Final Exam, Theory of Computation 2023

- Books, notes, communication, calculators, cell phones, computers, etc... are not allowed.
- Your explanations and proofs should be clear enough and in sufficient detail so that they are easy to understand and have no ambiguities.
- You are allowed to refer to material covered in lectures (but not exercises) including theorems without reproving them.
- **Do not touch until the start of the exam.**

Good luck!

Name: _____ **N° Sciper:** _____

Problem 1 / 15 points	Problem 2 / 15 points	Problem 3 / 20 points	Problem 4 / 20 points	Problem 5 / 15 points	Problem 6 / 15 points

Total / 100

1 *(15 pts)* **Quick-fire round.** Consider the following statements.

1. Every regular language is in **P**.
2. If $A \subseteq B \subseteq C$ and both A and C are regular, then B is regular.
3. If A and B are regular, then $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$ is regular.
4. If $A \leq_m \text{HALT}$ and $\overline{A} \leq_m \text{HALT}$, then A is decidable.
5. If A is recognisable, then $\overline{A}$ is recognisable.
6. If $\text{HALT} \leq_p A$, then $A \notin \mathbf{NP}$.
7. If $A \in \mathbf{NP}$ and $B \leq_p A$, then $B \in \mathbf{NP}$.
8. Graph-Isomorphism is **NP**-complete.
9. $\text{SAT} \notin \mathbf{coNP}$.
10. Every CNF formula of size n can be equivalently written as DNF formula of size $O(n^2)$.
11. The language $\{(ab)^n : n \text{ is divisible by } 2023\}$ is regular.
12. The language $\{\langle M \rangle : M \text{ is a TM that halts on all inputs}\}$ is recognisable.
13. The language $\{\langle \varphi \rangle : \varphi \text{ is a CNF such that } \varphi(x) = \varphi(x') \text{ for all inputs } x, x'\}$ is in **NP**.
14. The language $\{\langle p \rangle : p \text{ is an } n\text{-variate polynomial s.t. } p(x) = 0 \text{ for some } x \in \mathbb{Z}^n\}$ is decidable.
15. The language $\{\langle D, 1^n \rangle : D \text{ is a DFA that rejects some } n\text{-bit string } x \in \{0,1\}^n\}$ is in **P**. (*Compare this to Problems 3 and 5 below!* ☺)

For each box below, write one of the following symbols:

- **T** if the statement is known to be true.
- **F** if the statement is false *or not known to be true.* E.g., both $\mathbf{P} = \mathbf{NP}$ and $\mathbf{P} \neq \mathbf{NP}$ should be marked **F**.
- or leave the box empty.

A correct **T**/**F** answer is worth +1 point, an incorrect answer is worth −1 point, and an empty answer is worth 0 points.

Solution:

1.	T
2.	F
3.	T
4.	T
5.	F

6.	T
7.	T
8.	F
9.	F
10.	F

11.	T
12.	F
13.	F
14.	F
15.	T

2 *(15 pts)* **Regular languages.** Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Show that $L(D)$ is infinite if and only if D accepts some string $x \in \Sigma^*$ whose length $|x|$ satisfies $|Q| \leq |x| \leq 2|Q|$.

Solution:

Let us first suppose D accepts some string $x \in \sum^*$ whose length satisfies $|Q| \leq |x| \leq 2|Q|$. Let $m \coloneqq |x|$ denote the length of x, and $a_0, a_1, \ldots, a_m$ denote the states visited when running D with input x. Since $m \geq |Q|$, by the pigeonhole principle, there is some state which has been visited at least twice, i.e. there exists $0 \leq i < j \leq m$ s.t. $a_i = a_j$. Let $x = pqr$ be a decomposition of x where $|p| = i, |q| = j - i, |r| = m - j$. Then if we run from the state a_i with input y, we will get back to a_i. Thus for any $k \geq 0$, $pq^k r \in L(D)$, which implies $L(D)$ is infinite. (The above is essentially the proof of pumping lemma.)

