Automaten und formale Sprachen

Exam: IN2041 / Retake  Date: Tuesday 4th April, 2023
Examiner: Prof. Javier Esparza  Time: 17:00 – 19:00

Working instructions

- This exam consists of 18 pages with a total of 8 problems. Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 45 credits.
- Detaching pages from the exam is prohibited.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.
Problem 1  NFAs and regular expressions (6 credits)

Let NFA-regtoNFA-$\epsilon$ be the algorithm given in the lectures, which given an NFA-reg $M$ as input produces as output an NFA-$\epsilon$ which recognizes the same language as $M$.

a) Let $A$ be the following NFA-reg over the alphabet $\Sigma = \{a, b\}$.

Apply the NFA-regtoNFA-$\epsilon$ algorithm on $A$ to produce an NFA-$\epsilon$ $B$.

We obtain $B$ by the following steps.
b) Consider the NFA-ε $B$ from the previous subproblem. Apply any algorithm which converts an NFA-ε to an NFA recognizing the same language (for example, the NFA-ε to NFA algorithm given in the lectures), on the NFA-ε $B$, to produce an NFA $C$.

The following is one possible solution for $C$. 

![Diagram of NFA C]
c) Give a minimal NFA recognizing $L(ab^* + ba^*)$. Note that you have to produce such an NFA and prove that any NFA which has strictly less states cannot recognize $L(ab^* + ba^*)$.

The following is a minimal NFA for $L(ab^* + ba^*)$.

We now prove that there is no 2-state NFA that can recognize $L = L(ab^* + ba^*)$. For the sake of contradiction, suppose $D$ is a 2-state NFA which recognizes $L$. Let $q$ be some initial state of $D$. $q$ cannot be final as otherwise $D$ accepts $\epsilon \notin L$. $D$ must have a final state, as otherwise $D$ accepts nothing. Let $q' \neq q$ be a final state of $D$. Note that $q'$ also cannot be initial as otherwise $D$ accepts $\epsilon$. Hence, we have exactly one initial state $q$ and one final state $q' \neq q$.

Since $a, b \in L$, it follows that $q \xrightarrow{a} q'$ and $q \xrightarrow{b} q'$ are transitions of $D$. Further since $ab \in L$, it follows that $q \xrightarrow{a} p \xrightarrow{b} q'$ for some $p \in \{q, q'\}$. If $p = q$, then $aa$ is accepted by $D$ because of the run $q \xrightarrow{a} q \xrightarrow{a} q'$. If $p = q'$, then $bb$ is accepted by $D$ because of the run $q \xrightarrow{b} q' \xrightarrow{b} q'$. In either case, we have a contradiction.

Remark: We note that a residual-based argument does not work here, as it works only for DFAs.
Problem 2  Occurrences of subwords (5 credits)

Let $\Sigma = \{a, b\}$.

a) Let $L$ be the language of finite words over $\Sigma$ defined as

$$L = \{w : w \text{ contains no occurrence of } aba\}$$

Give the minimal DFA for the language $L$.  **Hint:** It might help to think in terms of pattern matching.

We consider the minimal DFA obtained by the pattern matching algorithm for the pattern $p = aba$, with the difference that the final state has self-loops for $\Sigma = \{a, b\}$.

![DFA Diagram]

Note that this is the minimal DFA for $\Sigma^* aba \Sigma^*$. Flipping the accepting and rejecting states gives the minimal DFA for $L$.

b) Let $L'$ be the language of finite words over $\Sigma$ defined as

$$L' = \{w : w \text{ contains at least two distinct (but possibly overlapping) occurrences of } aba\}$$

For example, the words $baba\overline{aba}a\overline{aba}$, $abaaba \overline{aba} \in L'$ but $ab, abaab \not\in L'$.

Give a regular expression for the language $L'$.

One possible solution is $\Sigma^* aba \Sigma^* aba \Sigma^* + \Sigma^* ababa \Sigma^*$.
c) Give the minimal DFA for the language \( L' \) defined in the previous subproblem. **Hint:** It might help to think in terms of pattern matching. The final answer should have 7 states.

Consider two copies of the minimal DFA for the pattern \( p = aba \), with the difference that the final state has self-loops for \( \Sigma = \{a, b\} \).

