Homework I, Theory of Computation 2025

Submission: **The deadline for Homework 1 is 23:59 on 20 March.** Please submit your solutions on Moodle. Typing your solutions using a typesetting system such as LaTeX is strongly encouraged! If you must handwrite your solutions, write cleanly and with a pen. Messy and unreadable homeworks will not be graded. No late homeworks will be accepted.

Writing: Please be precise, concise and (reasonably) formal. Keep in mind that many of the problems ask you to provide a proof of a statement (as opposed to, say, just to provide an example). Therefore, make sure that your reasoning is correct and there are no holes in it. A solution that is hard/impossible to decipher/follow might not get full credit (even if it is in principle correct). You do not need to reprove anything that was shown in the class—just state clearly what was proved and where.

Collaboration: These problem sets are meant to be worked on in groups of 2–4 students. Please submit only one writeup per team—it should contain the names of all the students. You are strongly encouraged to solve these problems by yourself. If you must, you may use books or online resources to help solve homework problems, but you must credit all such sources in your writeup and you must never copy material verbatim. Even though only one writeup is submitted, it is expected that each one of the team members is able to fully explain the solutions if requested to do so.

Grading: Each of the two problems will be graded on a scale from 0 to 5.

Warning: Your attention is drawn to the EPFL policy on academic dishonesty. In particular, you should be aware that copying solutions, in whole or in part, from other students in the class or any other source (e.g., ChatGPT) without acknowledgement constitutes cheating. Any student found to be cheating risks automatically failing the class and being referred to the appropriate office.

Homework 1

1 We say that a word $u = a_1 \cdots a_n$ is a *compression* of $w \in \Sigma^*$, denoted by $u \preceq w$, if there are words $w_0, \ldots, w_n \in \Sigma^*$ such that $w = w_0 a_1 w_1 a_2 \cdots a_n w_n$. For a language $L \subseteq \Sigma^*$ define

$$\begin{aligned} \uparrow L &= \{u \in \Sigma^* : w \preceq u \text{ for some } w \in L\}, \\ \downarrow L &= \{u \in \Sigma^* : u \preceq w \text{ for some } w \in L\}. \end{aligned}$$

Show that if L is regular, then both $\uparrow L$ and $\downarrow L$ are also regular.

2 Let $\Sigma_n = \{1, 2, ..., n\}$ be an alphabet with n symbols. Define language L_n to be the set of all words $w \in \Sigma_n^*$ such that at least one symbol of Σ_n does not appear in w. So, for instance, $2112, 23, 2222 \in L_3$ and $123, 3312 \notin L_3$.

2a Give a NFA for L_n with $O(n)$ states.

2b Give a DFA for L_n with 2^n states.

2c Show that any DFA for L_n has at least 2^n states.

(*Hint: If a DFA uses fewer than 2^n states, use a pigeonhole argument to find two (cleverly chosen) words that lead to the same state.*)

1 Solution to Question 1

We say that a word $u = a_1 \cdots a_n$ is a *compression* of $w \in \Sigma^*$, denoted by $u \preceq w$, if there are words $w_0, \ldots, w_n \in \Sigma^*$ such that $w = w_0 a_1 w_1 a_2 \cdots a_n w_n$. For a language $L \subseteq \Sigma^*$ define

$$\begin{aligned} \uparrow L &= \{u \in \Sigma^* : w \preceq u \text{ for some } w \in L\}, \\ \downarrow L &= \{u \in \Sigma^* : u \preceq w \text{ for some } w \in L\}. \end{aligned}$$

Show that if L is regular, then both $\uparrow L$ and $\downarrow L$ are also regular.

Solution : To prove that $\uparrow L$ is regular we design a NFA that accepts $\uparrow L$. Since L is a regular language this means that there exists a DFA that accepts it. Let $M = (Q, \Sigma, \delta, q_0, F)$ be this DFA. A word u belongs to $\uparrow L$ if and only if there exist a compression of u that belongs to L. Essentially, this means that if we remove some characters of u we obtain a string in L. The problem is that we do not know which characters we should keep in u and which one we should remove. This is why we will build an NFA for $\uparrow L$. The NFA will account for this uncertainty.

Let $N_\uparrow = (Q, \Sigma, \delta_\uparrow, q_0, F)$ be a NFA. Note that the individual components of $N_\uparrow$ are practically identic to the ones of M up to transition function that differs. We define the transition function as follow :

$$\delta_\uparrow(q, a) = \begin{cases} \{\delta(q, a), q\}, & \text{if } a \in \Sigma \\ \emptyset, & \text{if } a = \varepsilon \end{cases}$$

Claim : $N_\uparrow$ recognizes $\uparrow L$. What the transition function $\delta_\uparrow$ allows us to do is to ignore some characters of u non-deterministically by staying in our current state q. If a word u belongs to $\uparrow L$ we know that u can be decomposed in the following way : $u = w_0 a_0 w_1 a_1 ... w_n a_n$ with $w_0, ...w_n, a_1, ..., a_n \in \Sigma^*$ such that $w = w_0 w_1 ... w_n \in L$. This is because we know that there exists at least one word w in L that is a compression of u. Thus, crossing out all the characters of

