4.1. Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, 0100\rangle \in A_{DFA}$?
b. Is $\langle M, 011\rangle \in A_{DFA}$?
c. Is $\langle M\rangle \in A_{DFA}$?
d. Is $\langle M, 0100\rangle \in A_{REX}$?
e. Is $\langle M\rangle \in E_{DFA}$?
f. Is $\langle M,M\rangle \in EQ_{DFA}$?

Solutie:

(a) Yes. The DFA M accepts 0100.
(b) No. M doesn't accept 011.
(c) No. This input has only a single component and thus is not of the correct form.
(d) No. The first component is not a regular expression and so the input is not of the correct form.
(e) No. M's language isn't empty.
(f) Yes. M accepts the same language as itself.

Exercitiu:

4.5. Let $E_{TM} = \{\langle M\rangle |\ M \text{ is a TM and } L(M) = \varnothing\}$. Show that $\overline{E_{TM}}$, the complement of E_{TM}, is Turing-recognizable.

Solutie:

Let $s_1, s_2, \ldots$ be a list of all strings in Σ^*. The following TM recognizes $\overline{E_{TM}}$.

"On input $\langle M\rangle$, where M is a TM:

1. Repeat the following for i = 1, 2, 3,... .
2. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
3. If M has accepted any of these, *accept*. Otherwise, continue."

Exercitiu:

4.6. Let X be the set {1, 2, 3, 4, 5} and Y be the set {6, 7, 8, 9, 10}. We describe the functions f : X→Y and g : X→Y in the following tables. Answer each part and give a reason for each negative answer.

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

*a. Is f one-to-one?
b. Is f onto?
c. Is f a correspondence?
*d. Is g one-to-one?
e. Is g onto?
f. Is g a correspondence?

Solutie:

(a) No, f is not one-to-one because f(1) = f(3).
(d) Yes, g is one-to-one.

Exercitiu:

4.10. Let $INFINITE_{DFA}$ = {⟨A⟩| A is a DFA and L(A) is an infinite language}. Show that $INFINITE_{DFA}$ is decidable.

Solutie:

The following TM I decides $INFINITE_{DFA}$.
I = "On input ⟨A⟩, where A is a DFA:

1. Let k be the number of states of A.
2. Construct a DFA D that accepts all strings of length k or more.
3. Construct a DFA M such that L(M) = L(A) ∩ L(D).
4. Test L(M) = ∅ using the EDFA decider T from Theorem 4.4.
5. If T accepts, reject; if T rejects, *accept*."

This algorithm works because a DFA that accepts infinitely many strings must accept arbitrarily long strings. Therefore, this algorithm accepts such DFAs. Conversely, if the algorithm accepts a DFA, the DFA accepts some string of length k or more, where k is the number of states of the DFA. This string may be pumped in the manner of the pumping lemma for regular languages to obtain infinitely many accepted strings.

Exercitiu:

4.12.

Let A = {⟨M⟩| M is a DFA that doesn't accept any string containing an odd number of 1s}. Show that A is decidable.

Solutie:

The following TM decides A.

"On input ⟨M⟩:

1. Construct a DFA O that accepts every string containing an odd number of 1s.
2. Construct a DFA B such that $L(B) = L(M) \cap L(O)$.
3. Test whether $L(B) = \varnothing$ using the EDFA decider T from Theorem 4.4.
4. If T accepts, *accept*; if T rejects, *reject*."

Exercitiu:

4.14. Let $\Sigma = \{0,1\}$. Show that the problem of determining whether a CFG generates some string in 1^* is decidable. In other words, show that

$\{\langle G\rangle |\ G \text{ is a CFG over } \{0,1\} \text{ and } 1^* \cap L(G) \neq \varnothing\}$

is a decidable language.

Solutie:

You showed in Problem 2.18 that if C is a context-free language and R is a regular language, then $C \cap R$ is context free. Therefore, $1^* \cap L(G)$ is context free. The following TM decides the language of this problem.

"On input ⟨G⟩:

1. Construct CFG H such that $L(H) = 1^* \cap L(G)$.
2. Test whether $L(H) = \varnothing$ using the ECFG decider R from Theorem 4.8.
3. If R accepts, *reject*; if R rejects, *accept*."

Exercitiu:

4.23.

Say that an NFA is ***ambiguous*** if it accepts some string along two different computation branches. Let $AMBIG_{NFA}$ = {⟨N⟩| N is an ambiguous NFA}. Show that $AMBIG_{NFA}$ is decidable. (Suggestion: One elegant way to solve this problem is to construct a suitable DFA and then run E_{DFA} on it.)