We remove the accepting state in the first copy and divert its incoming \( a \) transition to the second state of the second copy, to get our required answer.
Problem 3  Fixed-length languages (4 credits)

For a fixed-length language $L$ over $\Sigma = \{a, b\}$ we denote by $q_L$ the state of the master automaton representing $L$. We also denote by $M(L)$ the fragment of the master automaton that contains $q_L$ and all its residuals, that is, it contains all the states between $q_L$ and $q_\emptyset$, \textit{including} $q_L$ and $q_\emptyset$ (and no other states).

How many fixed-length languages $L$ of length 3 exist such that $M(L)$ contains exactly 5 states? \textbf{For instance, here is an example of a language $L$ of length 3 such that $M(L)$ contains exactly 5 states.}

\begin{center}
\begin{tikzpicture}
  \node (A) at (0,0) [circle, draw] {$\{aab, abb\}$};
  \node (B) at (1,0) [circle, draw] {$\{ab, bb\}$};
  \node (C) at (2,0) [circle, draw] {$\{b\}$};
  \node (D) at (3,0) [circle, draw] {$\emptyset$};

  \path
    (A) edge [above] node {$a$} (B)
    (B) edge [above] node {$a$} (C)
    (C) edge [above] node {$a, b$} (D)
    (D) edge [below] node {$b$} (C)
    (B) edge [below] node {$b$} (D);
\end{tikzpicture}
\end{center}

If there are 5 states in $M(L)$, 2 of them must be in level 0 (those are $q_\emptyset$ and $q_\epsilon$) and in every other layer there is exactly 1 state. A transition from the state in level $i$ can either go to the state in level $i - 1$ or to $q_\emptyset$, with the restriction that at least one transition must go to the state in level $i - 1$. Hence, the edge between consecutive levels can be labeled either with $a$ or with $b$ or with $a, b$. Since we have 3 levels and 3 options for each level, there are in total $3^3 = 27$ different languages.
Problem 4  First-order logic on words (6 credits)

Let $\Sigma = \{0, 1\}$ and let $n \geq 1$ be some natural number. Given a string $w \in \Sigma^*$, let $\text{msbf}(w)$ denote the number represented by $w$ in binary in the most significant bit first encoding. For example, if $w = 0011$, then $\text{msbf}(w) = 3$ and if $w = 1011$, then $\text{msbf}(w) = 11$.

For the purposes of this exercise, whenever you are asked to construct a formula over $\text{FO}(\Sigma)$, in addition to the syntax of $\text{FO}(\Sigma)$, you are only allowed to use the following macros: $\text{first}(x)$, $\text{last}(x)$, $x = y$, $y = x + k$, $y < x + k$ and $y < k$ for some number $k$. If you use any other macros, you have to explicitly give the FO formulas that these macros stand for.

a) For $n \geq 1$, consider the language $L_n := \{w : w \in \Sigma^{2n}\}$. Give a formula $\phi_n$ over $\text{FO}(\Sigma)$ which recognizes $L_n$. The formula $\phi_n$ must be of size polynomial in $n$, i.e., there must be a polynomial $p$ such that the size of each $\phi_n$ is at most $p(n)$.

One possible solution is

$$\phi_n := \exists x, y. \text{first}(x) \land \text{last}(y) \land y = 2n - 1 + x$$

Intuitively, this formula states that there are two positions $x$ and $y$ such that $x$ is the first position, $y$ is the last position and the distance between them is $2n$.

b) For $n \geq 1$, consider the language $L'_n := \{uu : u \in \Sigma^n\}$. Give a formula $\phi'_n$ over $\text{FO}(\Sigma)$ which recognizes $L'_n$. The formula $\phi'_n$ must be of size polynomial in $n$, i.e., there must be a polynomial $p'$ such that the size of each $\phi'_n$ is at most $p'(n)$.

One possible solution is

$$\phi'_n := \phi_n \land \forall x. x < n + 1 \implies \exists y. (y = x + n \land (Q_0(x) \iff Q_0(y)) \land (Q_1(x) \iff Q_1(y)))$$

Intuitively, this formula states that the word has length exactly $2n$ and further for every position $x < n+1$, the letter at position $x$ is the same as the letter at position $y = x + n$. 

