DD2371 Automata Theory
– Additional Exercises –

Dilian Gurov
Royal Institute of Technology – KTH
e–mail: dilian@csc.kth.se

1 Finite Automata and Regular Languages

1.1 Simple Constructions on Finite Automata

1. Give a deterministic finite automaton over the alphabet $\{a, b\}$ which accepts all strings containing no more than two consecutive occurrences of the same input letter. (For example, *abba* should be accepted but not *abaaab*.)

2. Convert the nondeterministic automaton given below to an equivalent deterministic one using the subset construction. Omit inaccessible states. Draw the graph of the resulting DFA.

	a	b
$\rightarrow$ q_1	$\{q_2\}$	$\emptyset$
$\rightarrow$ q_2 F	$\emptyset$	$\{q_1, q_3\}$
q_3 F	$\{q_2, q_3\}$	$\{q_1\}$

Solution: (presented as a table)

	a	b
$\rightarrow$ $\{q_1, q_2\}$ F	$\{q_2\}$	$\{q_1, q_3\}$
$\{q_1, q_3\}$ F	$\{q_2, q_3\}$	$\{q_1\}$
$\{q_2, q_3\}$ F	$\{q_2, q_3\}$	$\{q_1, q_3\}$
$\{q_1\}$	$\{q_2\}$	$\emptyset$
$\{q_2\}$ F	$\emptyset$	$\{q_1, q_3\}$
$\emptyset$	$\emptyset$	$\emptyset$

3. Convert the nondeterministic automaton given below to an equivalent deterministic one using the subset construction. Omit inaccessible states. Draw the graph of the resulting DFA.

	a	b	
$\rightarrow$ q_1	q_1, q_2	q_3	F
q_2	$-$	q_4	F
$\rightarrow$ q_3	q_4	$-$	
q_4	$-$	q_1, q_4	

Solution: (given as a table)

	a	b	
$\to$ $\{q_1, q_3\}$	$\{q_1, q_2, q_4\}$	$\{q_3\}$	F
$\{q_1, q_2, q_4\}$	$\{q_1, q_2\}$	$\{q_1, q_3, q_4\}$	F
$\{q_1, q_3, q_4\}$	$\{q_1, q_2, q_4\}$	$\{q_1, q_3, q_4\}$	F
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_3, q_4\}$	F
$\{q_3, q_4\}$	$\{q_4\}$	$\{q_1, q_4\}$	
$\{q_1, q_4\}$	$\{q_1, q_2\}$	$\{q_1, q_3, q_4\}$	F
$\{q_4\}$	$\{\}$	$\{q_1, q_4\}$	
$\{q_3\}$	$\{q_4\}$	$\{\}$	
$\{\}$	$\{\}$	$\{\}$	

4. Convert the nondeterministic automaton given below to an equivalent deterministic one using the subset construction (textbook Lecture 6). Omit inaccessible states. Draw the graph of the resulting DFA.

	a	b
$\to$ q_0	$\{q_2\}$	$\emptyset$
$\to$ q_1 F	$\{q_0, q_2\}$	$\emptyset$
q_2 F	$\emptyset$	$\{q_1, q_2\}$

Solution: (as a table)

	a	b
$\to$ $\{q_0, q_1\}$ F	$\{q_0, q_2\}$	$\emptyset$
$\{q_0, q_2\}$ F	$\{q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$ F	$\{q_0, q_2\}$	$\{q_1, q_2\}$
$\{q_2\}$ F	$\emptyset$	$\{q_1, q_2\}$
$\emptyset$	$\emptyset$	$\emptyset$

5. Convert the nondeterministic automaton given below to an equivalent deterministic one using the subset construction, textbook Lecture 6. Omit inaccessible states. Draw the graph of the resulting DFA.

	a	b
$\to$ q_0	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_0, q_1\}$	$\{q_0\}$
q_2 F	$\emptyset$	$\{q_1, q_2\}$

Solution: Easy: has 8 states, and hence no inaccessible ones!

6. Give a (as simple as possible) nondeterministic finite automaton for the language defined by the regular expression $ab^* + a(ba)^*$. Apply the subset construction to obtain an equivalent deterministic automaton. Draw the graph of this DFA, omitting inaccessible states.