Now let us prove the other direction. Suppose that $L(D)$ is infinite. Then there is some $x \in L(D)$ of length $|x| \geq |Q|$. If $|x| \leq 2|Q|$, we are done. Otherwise, using the same argument as above, there is a decomposition of $x = pqr$, such that $|pq| \leq |Q|$ and $pr \in L(D)$. We then replace x with pr and repeat the above process until we get some x^* of length at most $2|Q|$. We claim that x^* must have length at least $|Q|$. This is because if $|x^*| < |Q|$, then we remove at least $|Q| + 1$ letters in the last iteration, a contradiction. Therefore, there exists $x \in L(D)$ s.t. $|Q| \leq |x| \leq 2|Q|$.

Common mistakes and grading scheme:

- 7 points for the if direction, 8 points for the only if direction
- **(if)** -2 points for not showing why there is a cycle.
- **(if)** -5/6 points for incorrect or very incomplete solutions
- **(only if)** -1/2 points for small bugs (uncommon)
- **(only if)** -4/5 points for using the pumping lemma in a wrong way. In fact, one can only apply the pumping lemma to a string of length at least $|Q|$. Thus the right way to find the desired string is to remove cycles instead of adding ones.
- **(only if)** -6/7 points for answers even farther from the right solution

3 *(20 pts)* **NFAs and NP-completeness.** Show that the following language is **NP**-complete:

$$R_{\text{NFA}} \;=\; \big\{\langle N, 1^n\rangle : N \text{ is an NFA that rejects some } n\text{-bit string } x \in \{0,1\}^n\big\}.$$

(Hint: Reduce from SAT*: given a CNF formula φ over n variables, show how to construct a polynomial-size NFA N such that $\varphi(x) = 1$ iff N rejects x.)*

Solution:

NP membership. The certificate for our verifier is the string $x \in \{0,1\}^n$ that is potentially rejected. The verifier checks whether x is indeed rejected. For a DFA this task is trivial, we simply traverse it from the starting state according to the symbols of x and check whether we reached the final state. For an NFA this is less trivial, we need to determine whether any accepting state of the NFA is reachable from the starting state if the transitions are made according to x. Let $\textsc{Cl}(S)$ for a set of states S be defined as the set of states reachable from S by ϵ-transitions. Observe that $\textsc{Cl}$ can be computed in linear time in the size of the NFA with depth-first-search. Let S_i for $i \in \{0, 1, \ldots, n\}$ be the set of states reachable from the starting state of the NFA if the transitions are made according to $x_1 \ldots x_i$. We have $S_0 = \textsc{Cl}(\{q_{\text{start}}\})$. Moreover, S_i is computed from S_{i-1} by $S_i \coloneqq \textsc{Cl}(\bigcup_{q \in S_{i-1}} \delta(q, x_i))$. All of this can be done in polynomial time. Then S_n contains accepting states iff the NFA accepts x.

Remark: Using a similar idea, one can check whether a DFA rejects any string in $\{0,1\}^n$ in polynomial time (keep track of a set $S_i \subseteq Q$ consisting of the states that are reachable from q_0 by some string of length i). Thus $R_{\text{DFA}} \in \mathbf{P}$, which shows that item 15 in Problem 1 is true. This implies, in particular, that no reduction from SAT to R_{NFA} that constructs a DFA can be correct (unless $\mathbf{P} = \mathbf{NP}$).

NP hardness. Let $\phi = \bigwedge_{i \in [m]} C_i$ (where $[m] = \{1, \ldots, m\}$) be a 3-CNF with clauses $C_1, \ldots, C_m$. We are going to design an NFA that recognizes the language of assignments $x \in \{0,1\}^n$ that *falsify* ϕ. First, we construct DFAs $N_1, \ldots, N_m$ such that N_i recognizes the language $\{x \in \{0,1\} \mid C_i(x) = 0\}$. Each of those DFAs is going to have $n+2$ states. Then we construct an NFA N that recognizes the union of the languages recognized by N_i, which is exactly $\{x \in \{0,1\}^n \mid \phi(x) = 0\}$. Then N rejects some string x if and only if ϕ is satisfiable. Now we need to check two things: (a) we can indeed construct N_i (b) we can construct a *polynomial-size* NFA recognising the union of the languages. Notice that step (b) is where this solution fails for DFAs.