A Sample Solution
For $n \geq 1$, consider the language $L''_n := \{uv : u, v \in \Sigma^n, \text{msbf}(u) \geq \text{msbf}(v)\}$. Give a formula $\phi''_n$ over $\text{FO}(\Sigma)$ which recognizes $L''_n$. The formula $\phi''_n$ must be of size polynomial in $n$, i.e., there must be a polynomial $p''$ such that the size of each $\phi''_n$ is at most $p''(n)$.

One possible solution is

$$
\phi''_n := \phi_n \land (\phi'_n \lor (\exists x, y. \ x < n + 1 \land y = x + n \land Q_1(x) \land Q_0(y) \land
(\forall x'. x' < x \implies \exists y'. (y' = x' + n \land (Q_0(x') \iff Q_0(y')) \land (Q_1(x') \iff Q_1(y')))))
$$

Intuitively, this formula states that the word has length exactly $2n$ and

- Either $\phi'_n$ holds, in which case the word is of the form $uu$ with $u \in \Sigma^n$
- Or there is a position $x < n + 1$ such that the letter at $x$ is 1 and the letter at $y = x + n$ is 0 and for every $x' < x$, the letter at $x'$ and the letter at $y' = x' + n$ are the same.
Problem 5  Operations on languages (6 credits)

Let $\Sigma = \{a, b\}$. Let $L \subseteq \Sigma^*$ be any language consisting of finite words over $\Sigma$. We define the $\omega$-language $L_{\omega} \subseteq \Sigma^\omega$ as

$$L_{\omega} = \{wa^\omega : w \in L\}$$

Note that $L_{\omega}$ is a language of infinite words over $\Sigma$. Intuitively, each word in $L_{\omega}$ is obtained by first taking some finite word $w \in L$ and then adding the infinite suffix $a^\omega$ to it.

a) Prove or disprove: If $L \subseteq \Sigma^*$ is regular, then $L_{\omega}$ is $\omega$-regular.

The claim is true. Suppose $L$ is a regular language. Let $r$ be a regular expression for $L$. Then, the $\omega$-regular expression $r \cdot \{a\}^\omega$ recognizes $L_{\omega}$.

b) Prove or disprove: If $L_{\omega}$ is $\omega$-regular for some $L \subseteq \Sigma^*$, then $L$ is regular.

The claim is false. Let $L$ be any non-regular language over $\{a\}$, for example $\{a^{2n} : n \geq 1\}$. Then $L_{\omega} = a^\omega$ which is $\omega$-regular.
The claim is true. Suppose \( L' := (L \cdot \{ b \})a^\omega \) is \( \omega \)-regular. Let \( A = (Q, \Sigma, \delta, Q_0, F) \) be an NBA which recognizes \( L' \). Let \( Q' \) be the set of states of \( A \) which accept \( ba^\omega \). Let \( B \) be the NFA given by \( B = (Q, \Sigma, \delta, Q_0, Q') \). We claim that \( B \) recognizes \( L \).

Suppose \( w \in L \). Then there is an accepting run for \( wba^\omega \) over \( A \). Let \( q \) be the state that is reached along this run after reading \( w \). By definition, \( q \in Q' \) and so it follows that \( w \) is also accepted over \( B \).

Suppose \( w \) is accepted by \( B \). Then there is an accepting run of \( w \) over \( B \) which ends in some state in \( Q' \). By definition, this means that there is an accepting run for \( wba^\omega \) over \( A \) and so \( wba^\omega \in L' = (L \cdot \{ b \})a^\omega \). Hence, \( wba^\omega = w'ba^\omega \) for some \( w' \in L \). If \( w \) is a strict prefix of \( w' \), then let \( w' = ww'' \) for some \( w'' \neq \epsilon \). We then have \( ba^\omega = w''ba^\omega \), which leads to a contradiction. A similar argument can be made for the case of \( w' \) being a strict prefix of \( w \). It follows then that \( w = w' \) and so \( w \in L \).
Problem 6  Acceptance conditions (10 credits)
Throughout this exercise, we will only be considering languages of infinite words over $\Sigma = \{a, b, c\}$.

a) Consider the $\omega$-regular language $L_1$ defined as

$$L_1 = \{ w \in \Sigma^\omega : ab \text{ and } ac \text{ appear infinitely often in } w \}$$

Give a non-deterministic Büchi automaton $(A_1, F_1)$ which accepts $L_1$ such that $A_1$ has at most 5 states.

The following is one possible solution.

