University of Freiburg
Department of Computer Science
Prof. Dr. F. Kuhn

Exam Theoretical Computer Science - Bridging Course

Tuesday, March 13, 2018, 10:00-11:30

Name: .....

Matriculation No.: .....

Signature: .....

Do not open or turn until told so by the supervisor!

- Write your **name** and **matriculation number** on this page and sign the document!
- Write your name on **all sheets**!
- Your **signature** confirms that you feel physically and mentally able to write the exam and that you have answered all questions without any help.
- Write legibly and only use a pen (ink or ball point). Do **not use red**! Do **not use a pencil**!
- This is an **open book exam** therefore printed or hand-written material is allowed.
- However, **no electronic devices** are allowed.
- There are **eight tasks** (with several sub-tasks each) and there is a **total of 100 points**.
- **35 points are sufficient** in order to pass the exam. **70 points** are sufficient to get the best mark.
- Only **one solution per task** is considered! Make sure to strike out alternative solutions, otherwise the one yielding the minimal number of points is considered.
- **Detailed steps** might help you to get more points in case your final result is incorrect.
- The keywords **Show...** or **Prove...** indicate that you need to prove or explain your answer carefully.
- The keywords **Give...** or **State...** indicate that you only need to provide a plain answer.
- You may use information given in a **Hint** without explaining them.
- **Read each task thoroughly** and make sure you understand what is expected from you.
- **Raise your hand** if you have a question regarding the formulation of a task.
- **Use the space below each task and the back of the sheet for your solution.** The last two sheets of this exam are blank and can be used for solutions. If you need additional sheets, raise your hand.

Question	1	2	3	4	5	6	7	8	**Total**
Points									
Maximum	9	23	7	6	11	17	14	13	100

Task 1: Basic Mathematical Skills (9 Points)

1. Prove the equation $\sum_{i=1}^{n}(i - 1/2) = \frac{n^2}{2}$ for all $n \in \mathbb{N}$ **by induction** on n. *(5 Points)*
2. Let A, B, C be sets. Prove the following **implication**: *(4 Points)*

$$A \cap C = \emptyset \quad \Longrightarrow \quad (B \setminus A) \cap C = B \cap C.$$

***Remark**: $A \setminus B := A \cap \overline{B}$ is the 'set minus' operator, describing all elements of A that are not in B.*

Sample Solution

1. *Induction base:* The statement is true for $n = 1$, since $\sum_{i=1}^{1} i - 1/2 = 1/2 = \frac{1^2}{2}$. *(1 Points)*

   Induction hypothesis: Presume the statement holds for an arbitrary, fixed $n \in \mathbb{N}$.

   Induction step: $\sum_{i=1}^{n+1} i - 1/2 = n + 1 - 1/2 + \sum_{i=1}^{n} i - 1/2 \overset{\text{Ind. hyp.}}{=} n + 1/2 + \frac{n^2}{2} = n + 1/2 + \frac{(n+1)^2 - 2n - 1}{2} = \frac{(n+1)^2}{2}$. *(4 Points)*

2. Let $A \cap C = \emptyset$. We show both inclusions. $\subseteq$: Let $x \in (B \setminus A) \cap C$. Then x is obviously also contained in $B \cap C$.

   $\supseteq$: Let $x \in B \cap C$. Then $x \in C$ and due to $A \cap C = \emptyset$ it is not contained in A. From $x \in B$ and $x \notin A$ we deduce that $x \in (B \setminus A)$. That is $x \in (B \setminus A) \cap C$.

Task 2: DFAs, NFAs (23 Points)

1. Consider the following languages over the alphabet $\Sigma = \{a, b\}$. Let L' be the language defined by the regular expression $(aa)^*(bb)^* \cup b$. Let $L = \Sigma^* \setminus L'$.

   (a) What is the shortest word in the language L? *(2 Points)*

   (b) Draw a DFA for language L'. *(6 Points)*
   Remark: You get partial points if you draw an automaton for the language defined by the regular expression $(aa)^*(bb)^*$.

   (c) Draw a DFA for language L or explain how to modify the DFA for L' to obtain a DFA for L. *(4 Points)*

2. Let S be the language consisting of all words of the form $w_1w_2w_3$ with $w_1, w_2, w_3 \in \{a, b, c\}^*$ and

   - w_1 contains no a's, and
   - w_2 contains no b's and no c's and an even number of a's, and
   - w_3 contains no b's and no c's.