Solutie:

The following procedure decides $AMBIG_{NFA}$. Given an NFA N, we design a DFA D that simulates N and accepts a string iff it is accepted by N along two different computational branches. Then we use a decider for E_{DFA} to determine whether D accepts any strings.

Our strategy for constructing D is similar to the NFA-to-DFA conversion in the proof of Theorem 1.39. We simulate N by keeping a pebble on each active state. We begin by putting a red pebble on the start state and on each state reachable from the start state along ε transitions. We move, add, and remove pebbles in accordance with N's transitions, preserving the color of the pebbles. Whenever two or more pebbles are moved to the same state, we replace its pebbles with a blue pebble. After reading the input, we accept if a blue pebble is on an accept state of N or if two different accept states of N have red pebbles on them.

The DFA D has a state corresponding to each possible position of pebbles. For each state of N, three possibilities occur: It can contain a red pebble, a blue pebble, or no pebble. Thus, if N has n states, D will have 3n states. Its start state, accept states, and transition function are defined to carry out the simulation.

Exercitiu:

4.25. Let BAL_{DFA} = {⟨M⟩| M is a DFA that accepts some string containing an equal number of 0s and 1s}. Show that BAL_{DFA} is decidable. (Hint: Theorems about CFLs are helpful here.)

Solutie:

The language of all strings with an equal number of 0s and 1s is a context-free language, generated by the grammar S → 1S0S | 0S1S | ε. Let P be the PDA that recognizes this language. Build a TM M for BAL_{DFA}, which operates as follows. On input ⟨B⟩, where B is a DFA, use B and P to construct a new PDA R that recognizes the intersection of the languages of B and P. Then test whether R's language is empty. If its language is empty, *reject*; otherwise, *accept*.

Capitolul 5

Exercitiu:

5.5. Show that A_{TM} is not mapping reducible to E_{TM}. In other words, show that no computable function reduces A_{TM} to E_{TM}. (Hint: Use a proof by contradiction, and facts you already know about A_{TM} and E_{TM}.)

Solutie:

Suppose for a contradiction that $ATM \leq_m E_{TM}$ via reduction f. It follows from the definition of mapping reducibility that $\overline{A_{TM}} \leq_m \overline{E_{TM}}$ via the same reduction function f. However, $\overline{E_{TM}}$ is Turing-recognizable (see the solution to Exercise 4.5) and ATM is not Turing-recognizable, contradicting Theorem 5.28.

Exercitiu:

5.6. Show that $\leq_m$ is a transitive relation.

Solutie:

Suppose $A \leq_m B$ and $B \leq_m C$. Then there are computable functions f and g such that $x \in A \Leftrightarrow f(x) \in B$ and $y \in B \Leftrightarrow g(y) \in C$. Consider the composition function $h(x) = g(f(x))$.

We can build a TM that computes h as follows: First, simulate a TM for f (such a TM exists because we assumed that f is computable) on input x and call the output y.

Then simulate a TM for g on y. The output is $h(x) = g(f(x))$.

Therefore, h is a computable function. Moreover, $x \in A \Leftrightarrow h(x) \in C$. Hence $A \leq_m C$ via the reduction function h.

Exercitiu:

5.7. Show that if A is Turing-recognizable and $A \leq_m \overline{A}$, then A is decidable

Solutie:

Suppose that $A \leq_m \overline{A}$. Then $\overline{A} \leq_m A$ via the same mapping reduction. Because A is Turing-recognizable, Theorem 5.28 implies that A is Turing-recognizable, and then Theorem 4.22 implies that A is decidable.

Exercitiu:

5.8. In the proof of Theorem 5.15, we modified the Turing machine M so that it never tries to move its head off the left-hand end of the tape. Suppose that we did not make this modification to M. Modify the PCP construction to handle this case.

Solutie:

You need to handle the case where the head is at the leftmost tape cell and attempts to move left. To do so, add dominos

$$\left[\frac{\#qa}{\#rb}\right]$$

for every q, r ∈ Q and a, b ∈ Γ, where δ(q, a)=(r, b,L). Additionally, replace the first domino with

$$\left[\frac{\#}{\#\#q_0 w_1 w_2 \cdots w_n}\right]$$

to handle the case where the head attempts to move left in the very first move.

Exercitiu:

5.10. Consider the problem of determining whether a two-tape Turing machine ever writes a nonblank symbol on its second tape when it is run on input *w*. Formulate this problem as a language and show that it is undecidable.