$a_0, a_1, ..., a_n$ in u will effectively yield a word in L. It is guaranteed that one of the computing branches of the NFA $N_\uparrow$ will ignore exactly those characters in $a_0, a_1, ..., a_n$ yielding to a path that accepts u. Finally if u is not in $\uparrow L$ then no matter what characters we ignore we will never reach an accept state since no compression of u is in L. Consequently, all the computational branches of $N_\uparrow$ will reject u. From this we get that u is accepted by $N_\uparrow$ if and only if $u \in \uparrow L$ ie. $N_\uparrow$ recognizes $\uparrow L$. From this we immediatly get that $\uparrow L$ is regular.

Now let's prove that $\downarrow L$ is regular. Again, we will design a NFA that accepts $\downarrow L$ for that purpose. A word u belongs to $\downarrow L$ if and only if it is a compression of some word $w \in L$. This essentially means that u by inserting some characters in u we obtain a word w in L. The issue here is that we do not know which characters to insert, nor do we know where. Hence why, again, there is use for a non-deterministic finite automaton here.

Again, consider the DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L. Let $N_\downarrow = (Q, \Sigma, \delta_\downarrow, q_0, F)$ be a NFA. Note that, again, the components of the tuple of $N_\downarrow$ are identical to the elements of M up to the transition function. We define the transition function as follow :

$$\delta_\downarrow(q, a) = \begin{cases} \{\delta(q, a)\}, & \text{if } a \in \Sigma \\ \{q' \in Q \mid \exists s \in \Sigma : \delta(q, s) = q'\}, & \text{if } a = \varepsilon \end{cases}$$

Claim : $N_\downarrow$ recognizes $\downarrow L$. $N_\downarrow$ is quite similar to M. The only change we made is that we added some epsilon transitions to it. These epsilon transitions allow us to do as if we were adding some characters to our string u to make it belong to L. This is done in a non-deterministic way to account for the fact that we do not know where and what characters to insert in u. More formally, given a word $u \in \downarrow L$, we now that there exists a word $w \in L$ such that w can be decomposed in the following way : $w = u_0 a_0 u_1 a_1 ... u_n a_n$ with $u_0, ..., u_n, a_0, ..., a_n \in \Sigma^*$ such that $u = u_0 u_1 ... u_n$. By inserting the words $a_0, ..., a_n$ in u in the right positions, we would obtain the word $w \in L$. There is at least one computational path in $N_\downarrow$ that will take the right epsilon transitions to do as if those words $a_0, ..., a_n$ were added to u at the right position. This path will lead to an accept state as $w \in L$. Thus $N_\downarrow$ will accept u. Conversely if u does not belong to $\downarrow L$ then no matter what words and where we insert these words in u we will never obtain a word w in L. Suppose that one computational branch in $N_\downarrow$ ends up in an accept state. This would mean that by inserting some characters in u (or inserting none) we obtain a word belonging to L. This would imply that u belongs to $\downarrow L$, contradicting our hypothesis. Thus no computational branch in $N_\downarrow$ terminates in an accept state ie. $N_\downarrow$ does not accept u. We have showed that $N_\downarrow$ accepts u if and only if u belongs to $\downarrow L$. Thus, $\downarrow L$ is indeed regular.

2 Grading scheme to Question 1

1. (1.5 Points) Explains clearly that since L is regular it has a corresponding DFA/NFA. Uses the DFA/NFA of L to build a DFA/NFA for $\uparrow L$ and $\downarrow L$.
2. (1.0 Points) Gives a correct DFA/NFA for $\uparrow L$
3. (0.5 Points) Solid explanation for the correctness of the given DFA/NFA for $\uparrow L$. Full points for an argument involving a double implication ($u \in \mathcal{L}(N) \iff u \in \uparrow L$).
4. (1.0 Points) Gives a correct DFA/NFA for $\downarrow L$
5. (0.5 Points) Solid explanation for the correctness of the given DFA/NFA for $\downarrow L$. Full points for an argument involving a double implication ($u \in \mathcal{L}(N) \iff u \in \downarrow L$).
6. (0.5 Points) The formalism is correct. The notation used is clear. No confusion between singletons and states (between q and $\{q\}$).

3 Solution to Question 2

2a: Construction of an NFA

Consider a language $M_{n,i} \coloneqq \{x \in \Sigma_n^* \mid x \text{ does not contain } i\}$. Then $L_n = \bigcup_{i\in[n]} M_{n,i}$ (here $[n] = \{1, 2, \ldots, n\}$).