   Give a regular expression that generates S and uses at most three letters of Σ. Here a letter is counted more than once if it occurs more than once in the regular expression, e.g., the expression $aa(bbb)^*ca$ uses 7 letters, three times the letters a, three times the letters b and once the letter c. *(4 Points)*

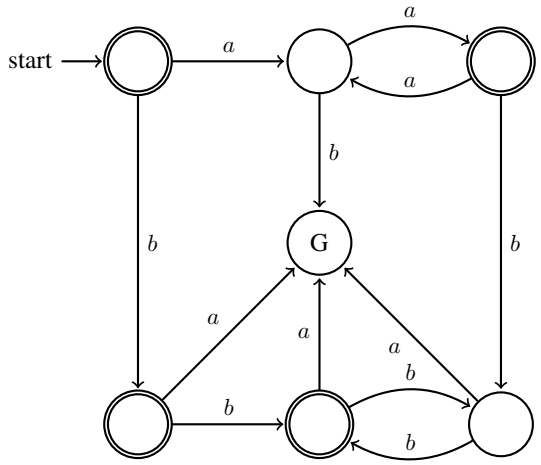
3. State **the difference** between **deterministic** and **non-deterministic** finite automatons. Which one (DFA or NFA) can recognize the larger class of languages? *(2 Points)*

4. Let $T := \{a^n cb^{n+2} \mid n \geq 0\}$. Either prove that T is regular by giving the corresponding DFA or prove that T is not regular. *(5 Points)*

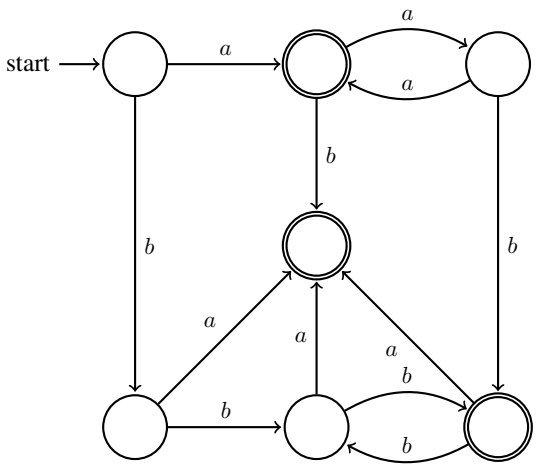
Sample Solution

1. (a) As ϵ and b are not contained in L the shortest word in L is a.

   (b)



(c) Exchange accepting and non-accepting states. This does not work for an NFA. Make sure that there is a garbage state!



2. $(b \cup c)^* \cdot (aa)^* a^* = (b \cup c)^* \cdot a^*$.

3. The transition of a DFA is $\delta : Q \times \Sigma \to Q$. The transition function of an NFA is $\delta : Q \times \Sigma_\epsilon \to 2^Q$, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. Both recognize the same class of languages.

4. We proof that T is not regular by using the pumping lemma. Assume T was regular and let p be the pumping length. We consider the word $w = a^p c b^{p+2}$. Consider any partition $w = xyz$ with $|xy| \leq p$ and $|y| > 0$. Then y only consists of as, so $xyyyz$ contains at least as many as as bs, which means that it is not in T.

Task 3: Context-Free Languages (7 Points)

1. **Give** an **example** of a language that can be recognized by a pushdown automaton but not by a non-deterministic finite automaton. *(no proof required)* *(1 Points)*

Consider the following language.

$$L = \{(ab)^n(ba)^{2n} \mid n \geq 1\} .$$

2. What is the shortest word in the language? *(1 Points)*
3. **Give** a context free grammar that generates L. *(5 Points)*

Sample Solution

1. $\{a^n b^n \mid n \in \mathbb{N}\}$
2. $abbaba$
3. $$S \to abS_1baba$$
$$S_1 \to abS_1baba \mid \varepsilon$$

Task 4: Turing machines (6 Points)

1. **Is there** a Turing machine that recognizes the language L defined in task 2.1 of this exam?

   Remark: Do not just answer 'yes' or 'no' but explain your answer with a few (!) words. (2 Points)

2. Consider the following types of computational models: GNFA, NFA, DFA, multitape non-deterministic Turing machine, single-tape deterministic Turing machine, PDA, and multitape deterministic Turing machine.

   Order these types according to their **computational power**. Write $X = Y$ if the class of languages recognized by machines of type X equals the class of languages recognized by machines of type Y, and write $X < Y$ if there is a language that is recognized by a machine of type Y but by none of type X. *(4 Points)*

Sample Solution

1. Yes, any DFA can be easily simulated by a Turing machine.

2. DFA $=$ NFA $=$ GNFA $<$ PDA $<$ single-tape deterministic Turing machine $=$ multitape deterministic Turing machine $=$ multitape non-deterministic Turing machine

Task 5: $\mathcal{O}$ - Notation (11 Points)

State whether the following claims are true or false *(1 point each)*. Then **prove or disprove** the claim. Use the definition of the $\mathcal{O}$-notation.