Part (a). Let $C_i = \ell_1 \vee \ell_2 \vee \ell_3$, where ℓ_j is a literal (a variable or negation of a variable). Let the set of states of N_i be $[n+1] \cup \{\bot\}$ and let for $j \in [n]$, $\sigma \in \{0,1\}$ set $\delta(j, \sigma) = j + 1$ if $x_j, \neg x_j \notin \{\ell_1, \ell_2, \ell_3\}$ or if the value σ *falsifies* the literal of the variable x_j appearing in C_i. Otherwise let $\delta(j, \sigma) = \bot$. Let 1 be the starting state and $n+1$ be the only accepting state. Clearly a word $x_1, \ldots, x_j$ reaches the state $j+1$ in N_i if and only if none of the values of $x_1, \ldots, x_j$ falsify the corresponding literals of C_i.

Part (b). Let us unite the sets of states of $N_1, \ldots, N_m$, and add an additional starting state q_0, that we connect to the starting state of each of $N_1, \ldots, N_m$ with an ϵ-transition. The set of accepting states is the union of these sets for $N_1, \ldots, N_m$.

Grading scheme: Up to 8 points for the **NP** membership and up to 12 points for reducing SAT to R_{NFA}. Common mistakes and penalties:

- **(Membership)** Not describing how one checks if an NFA accepts or rejects a string: -4 points.

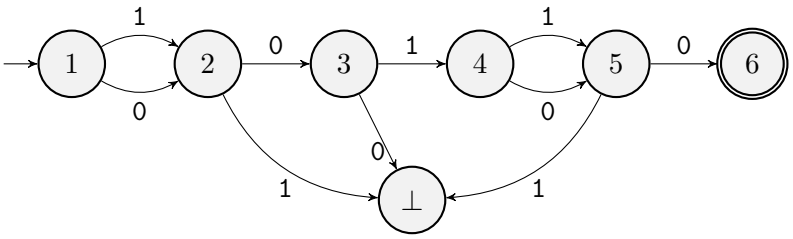
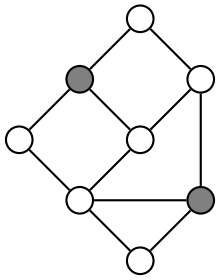


Figure 1. An NFA corresponding to a clause $x_2 \vee \neg x_3 \vee x_5$.

- **(Membership)** Giving an exponential-time verifier -3 to -5 points, depending on the reflection on the issue above.
- **(Hardness)** Incorrect DFAs for clauses (e.g. assuming that a clause mentions all the variables of the CNF), but correct and well-explained construction for the part (b): -6 points.
- **(Hardness)** Having an exponential blow-up in the reduction: -10 points. If ideas about part (b) are explained, the fine is reduced by 2-3 points.
- **(Hardness)** Severe flaws (-9 to -12 points):
  - the reduction is in the wrong direction: -12;
  - the reduction maps a pair (ϕ, x) to an NFA: -11 (notice that this is a reduction from the problem of checking whether the given assignment satisfies the given CNF, not from checking whether such an assignment *exists*).
  - the reduction implies $\mathbf{P} = \mathbf{NP}$: -10.
- **(Hardness)** Correct reduction that was not clearly explained -1 to -4 points depending on the severity. If what is left vague may lead to one of the issues above, the corresponding fines above apply.

4 *(20 pts)* **NP-completeness.** Let $G = (V, E)$ be an undirected graph. We say that $S \subseteq V$ is a *blocking set* if it contains at least one node from each cycle in G. In other words, if we remove the nodes S (and all edges adjacent to S) from G, then we are left with a *forest* (graph with no cycles). For example, the highlighted nodes below form a blocking set of size 2.



Show that the following problem is **NP**-complete:

$$\textsc{Block} = \{\langle G, k\rangle : G \text{ is a graph that contains a blocking set of of size } k\}.$$

(You may assume the **NP***-completeness of any of the problems discussed in the lectures:* SAT, Independent-Set, Clique, Vertex-Cover, Set-Cover, Subset-Sum*, etc. Make sure to prove that your reduction is correct!)*

Solution: For **NP**-membership, let us describe a polynomial-time verifier. As a certificate, it takes a blocking set of size k. Verifier deletes all the vertices from blocking set with their adjacent edges from a graph G, and then checks that G does not contain any cycles (for example, with DFS or BFS).