Solutie:

Let B = {⟨M,w⟩| M is a two-tape TM that writes a nonblank symbol on its second tape when it is run on w}. Show that A_{TM} reduces to B. Assume for the sake of contradiction that TM R decides B. Then construct a TM S that uses R to decide A_{TM}.

S = “On input ⟨M,w⟩:

1. Use M to construct the following two-tape TM T . T = “On input x:
    1. Simulate M on x using the first tape.
    2. If the simulation shows that M accepts, write a nonblank symbol on the second tape.”
2. Run R on ⟨T,w⟩ to determine whether T on input w writes a nonblank symbol on its second tape.
3. If R accepts, M accepts w, so *accept*. Otherwise, *reject*.”

Exercitiu:

5.11. Consider the problem of determining whether a two-tape Turing machine ever writes a nonblank symbol on its second tape during the course of its computation on any input string. Formulate this problem as a language and show that it is undecidable.

Solutie:

Let C = {⟨M⟩| M is a two-tape TM that writes a nonblank symbol on its second tape when it is run on some input}. Show that A_{TM} reduces to C. Assume for the sake of contradiction that TM R decides C. Construct a TM S that uses R to decide A_{TM}.

S = "On input ⟨M,w⟩:

1. Use M and *w* to construct the following two-tape TM T_w. T_w = "On any input:
    1. Simulate M on w using the first tape.
    2. If the simulation shows that M accepts, write a nonblank symbol on the second tape."
2. Run R on $\langle T_w \rangle$ to determine whether Tw ever writes a nonblank symbol on its second tape.
3. If R accepts, M accepts *w*, so *accept*. Otherwise, *reject*."

Exercitiu:

5.28. **Rice's theorem**. Let P be any nontrivial property of the language of a Turing machine. Prove that the problem of determining whether a given Turing machine's language has property P is undecidable.

In more formal terms, let P be a language consisting of Turing machine descriptions where P fulfills two conditions. First, P is nontrivial—it contains some, but not all, TM descriptions. Second, P is a property of the TM's language—whenever L(M1) = L(M2), we have ⟨M1⟩ ∈ P iff ⟨M2⟩ ∈ P. Here, M_1 and M_2 are any TMs. Prove that P is an undecidable language.

Solutie:

Assume for the sake of contradiction that P is a decidable language satisfying the properties and let R_P be a TM that decides P. We show how to decide ATM using R_P by constructing TM S. First, let T∅ be a TM that always rejects, so $L(T_\varnothing) = \varnothing$. You may assume that $\langle T_\varnothing \rangle \notin$ P without loss of generality because you could proceed with P instead of P if ⟨T∅⟩ ∈ P. Because P is not trivial, there exists a TM T with ⟨T ⟩ ∈ P. Design S to decide ATM using R_P 's ability to distinguish between $T_\varnothing$ and T .

S = "On input ⟨M,*w*⟩:

1. Use M and w to construct the following TM M_w. M_w = "On input x:
    1. Simulate M on *w*. If it halts and rejects, *reject*. If it accepts, proceed to stage 2.
    2. Simulate T on x. If it accepts, *accept*."
2. Use TM R_P to determine whether $\langle M_w \rangle \in$ P. If YES, *accept*. If NO, *reject*."

Exercitiu:

5.30. Use Rice's theorem, which appears in Problem 5.28, to prove the undecidability of each of the following languages.

*a. $INFINITE_{TM}$ = {⟨M⟩| M is a TM and L(M) is an infinite language}.

b. {⟨M⟩| M is a TM and 1011 ∈ L(M)}.

c. ALL_{TM} = {⟨M⟩| M is a TM and L(M) = Σ^*}.

Solutie:

(a) $INFINITE_{TM}$ is a language of TM descriptions. It satisfies the two conditions of Rice's theorem. First, it is nontrivial because some TMs have infinite languages and others do not. Second, it depends only on the language. If two TMs recognize the same language, either both have descriptions in $INFINITE_{TM}$ or neither do. Consequently, Rice's theorem implies that $INFINITE_{TM}$ is undecidable.

Capitolul 7

Exercitiu:

7.1. Answer each part TRUE or FALSE.

a. $2n = O(n)$.

b. $n^2 = O(n)$.

*c. $n^2 = O(n \log^2 n)$.

*d. $n \log n = O(n^2)$.

e. $3^n = 2^{O(n)}$.

f. $2^{2^{\wedge}n} = O(2^{2^{\wedge}n})$.

Solutie:

(c) FALSE; (d) TRUE.

Exercitiu:

7.2.Answer each part TRUE or FALSE.

a. $n = o(2n)$.

b. $2n = o(n^2)$.

*c. $2^n = o(3^n)$.