1. $\sqrt{2}^{\log_2 n} \in \mathcal{O}(\sqrt{2} \cdot n)$. *(1+4 Points)*
2. $3^{2n} \in \mathcal{O}(2^{3n})$. *(1+5 Points)*

Sample Solution

1. The claim is true. $\sqrt{2}^{\log_2 n} = \sqrt{2^{\log_2 n}} = \sqrt{n} \leq \sqrt{2} \cdot n$ for $n \geq 1$ (i.e., choose $c = 1$ and $n_0 = 1$).
2. The claim is false. Let $c > 0$.

$$\begin{aligned} & 3^{2n} \leq c \cdot 2^{3n} \\ \Leftrightarrow & 9^n \leq c \cdot 8^n \\ \Leftrightarrow & \left(\frac{9}{8}\right)^n \leq c \\ \Leftrightarrow & n \leq \log_{\frac{9}{8}}(c) \end{aligned}$$

This means that for any $c > 0$ and any n_0, there is an $n \geq n_0$ such that $3^{2n} > c \cdot 2^{3n}$ (choose $n = \max\{n_0, \log_{\frac{9}{8}}(c) + 1\}$).

Task 6: Decidability (17 Points)

1. Consider the problem MATCHING:

   $$\text{MATCHING} := \{\langle G, k\rangle \mid G \text{ is a simple graph and has a } \textbf{k-matching}\}.$$

   A **k-matching** of $G = (V, E)$ is a subset $M \subseteq E$ with size k such that there are no two adjacent edges $e \neq e' \in M$.

   Show that MATCHING is **decidable** by giving an algorithm (abstract description or pseudo-code) that decides whether a graph has a k-matching. *(6 Points)*

2. Let H be the language of the halting problem. Give a language L such that $L \cap H$ decidable and give a language K such that $K \cap H$ is undecidable. Prove your claims. *(5 Points)*

3. Let Σ be a fixed finite alphabet. Show that the language of deterministic finite automatons (DFAs) on Σ that accept no word is decidable. Formally, show that

   $$L = \{\langle A\rangle \mid A \text{ is a deterministic finite automaton with } L(A) = \emptyset\}$$

   is a decidable language.

   Remark: You can use that it is not difficult to construct a Turing machine which tests whether an input is the well formed encoding of a deterministic finite automaton. (6 Points)

Sample Solution

1. Given an input graph $G = (V, E)$ with input number k, we test for all of the at most $\mathcal{O}(|E|^k)$ possible subsets of the edges of size k whether they form a matching. *Remark: The exact number of possible sets does not matter, it is sufficient to state that there are only finitely many.*

   To check whether a subset M is a matching we iterate through all pairs of edges $e, e' \in M$. For all $\{u, v\} = e \neq e' = \{u', v'\}$ we test whether they are adjacent. The edges are adjacent if $|\{u, v, u', v'\}| \leq 3$. (Remark: One could argue more detailed here about how to do this test.) If we do not find such an adjacent pair we accept the set M and (G, k) if we find an adjacent pair of edges for each possible set M we reject (G, k). *(1 Points)*

   Remark: The run-time of the algorithm does not need to be analyzed in order to obtain the point. It suffices to point out that all involved steps are finished after finite time.

2. $L = \emptyset$ and $K = H$.

3. Let B be a DFA such that the language generated by B is $\emptyset$. That is, $L_B = \emptyset$. (It is easy to see that this is always possible regardless of the alphabet Σ simply by not having an accepting state.) We have shown in the video lecture that testing equivalence for two DFA is a decidable problem. Let M be such a Turing machine that can test equivalence for two DFA. We can construct a Turing machine M', such that upon input A where A is a DFA,

it will run M with input $\langle A, B \rangle$. If M accepts the input, then M' accepts the input A as well, otherwise M' rejects. Since M will give a definite answer in finite time, we know M' will give a definite answer in finite time as well. Hence, we know $\mathcal{L}$ is decidable.

Task 7: Complexity Theory (14 Points)

1. **Give** a language that cannot be decided by any Turing machine that runs in at most $O(n!)$ steps where n is the length of the input. *(2 Points)*

2. Given a graph $G = (V, E)$, a **vertex cover** is a subset $U \subseteq V$ of nodes of G such that every edge of G is adjacent to a node in the subset U.