For **NP**-hardness, let us describe a reduction from Vertex-Cover to Block. Given an instance of Vertex-Cover, $\langle G, k\rangle$, we map it to an instance of Block, $\langle G', k\rangle$.

G' is constructed from G as follows. For every edge $e = (v_1, v_2) \in G$ we have 3 vertices in G': v_1, v_2, v_e, and we connect each two of them with an edge such that they form a cycle of length 3.

Consider a vertex cover of size k in G. Its vertices exactly correspond to a blocking set in G'.

Let us note that it is enough for a blocking set to hit at least one vertex in every simple cycle, and from that it follows that it contains at least one node in every cycle.

As vertex cover in G contains as least one endpoint of each edge, in graph G' this set contains at least one vertex from each cycle of length 3 that we introduced in our reduction. The same is true for the rest of the cycles of the graph G', as vertex cover in G is also a blocking set in G.

For the other direction, consider a blocking set of size k in G'. Without loss of generality, we can assume that it does not contain any v_e for e an edge in G. If it does, simply swap this vertex for any of the two vertices v_1, v_2 such that $(v_1, v_2) = e \in G$.

After that, the blocking set in G' exactly corresponds to a vertex cover in G, as for every edge in contains at least one its endpoint.

Grading scheme: 5 points for proof of **NP**-membership and 15 for **NP**-hardness. Partial solution are graded as follows.

NP-membership:

- 4 points if verifier has minor inaccuracies;
- 3 points if verifier is not polynomial (e.g., "find all cycles in the graph"), but still works;
- $0 - 2$ for more serious bugs.

NP-hardness:

- 10 points for correct reductions that have moderate bugs in the proof;
- $3-5$ for reductions that do not work, but are in the right direction and show some level of understanding of what it means to construct a reduction and to prove its correctness;
- $0-2$ for more severe mistakes.

5 *(15 pts)* **Undecidability.** Consider the language

$$R_{\text{TM}} = \{\langle M, 1^n\rangle : M \text{ is a TM that rejects some } n\text{-bit string } x \in \{0,1\}^n\}.$$

Classify R_{TM} as one of (i) decidable, (ii) undecidable but recognisable, (iii) unrecognisable. Justify your answer with a proof.

Solution: The correct answer is (ii), R_{TM} is undecidable but recognizable.

Part 1. We will prove that R_{TM} is recognizable by providing the following recognizer M_r:

$M_r(\langle M, 1^n\rangle)$:
1. For every $l \in \mathbb{N}$ and every $x \in \{0,1\}^n$:
   (a) Simulate $M(x)$ for l steps.
   (b) If M rejects within l steps, accept.

For every TM M and $n \in \mathbb{N}$, if $\langle M, 1^n\rangle \in R_{\text{TM}}$, then M must reject some input $x \in \{0,1\}^n$ after a finite amount of steps k. The recognizer M_r is simulating the TM M on all of the 2^n inputs of interest, for an increasing amount of steps, in parallel. When M_r will simulate M for k steps, then M will reject and M_r will accept the input $\langle M, 1^n\rangle$. Apart from that, if it accepts some input $\langle M, 1^n\rangle$, it means that it found M rejecting some $x \in \{0,1\}^n$, therefore it holds that $\langle M, 1^n\rangle \in R_{\text{TM}}$.

Part 2. We will prove the undecidability of R_{TM} by proving that $\text{HALT}_{\text{TM}} \leq_m R_{\text{TM}}$. We construct f as follows:

$f(\langle M, w\rangle)$:
1. Simulate $M(w)$.
2. If M halts on w then reject.

We now have to show that $\langle M, w\rangle \in A_{\text{TM}} \Leftrightarrow f(\langle M, w\rangle) \in R_{\text{TM}}$

($\Rightarrow$) If $\langle M, w\rangle \in \text{HALT}_{\text{TM}}$, this means that M halts on w. Therefore, $f(\langle M, w\rangle)$ is a TM that rejects all inputs. As a result, $f(\langle M, w\rangle) \in R_{\text{TM}}$.

($\Leftarrow$) If $T = f(\langle M, w\rangle) \in R_{\text{TM}}$, then there exists some $x \in \{0,1\}^n$ for which T rejects x. However, T is a TM that ignores the input and simulates $M(w)$. This means that if T rejected some input, then M must have halted w. Therefore $\langle M, w\rangle \in \text{HALT}_{\text{TM}}$.

