Theoretical Computer Science (Bridging Course)

Dr. G. D. Tipaldi
F. Boniardi
Winter Semester 2014/2015

University of Freiburg
Department of Computer Science

Revision Sheet

Question 1 (Finite Automata, $8 + 6$ points)

(a) Give a regular expression for each of the following languages:

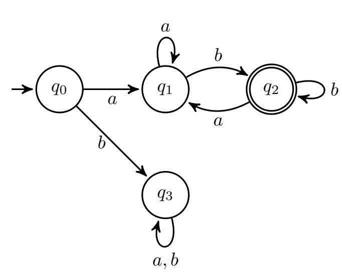
- (i) all strings over $\{0, 1\}$ that are at least three symbols long and have a 0 at their resp. 3rd positions
- (ii) all strings over $\{0, 1\}$ that have odd length, if starting with a 0, and even length otherwise
- (iii) all strings over $\{a, b\}$ that contain the substrings aa or bab
- (iv) all strings over $\{a, b\}$ that do not contain the substring ba

Solution: Setting $\Sigma := \{0, 1\}$ or $\Sigma := \{a, b\}$ according to the context, some possible solutions are

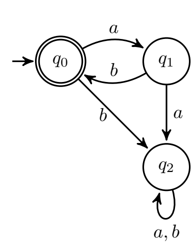
- (i) $\Sigma\Sigma 0 \Sigma^*$.
- (ii) $(0 \cup 1\Sigma)(\Sigma\Sigma)^* \cup \epsilon$.
- (iii) $\Sigma^*(aa \cup bab)\Sigma^*$.
- (iv) If a string on $\{a, b\}$ does not contain ba as a substring, it means that the sequence of symbols is not decreasing (wrt the lexicographic order). Thus a solution is a^*b^*.

(b) Draw a DFA equivalent to each of the following regular expressions:

- (i) $a(a \cup b)^*b$.
  Solution: a possible DFA is:



- (ii) $(ab)^*$
  Solution: below an example of DFA that accepts such language



Question 2 (Regular languages, 14 points)

Let $\Sigma = \{a, b\}$. Use the pumping lemma to prove that:

$$L = \{a^n b^{2n} a^{3n} \mid n \geq 0\}$$

is not regular.

Any other proof techniques will **not** receive any points.

Solution: let's assume that L is a regular language. According to the Pumping Lemma, there must exist an integer $p > 0$ so that for any word $w \in L$ with $|w| \geq p$, there exists a decomposition of w in substrings xyz (i.e. $w = xyz$) such that the following properties are satisfied:

- $|xy| \leq p$.
- $y \neq \epsilon$
- $xy^k z \in L$ for any $k \in \mathbb{N}_0$.

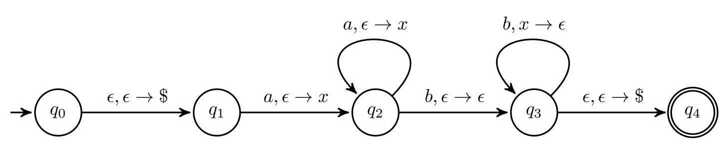
It is apparent that if we select $w = a^p b^{2p} a^{3p} \in L$, then $|w| \geq p$. Furthermore, if such x, y, z exist, then $xy^0 z = a^{p-|y|} b^{2p} a^{3p} \notin L$. This contradicts the Pumping Lemma.

Question 3 (Context-free languages, 7+7 points)

(a) Give the state diagram of a PDA recognizing the language

$$A = \{a^i b^j \mid i > 0 \text{ and } j = i + 1\}.$$

Solution: it is easy to see that the following PDA accept the language A:



(b) Let $G = \langle \{S, X, Y, Z\}, \{a, b, c, d\}, R, S\rangle$ be the CFG with rules:

$$\begin{aligned} S &\to XYZ \\ X &\to Xa \mid b \mid \varepsilon \\ Y &\to b \mid c \\ Z &\to cd \end{aligned}$$

Specify a CFG G_0 in Chomsky Normal Form such that $L(G_0) = L(G)$.

Solution: we apply the standard procedure to convert CFG in Chomsky Normal Form. The procedure is listed below:

Remove the ϵ-rules:

$$\begin{aligned} S &\to XYZ \mid YZ \\ X &\to Xa \mid b \mid a \\ Y &\to b \mid c \\ Z &\to cd \end{aligned}$$

Remove $[X \to Xa]$ by introducing the auxiliary variable A and the rule $[A \to a]$:

$$
\begin{aligned}
&S \to XYZ \mid YZ\\
&X \to XA \mid b \mid a\\
&Y \to b \mid c\\
&A \to a\\
&Z \to cd
\end{aligned}
$$

Add auxiliary variables U, C, D and related rules to complete the CNF.

$$
\begin{aligned}
&S \to XU \mid YZ\\
&U \to YZ\\
&X \to XA \mid b \mid a\\
&Z \to CD\\
&A \to a\\
&C \to c\\
&D \to d\\
&Y \to b \mid c
\end{aligned}
$$

Question 4 (NP-completeness, $7 + 7$ points)

Let $\mathcal{G} := (V, E)$ be an undirected graph. A *vertex cover* of $\mathcal{G}$ is a vertex set $C \subseteq V$ such that for all $\langle u, v\rangle \in E$, $u \in C$ or $v \in C$.