Lemma 3.1. *If languages $S_1, \ldots, S_n$ are recognized by NFAs $N_1, \ldots, N_n$ of sizes $\ell_1, \ldots, \ell_n$ respectively, then $S \coloneqq \bigcup_{i\in[n]} S_n$ is recognized by an NFA of size $\sum_{i\in[n]} \ell_i$.*

Proof. Construct the NFA for S as follows: let us unite the NFAs $N_1, \ldots, N_n$ preserving the transitions and final state, set the starting state q_{start} of the NFA as the starting state of N_1 and add ϵ-transitions from q_{start} to the starting states of $N_2, \ldots, N_n$. Then any word accepted by any of $N_1, \ldots, N_n$ is accepted by the new NFA (take the corresponding ϵ-transition and follow an accepting path in the corresponding NFA). On the other hand, if a word w is accepted by the constructed NFA, observe that the accepting path must either be within the copy of N_1 or start with an ϵ-transition and continue within some N_i, hence, removing the first ϵ-transition we get an accepting path in one of N_i. □

By Lemma 3.1 it suffices to show that each $M_{n,i}$ is recognizable by an DFA* with $O(1)$ states.

Consider an DFA with $Q = \{\mathsf{accept}, \mathsf{reject}\}$ and with the transitions defined as $\delta(\mathsf{accept}, i) = \mathsf{reject}$, and $\delta(q, j) = q$ for $(q, j) \in \{\mathsf{accept}, \mathsf{reject}\} \times [n] \setminus (\mathsf{accept}, i)$. The starting and the only accepting state is accept. It is easy to see that the given DFA recognizes $M_{n,i}$: for a word $w = u \circ i \circ v \in M_{n,i}$ we have

$$\delta(\mathsf{accept}, w) = \delta(\delta(\delta(\mathsf{accept}, u), i), v) = \delta(\delta(q, i), v) = \delta(\mathsf{reject}, v) = \mathsf{reject}.$$

For a word $w \notin M_{n,i}$ $\delta(\text{accept}, w) = \text{accept}$, since the i-transition is never taken.

2b: Construction of a DFA

Let $D_{n,i} = (Q_i, \Sigma_n, \delta_i, \mathsf{accept}_i, \mathsf{accept}_i)$ be the 2-state DFA that recognizes $M_{n,i}$. We need to construct a DFA that accepts a word if and only if one of $D_{n,1}, \ldots, D_{n,n}$ accepts it. Let $D = (Q_1 \times \cdots \times Q_n, \Sigma_n, \delta, (\mathsf{accept}_1, \ldots, \mathsf{accept}_n), F)$ with $\delta((q_1, \ldots, q_n), a) = (\delta_1(q_1, a), \ldots, \delta_n(q_n, a))$ and $F = \{(q_1, \ldots, q_n) \mid \exists i \in [n] \colon q_i = \mathsf{accept}_i\}$. In words, the state of D after reading a word w is the tuple of states of $M_{n,1}, \ldots, M_{n,i}$ after reading w. The number of states in D is $|Q_1| \cdot |Q_2| \cdot \cdots \cdot |Q_n| = 2^n$ as required. By definition of accepting states, D accepts precisely the words in $\bigcup_{i\in[n]} M_{n,i} = L_n$.

2c: Lower bound on the number of states in a DFA

Suppose that there exists a DFA D with fewer than 2^n states accepting L_n. For every set $I \subseteq [n]$ define $w_I = i_1 \circ \cdots \circ i_{|I|}$ where $I = \{i_1, \ldots, i_{|I|}\}$ and $i_1 < \cdots < i_{|I|}$ (the ordering is just for concreteness, any word containing all symbols in I and only them works). Then by the pigeonhole principle there exist two words w_I and w_J such that $I \neq J$ and $\delta(q_{\text{start}}, w_I) = \delta(q_{\text{start}}, w_J) = q$. Without loss of generality, suppose $I \setminus J \neq \emptyset$. Then $q_1 \coloneqq \delta(q_{\text{start}}, w_J \circ w_{[n]\setminus I})$ is an accepting state since $w_J \circ w_{[n]\setminus I}$ does not contain any symbols in $I \setminus J$. On the other hand, $q_2 \coloneqq \delta(q_{\text{start}}, w_I \circ w_{[n]\setminus I})$ is not an accepting state since the word $w_I \circ w_{[n]\setminus I}$ contains all symbols of Σ_n. But $q_1 = q_2 = \delta(q, w_{[n]\setminus I})$, which is a contradiction.

*An NFA would have sufficed, but an NFA has a constant number of state if and only if a DFA does.

Grading Scheme

(2a) 1.5 points; -0.5 points if the proof of correctness is imperfect or missing;

(2b) 1.5 points; -0.5 points if the proof of correctness is imperfect or missing;

(2c) 2 points:

- 0.5 points if there is some intuitive justification or a proof in assumption that the DFA must adhere to the structure described in 2b.
- 2 points for a perfect proof; -0.5 points for minor issues in a correct proof.

A common mistake: applying the pigeonhole principle we get that some words w and w' both arrive at some state q of an automaton. One cannot then assume that w and w' have some extra properties that some of the words in the collection do not have (e.g. w containing all symbols of Σ_n).