   The VERTEXCOVER-problem is defined as

   $$\text{VERTEXCOVER} := \{\langle G, k\rangle \mid G \text{ is a simple graph and has a } \textbf{vertex cover} \text{ of size at most } k\}.$$

   Show that VERTEXCOVER is $\mathcal{NP}$-complete. *(12 Points)*

   You may use that the problem INDEPENDENTSET is $\mathcal{NP}$-complete.

   $$\text{INDEPENDENTSET} := \{\langle G, k\rangle \mid G \text{ is a simple graph and has an } \textbf{independent set} \text{ of size at least } k\}.$$

   An **independent set** of size k of $G = (V, E)$ is a subset $I \subseteq V$ of nodes, such that $|I| = k$ and $\{u, v\} \notin E$ for all $u, v \in I$.

Sample Solution

1. The halting problem is not Turing-decidable and thus not decidable by a Turing machine in at most $O(n!)$ steps.

2. We reduce INDEPENDENTSET to VERTEXCOVER.

   Claim: If $G = (V, E)$ is a simple graph, then $S \subseteq V$ is an independent set iff $V \setminus S$ is a vertex cover.

   It follows that a graph has an independent set of size at least k iff it has a vertex cover of size at most $|V| - k$. So for a given INDEPENDENTSET instance $\langle G, k\rangle$ we check if $\langle G, |V| - k\rangle$ is in VERTEXCOVER. The transformation from $\langle G, k\rangle$ to $\langle G, |V| - k\rangle$ can be done in polynomial time.

   Proof of the claim: Suppose $S \subseteq V$ is an independent set. Let (v, w) be an edge. Not both v and w can be in S, thus one of it is in $V \setminus S$. Now suppose $V \setminus S$ is a vertex cover. Let v and w be in S. Then $(v, w) \notin S$, because otherwise (v, w) would not be covered.

   To show that VERTEXCOVER is in $\mathcal{NP}$ we use the guess and check procedure.

   Guess: Given an instance $\langle G, k\rangle$ of VERTEXCOVER guess a subset $U \subseteq V$ of size at most k.

   Check: For each edge $(v, w) \in E$, go through U and check if v or w is in U. This takes $\mathcal{O}(|E| \cdot k)$ steps.

Task 8: Logic (13 Points)

1. Consider the following propositional formula

$$\psi := (x \to y \vee z) \wedge (y \to \neg x) \wedge (x \wedge z \to y) \wedge x \ .$$

   (a) Transfer ψ into an equivalent formula in **conjunctive normal form (CNF)**. *(2 Points)*

   (b) Use the **resolution calculus** to show that ψ is unsatisfiable, i.e., convert ψ into an equivalent knowledge base KB (in CNF) and derive the empty clause from KB by resolution. *(5 Points)*

2. Consider the following **first order logical** formulae

$$\begin{aligned}\varphi_1 &:= \forall x \neg R(x,x)\\ \varphi_2 &:= \forall x,y,z\ (R(x,y) \wedge R(y,z) \to R(x,z))\\ \varphi_3 &:= \exists x \forall y\ (x \neq y \to R(x,y))\end{aligned}$$

   where x, y are variable symbols and R is a binary predicate. Give an interpretation

   (a) I_1 which is a model of $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$. *(3 Points)*

   (b) I_2 which is a model of $\varphi_1 \wedge \varphi_2 \wedge \neg\varphi_3$. *(3 Points)*

   ***Remark**: No proof required.*

Sample Solution

1. (a)
$$(\neg x \vee y \vee z) \wedge (\neg y \vee \neg x) \wedge (\neg x \vee \neg z \vee y) \wedge x$$

   (b) KB$= \{\{\neg x, y, z\}, \{\neg y, \neg x\}, \{\neg x, \neg z, y\}, \{x\}\}$.

$$\begin{aligned}\{x\}, \{\neg x, y, z\} &\vdash_{\mathbf{R}} \{y, z\}\\ \{y, z\}, \{\neg y, \neg x\} &\vdash_{\mathbf{R}} \{z, \neg x\}\\ \{\neg y, \neg x\}, \{\neg x, \neg z, y\} &\vdash_{\mathbf{R}} \{\neg x, \neg z\}\\ \{z, \neg x\}, \{\neg z, \neg x\} &\vdash_{\mathbf{R}} \{\neg x\}\\ \{x\}, \{\neg x\} &\vdash_{\mathbf{R}} \square\end{aligned}$$

2. (a) $I_1 := \langle \mathbb{N}, R^{I_1} \rangle$ where $R^{I_1}(x,y) :\Leftrightarrow x <_{\mathbb{N}} y$.

   (b) $I_2 := \langle \mathbb{Z}, R^{I_1} \rangle$ where $R^{I_2}(x,y) :\Leftrightarrow x <_{\mathbb{Z}} y$.