*d. $1 = o(n)$.

e. $n = o(\log n)$.

f. $1 = o(1/n)$.

Solutie:

(c) TRUE; (d) TRUE

Exercitiu:

7.16. Show that NP is closed under the star operation.

Solutie:

Let $A \in$ NP. Construct N_{TM} M to decide A^* in nondeterministic polynomial time.

M = “On input w:

1. Nondeterministically divide *w* into pieces $w = x_1 x_2 \cdots x_k$.
2. For each x_i, nondeterministically guess the certificates that show $x_i \in A$.
3. Verify all certificates if possible, then *accept*. Otherwise, if verification fails, *reject*.”

Exercitiu:

7.23. Let *HALF-CLIQUE* = {⟨G⟩| G is an undirected graph having a complete subgraph with at least m/2 nodes, where m is the number of nodes in G}. Show that HALF-CLIQUE is NP-complete.

Solutie:

We give a polynomial time mapping reduction from *CLIQUE* to *HALF-CLIQUE*.

The input to the reduction is a pair ⟨G, k⟩ and the reduction produces the graph ⟨H⟩ as output where H is as follows. If G has m nodes and k = m/2, then H = G. If k < m/2, then H is the graph obtained from G by adding j nodes, each connected to every one of the original nodes and to each other, where j = m − 2k.

Thus, H has m + j = 2m − 2k nodes. Observe that G has a k-clique iff H has a clique of size k + j = m − k, and so ⟨G, k⟩ ∈ *CLIQUE* iff ⟨H⟩ ∈ *HALF-CLIQUE*. If k > m/2, then H is the graph obtained by adding j nodes to G without any additional edges, where j = 2k − m.

Thus, H has m + j = 2k nodes, and so G has a k-clique iff H has a clique of size k.

Therefore, ⟨G, k⟩ ∈ *CLIQUE* iff ⟨H⟩ ∈ *HALF-CLIQUE*. We also need to show *HALF-CLIQUE* ∈ NP. The certificate is simply the clique.

Exercitiu:

7.33. In the following solitaire game, you are given an m × m board. On each of its m^2 positions lies either a blue stone, a red stone, or nothing at all. You play by removing stones from the board until each column contains only stones of a single color and each row contains at least one stone. You win if you achieve this objective. Winning may or may not be possible, depending upon the initial configuration. Let *SOLITAIRE* = {⟨G⟩| G is a winnable game configuration}. Prove that *SOLITAIRE* is NP-complete.

Solutie:

First, *SOLITAIRE* ∈ NP because we can verify that a solution works, in polynomial time. Second, we show that *3SAT* $\leq_P$ *SOLITAIRE*. Given φ with m variables x1,...,xm and k clauses c1,...,ck, construct the following k × m game G. We assume that φ has no clauses that contain both xi and xi because such clauses may be removed without affecting satisfiability.

If xi is in clause cj , put a blue stone in row cj , column xi. If $\overline{xi}$ is in clause cj , put a red stone in row cj , column xi. We can make the board square by repeating a row or adding a blank column as necessary without affecting solvability. We show that φ is satisfiable iff G has a solution.

(→) Take a satisfying assignment. If xi is true (false), remove the red (blue) stones from the corresponding column. So stones corresponding to true literals remain. Because every clause has a true literal, every row has a stone.

(←) Take a game solution. If the red (blue) stones were removed from a column, set the corresponding variable true (false). Every row has a stone remaining, so every clause has a true literal. Therefore, φ is satisfied.

Exercitiu:

7.40. Show that if P = NP, a polynomial time algorithm exists that takes an undirected graph as input and finds a largest clique contained in that graph. (See the note in Problem 7.38.)

Solutie:

If you assume that P = NP, then *CLIQUE* ∈ P, and you can test whether G contains a clique of size k in polynomial time, for any value of k. By testing whether G contains a clique of each size, from 1 to the number of nodes in G, you can determine the size t of a maximum clique in G in polynomial time.

Once you know t, you can find a clique with t nodes as follows. For each node x of G, remove x and calculate the resulting maximum clique size. If the resulting size decreases, replace x and continue with the next node.

If the resulting size is still t, keep x permanently removed and continue with the next node. When you have considered all nodes in this way, the remaining nodes are a t-clique.

Teoreme:

4.4. E_{DFA} is a decidable language.
4.8. E_{CFG} is a decidable language.
1.39. Every nondeterministic finite automaton has an equivalent deterministic finite automaton.
5.28. If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.
4.22. A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.
In other words, a language is decidable exactly when both it and its complement are Turing-recognizable.
5.15. PCP is undecidable.