Grading Scheme: Choosing the correct answer takes 3 points and each of the two necessary proofs are awarded 6 points each. Each of the proofs is graded in the following way:

- 6 points (out of 6) for a complete and formal proof
- 5 points (out of 6) for a formal proof that has minor inaccuracies
- 4 points (out of 6) for a proof attempt that has wrong details
- 2 points (out of 6) for informal/hand-waving attempts or major mistakes
- 0 points (out of 6) for fundamentally flawed hand-waving attempts

In the case of a wrong answer, the attempts are graded in the following way:

- 4 points (out of 15) for an almost technically sound proof that is wrong due to small inaccuracies or correct proofs whose implications are interpreted incorrectly.
- 2 points (out of 15) for formal proof attempts that have wrong details or wrong assumptions (e.g. using wrong definitions of languages).
- 0 points (out of 15) for no justification at all or largely flawed hand-waving arguments

6 *(15 pts)* **Busy Beaver.** Define the n-th *Busy Beaver* number, denoted $\mathrm{BB}(n)$, as the largest number $k \in \mathbb{N}$ such that there exists a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{halt}})$ such that

(1) M has $|Q| = n + 1$ states (that is, n states in addition to its halting state q_{halt}),
(2) M has a binary input/tape alphabet, $\Sigma = \Gamma = \{0, 1\}$,
(3) M, on the empty input ε, halts with 1^k written on its tape.

In other words, amongst all $(n+1)$-state TMs that halt on the empty input, what is the longest all-1 string that it can output? Note that $\mathrm{BB}(n)$ is always finite, as there are only finitely many distinct TMs satisfying (1)–(3) for any given n.

Show that the function $\mathrm{BB}\colon \mathbb{N} \to \mathbb{N}$ is not computable. That is, show that there is no TM that on input n will always halt with $\mathrm{BB}(n)$ on its tape.

(*Hint: If* BB *were computable, how would you decide the Halting problem?*)

Solution: Assume for contradiction that BB is computable and let M be a TM that on input n computes $\mathrm{BB}(n)$. Consider a pair $\langle T, w\rangle$ for which we want to decide whether the TM T halts on input w. Construct a TM N that simulates T on input w step by step while writing a 1 on the tape during every such step. Let $n = |Q(N)|$ be the number of states of the TM N. Note that by the definition of BB, N cannot write more than $\mathrm{BB}(n-1)$ 1's on the tape under the condition that it halts. Moreover, N halts if and only if T halts. Hence, $\mathrm{BB}(n-1)$ is also an upper bound on the number of steps T can make under the condition that it halts. We can decide the Halting problem for $\langle T, w\rangle$ in the following way. Run $M(n)$ to determine the upper bound $\mathrm{BB}(n-1)$. Run $T(w)$ for $\mathrm{BB}(n-1)$ steps. If $T(w)$ halts within $BB(n-1)$ steps, answer 'yes' otherwise answer 'no'. By the arguments above, $T(w)$ halts if and only if it halts within $BB(n-1)$ steps.

Grading Scheme:

- 15 points for solutions that build N, obtain the upper bound on the number of steps of $T(w)$ and check accordingly.
- 8 points if the student simply claims that $\mathrm{BB}(n)$ is an upper bound on the number of steps a TM T with $|Q| = n + 1$ states can make on the empty input.
- 7 points if the student simply claims that $\mathrm{BB}(n)$ is an upper bound on the number of steps a TM T with $|Q| = n + 1$ states can make on input w.
- 3 points if the student simply runs the TM T and compares the number of 1's it outputs against some upper bound obtained through BB.
- -2 points if mistakes were made when calculating $\mathrm{BB}(n)$. In particular, if the student calculated $\mathrm{BB}(|w|)$ where $|w|$ does not correspond to the number of states of a TM.
- -2 points for incorrect statements such as claiming a TM that halts within k steps on the empty input halts within $|w| * k$ steps on input w.
- -2 points if the solution was missing important arguments or contained flawed constructions.
- For solutions that tried to give a mapping reduction: 2 points if the student defined a language to reduce to, even if they did not prove that the language is decidable. 2 more points if the reduction is correct or 1 more point if the reduction builds on reasonable ideas.