Let $\mathcal{S} := (S, \mathcal{C})$ be a subset collection, i.e., S is a finite set and $\mathcal{C} = \{C_1, \ldots, C_n\}$ where $C_i \subseteq S$ for all $i \in \{1, \ldots, n\}$. A *hitting set* of $\mathcal{S}$ is a subset $H \subseteq S$ such that $H \cap C_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$.

The **VertexCover** and **HittingSet** decision problems are defined as:

$$
\begin{aligned}
\textbf{VertexCover} &= \{\langle \mathcal{G}, n\rangle \mid \mathcal{G} \text{ is a graph which has a vertex cover of size at most } n \in \mathbb{N}_1\}\\
\textbf{HittingSet} &= \{\langle \mathcal{S}, m\rangle \mid \mathcal{S} \text{ is a subset collection with a hitting set of size at most } m \in \mathbb{N}_1\}
\end{aligned}
$$

(a) Prove that **VertexCover** $\leq_{\mathrm{p}}$ **HittingSet**.

(b) Prove that **HittingSet** is NP-complete. (You may use your result from part (a) and that it is known that VertexCover is NP-complete.)

Solution:

(a) Our goal is to prove that there exists a function f which convert the input of **VertexCover** into the input of **HittingSet** and that can be computed in polynomial time by a Turing Machine. Formally, we need to provide a function f so that $\langle \mathcal{G}, n\rangle \in$ **VertexCover** iff $f(\langle \mathcal{G}, n\rangle) \in$ **HittingSet** and $f \in O(|\langle \mathcal{G}, n\rangle|^k)$ for some integer $k > 0$.

To do so, let $\mathcal{E} := \{\{u, v\} | \langle u, v\rangle \in E\}$. Then it easy to see that

C is a *vertex cover* of $\mathcal{G}$ of size at most $n \Leftrightarrow C$ is an *hitting set* of $(V, \mathcal{E})$ of size at most n

Let now f be the function defined so that $f(\langle \mathcal{G}, n\rangle) = \langle \mathcal{E}, n\rangle$, it is easy to see from the observation above that $\langle \mathcal{G}, n\rangle \in$ **VertexCover** iff $f(\langle \mathcal{G}, n\rangle) \in$ **HittingSet** and $f \in O(|\langle \mathcal{G}, n\rangle|^2)$.

(b) We have to show that

HittingSet is NP**-hard:** that is, $X \leq_{\mathrm{p}}$ **HittingSet** $\forall X \in \mathrm{NP}$.
This can easily proven just by observing that **VertexCover** is NP−complete, that is, $X \leq_{\mathrm{p}}$ **VertexCover** for all $X \in \mathrm{NP}$. Thus, $X \leq_{\mathrm{p}}$ **VertexCover** $\leq_{\mathrm{p}}$ **HittingSet** $\forall X \in \mathrm{NP}$.

HittingSet $\in$ NP**:** We can apply *guess–and–check* to solve **HittingSet**. Indeed, we can test whether a subset H is an hitting set of size at most m for $\mathcal{S}$ by testing the not-emptiness of the intersection $H \cap C_i$ for every set $C_i \in \mathcal{C}$. This can surely be done in $O(|H| \sum_{i=1}^{|\mathcal{C}|} |C_i|)$. Observing that $|C_i| \leq |S|$, $|\mathcal{C}| \leq m$ and $|H| \leq |S|$, then it easy to see that such test can be performed in $O(m|S|^2)$ computations. That is, a Turing machine M can run such test in $O(|\langle \mathcal{S}, m\rangle|^3)$. Thus we can create a NTM M' that on input $\langle \mathcal{S}, m\rangle$ choose non-deterministically a set $H \subseteq S$ and test whether H is an hitting set for $\langle \mathcal{S}, m\rangle$. M' solves **HittingSet** in polynomial time.

Question 5 (Decidability, $4 + 10$ points)

Consider the problem of testing whether a given single-tape Turing machine ever writes a blank symbol over a non-blank symbol during the course of its computation, for any input string.

(a) Formulate this problem as a language.

(b) Show that the problem is undecidable.

Solution:

(a) The language can be described as

$$L = \{\langle M, w\rangle \mid M \text{ is a TM that writes a blank symbol over a non-blank one } ...\}.$$

(b) To show that such language is undecidable, we can argument as follows. Let suppose that L is decidable and let D be a decider for L. We can define a Turing Machine $\mathcal{D}$ as follows:

$\mathcal{D}$ = "On input $\langle M, w\rangle$

1. Create a Turing Machine M' so that:
   1.a Whenever M writes a blank symbol, it writes a non-blank symbol γ.
   1.b Apply the transition defined by blank symbols whenever γ is read.
   1.c Before accepting, we write γ and then we overwrite a blank.
2. If $D(\langle M', w\rangle)$ accepts, *accept*. *reject* otherwise."