![Diagram of non-deterministic Büchi automaton](image)

b) Give a non-deterministic generalized Büchi automaton $(A'_1, F'_1)$ which accepts $L_1$ such that $A'_1$ has at most 3 states.

The following is one possible solution whose generalized Büchi condition is $\{\{2\}, \{3\}\}$.

![Diagram of non-deterministic generalized Büchi automaton](image)
c) Consider the $\omega$-regular language $L_2$ defined as

$$L_2 = \{ w \in \Sigma^\omega : ab \text{ appears infinitely often in } w \text{ and } ac \text{ appears finitely often in } w \}$$

Give a **deterministic** Rabin automaton $(A_2, F_2)$ which accepts $L_2$ such that $A_2$ has at most 4 states.

The following is one possible solution whose Rabin condition is $$\{\langle \{3\}, \{4\} \rangle\}$$, i.e., 3 must be visited infinitely often and 4 must be visited finitely often.

![Deterministic Rabin automaton](image)

d) **This is a bonus subproblem.**

Give a non-deterministic Muller automaton $(A'_2, F'_2)$ which accepts $L_2$ such that $A'_2$ has at most 3 states.

The following is one possible solution whose Muller condition is $$\{\{1, 2\}\}$$.

![Non-deterministic Muller automaton](image)
Consider the following Büchi automaton over \( \Sigma = \{a, b\} \).

![Büchi Automaton Diagram]

a) Draw \( \text{dag}(\text{aab}^\omega) \) and give an odd ranking for it.

The \( \text{dag}(\text{aab}^\omega) \) is presented below.

![Dag Diagram]

One possible way to define an odd ranking is:

\[
    f(s, i) = \begin{cases} 
        1 & \text{if } (s = p \text{ or } s = q) \text{ and } (s, i) \text{ appears in } \text{dag}(\text{aab}^\omega), \\
        0 & \text{if } s = r \text{ and } (s, i) \text{ appears in } \text{dag}(\text{aab}^\omega), \\
        \bot & \text{otherwise.}
    \end{cases}
\]
b) Find an $\omega$-word $w$ such that $\text{dag}(w)$ does not have an odd ranking. Draw $\text{dag}(w)$ and prove that it does not have an odd ranking by analyzing the $\text{dag}$.

For example, $ab^\omega$ has this property. Below we sketch $\text{dag}(ab^\omega)$.

There are only two infinite paths in this dag and starting from layer 2 both of them visit only the state $r$ which is accepting. Hence, there can be no odd ranking in this case.
Problem 8  Linear Temporal Logic (4 credits)

a) Let $AP = \{p, q, r\}$ and let $\Sigma = 2^{AP}$. Consider the formulas

$$\phi := (p U q) U r \quad \text{and} \quad \xi := p U (q U r)$$

Give four computations $\sigma_1, \sigma_2, \sigma_3, \sigma_4$, all of them over $AP$, such that

- $\sigma_1 \models \phi$ and $\sigma_1 \models \xi$
- $\sigma_2 \models \phi$ and $\sigma_2 \not\models \xi$
- $\sigma_3 \not\models \phi$ and $\sigma_3 \models \xi$
- $\sigma_4 \not\models \phi$ and $\sigma_4 \not\models \xi$

There are many possible solutions, here is an example:

- $\sigma_1 = \{r\}^\omega$
- $\sigma_2 = \{p\} \{q\} \{p\} \{q\} \{r\}^\omega$
- $\sigma_3 = \{p\} \{r\}^\omega$
- $\sigma_4 = \emptyset^\omega$

b) This is a bonus subproblem.

Let $AP = \{p, q\}$ and let $\Sigma = 2^{AP}$. Give an $\omega$-regular expression over $\Sigma$ for the set of all computations which satisfy the formula

$$\varphi := (p U q) U p$$

$$\{q\}^* (\{p\} + \{p, q\}) \Sigma^\omega$$
c) This is a bonus subproblem.

Consider the formula $\varphi$ defined in the previous subproblem. Use the $\omega$-regular expression you defined in the previous subproblem to derive a formula $\varphi'$ such that $\varphi'$ and $\varphi$ are equivalent and $\varphi'$ is of strictly smaller size than $\varphi$. 

$q \mathsf{U} p$
Additional space for solutions—clearly mark the (sub)problem your answers are related to and strike out invalid solutions.