It is easy to see that M' never writes a blank symbol unless M' accepts the input w. That is, $\mathcal{D}$ is a decider for A_{TM} which is clearly a contradiction.

Question 6 (Propositional Logic, $5 + 9$ points)

(a) Resolution is not a complete proof method. However, the *contradiction theorem* can be used to obtain a sound and complete method based on resolution for answering queries of the form "Does KB $\models \varphi$?".

Describe how this is done in general, i.e., to which set of clauses the resolution method is applied, and which outcome of the resolution method means that KB $\models \varphi$.

You may assume that KB is given as a set of clauses and φ as a conjunction of literals.

(b) Use the method described in part (a) to prove $\text{KB} \models P \wedge R$ for

$$\text{KB} = \{P \vee \neg Q, \quad P \vee Q \vee \neg R, \quad P \vee R, \quad Q \vee S, \quad R, \quad \neg R \vee S\}.$$

Solution:

(a) Discussed in class.

(b) Using contradiction theorem we have $\text{KB} \models P \vee R$ if and only if $\text{KB}' := \text{KB} \cup \{\neg(P \wedge R)\} \models \bot$. We first convert KB' into a clause set:

Formula (and equivalences)	**Clauses**
$P \vee \neg Q$	$\{P, \neg Q\}$
$P \vee Q \vee \neg R$	$\{P, Q, \neg R\}$
$P \vee R$	$\{P, R\}$
$Q \vee S$	$\{Q, S\}$
R	$\{R\}$
$\neg R \vee S$	$\{\neg R, S\}$
$\neg(P \wedge R) \equiv \neg P \vee \neg R$	$\{\neg P, \neg R\}$

That is, the clause set is

$$\Delta := \{\{P, \neg Q\}, \{P, Q, \neg R\}, \{P, R\}, \{Q, S\}, \{R\}, \{\neg R, S\}, \{\neg P, \neg R\}\}.$$

The following derivation turns into a contradiction:

$C_1 = \{P, \neg Q\}$	from Δ
$C_2 = \{P, Q, \neg R\}$	from Δ
$C_3 = \{P, \neg R\}$	from C_1 and C_2
$C_4 = \{R\}$	from Δ
$C_5 = \{P\}$	from C_3 and C_4
$C_6 = \{\neg P, \neg R\}$	from Δ
$C_7 = \{\neg R\}$	from C_6 and C_5
$C_8 = \square$	from C_4 and C_7

Question 7 (Propositional logic, $9 + 5$ points)

(a) Which of the following formulae are *satisfiable*? Which ones are *valid*? Which ones are *unsatisfiable*? For formulas belonging to several of these categories, please list *all* of them.

For all *satisfiable* cases, also provide a satisfying truth assignment. For the questions about validity and unsatisfiability, you do *not* need to justify your answers.

(*i*) $(A \vee \neg B) \rightarrow (A \wedge C)$

(*ii*) $(A \leftrightarrow B) \wedge (B \leftrightarrow \neg A)$

(*iii*) $(A \wedge B) \vee (\neg A \wedge \neg B)$

(*iv*) $(A \leftrightarrow B) \wedge (B \rightarrow \neg A)$

Solution:

(*i*) Satisfiable $(A \leftarrow \mathbf{T}, B \leftarrow \mathbf{T}, C \leftarrow \mathbf{T})$.

(*ii*) Unsatisfiable.

(*iii*) Satisfiable ($A \leftarrow \mathbf{T}, B \leftarrow \mathbf{T}$).

(*iv*) Satisfiable ($A \leftarrow \mathbf{F}, B \leftarrow \mathbf{F}$).

(b) Prove that

$$(A \wedge B) \to C \equiv A \to (B \to C)$$

by providing a sequence of logical equivalences that transforms the left-hand side into the right-hand side.

Solution: We can prove the equivalence as follows:

$$(A \wedge B) \to B \equiv A \to (B \to C)$$
$$\neg(A \wedge B) \vee C \equiv \neg A \vee (B \to C)$$
$$\neg A \vee \neg B \vee C \equiv \neg A \vee \neg B \vee C$$

Question 8 (Example of multiple choice question)

In which of the following cases is the logical formula to the left a *reasonable formalization* of the natural-language sentence to the right?

- ☐ $\forall x \forall y((\mathit{LivesIn}(x, y) \wedge \neg \mathit{EatsUp}(x)) \to \mathit{BadWeatherIn}(y))$ "Whenever someone who lives in some place does not eat up, the weather in that place will be bad."
- ☐ $\forall x \forall y(\mathit{Friend}(x, y) \wedge \mathit{Friend}(y, x))$ "Whenever A is a friend of B, B is a friend of A."
- ☐ $\forall x \forall y(\mathit{FatherOf}(x, \mathrm{me}) \wedge \mathit{DaughterOf}(y, x) \wedge \mathit{Female}(\mathrm{me})) \to (y = \mathrm{me})$ "If my father has a daughter and I am female, then that daughter is me."
- ☐ $\exists x \forall y \mathit{Father}(x, y)$ "Everybody has at least one father."
- ☐ $\mathit{DaughterOf}(\mathrm{me}, \mathit{Friend})$ "I am the daughter of my friend."

Solution: 1.