7. Give a DFA for the language defined by the regular expression a^*a, and another one for $a(ba)^*$. The union of the two DFAs defines a nondeterministic FA for the language $a^*a + a(ba)^*$.

   (a) Apply the subset construction to this NFA to produce a DFA for this language. Omit the inaccessible states. Draw the graph of the resulting DFA.

   (b) Is this DFA minimal? If not, which states are equivalent?

1.2 Closure Properties of Regular Languages

8. Show that regular languages are closed under doubling: If language L is regular, then so is the language $L_2 \overset{\text{def}}{=} \{\textbf{two}\ x \mid x \in L\}$, where string doubling is defined inductively by $\textbf{two}\ \epsilon \overset{\text{def}}{=} \epsilon$ and $\textbf{two}\ xa \overset{\text{def}}{=} (\textbf{two}\ x) \cdot aa$.

   Solution: Here is a standard solution using finite automata; alternative solutions exist using regular expressions or homomorphisms.

   Let L be regular. Then there is an NFA $N = (Q_N, \Sigma, \Delta_N, S_N, F_N)$ such that $L(N) = L$. Define another NFA, N_2, as follows: start with N and replace every edge $q \xrightarrow{a} q'$ with two edges $q \xrightarrow{a} q''$ and $q'' \xrightarrow{a} q'$ by inserting (for each edge!) a new state q''. We can formalize this idea by taking as states of N_2 the states of N plus the edges of N, the latter represented for example as the set of triples $\{(q_N, a, q'_N) \mid q_N \in Q_N,\ q'_N \in \Delta_N(q_N, a)\}$. So, we can define N_2 as follows:

   - $Q_{N_2} \overset{\text{def}}{=} Q_N \cup \{(q_N, a, q'_N) \mid q_N \in Q_N,\ q'_N \in \Delta_N(q_N, a)\}$
   - Δ_{N_2} is given by the two defining equations:
     $\Delta_{N_2}(q_N, a) \overset{\text{def}}{=} Q_N \cup \{(q_N, a, q'_N) \mid q'_N \in \Delta_N(q_N, a)\}$ and
     $\Delta_{N_2}((q_N, b, q'_N), a) \overset{\text{def}}{=}$ if $a = b$ then $\{q'_N\}$ else $\emptyset$
   - $S_{N_2} \overset{\text{def}}{=} S_N$
   - $F_{N_2} \overset{\text{def}}{=} F_N$

   It is straightforward to show that, for the so constructed NFA, $L(N_2) = L_2$, thus implying that L_2 is regular. Hence, regular languages are closed under doubling.

9. Let $A \subseteq \Sigma^*$ be a language. We define its prefix closure as:

$$\textbf{pref}\ A = \{x \in \Sigma^* \mid \exists y \in \Sigma^*.\ x \cdot y \in A\}$$

   Prove that regular languages are closed under prefixing: if language A is regular, then so is **pref** A.

   Solution: Let $A \subseteq \Sigma^*$ be regular. Then there must be a DFA $M_A = (Q, \Sigma, \delta, s, F)$ such that $L(M_A) = A$. Now, based on M_A, define another automaton $M = (Q, \Sigma, \delta, s, F')$ so that:

$$F' = \left\{q \in Q \mid \exists y \in \Sigma^*.\ \hat{\delta}(q, y) \in F\right\}$$

   We will show that this automaton accepts the language **pref** A, and hence that **pref** A is regular.

   Claim. $L(M) =$ **pref** A

   Proof.

$$\begin{array}{rcll} x \in L(M) & \Leftrightarrow & \hat{\delta}(s, x) \in F' & \{\text{def. } L(M)\} \\ & \Leftrightarrow & \exists y \in \Sigma^*.\ \hat{\delta}(\hat{\delta}(s, x), y) \in F & \{\text{def. } F'\} \\ & \Leftrightarrow & \exists y \in \Sigma^*.\ \hat{\delta}(s, x \cdot y) \in F & \{\text{HW 1.3, page 301}\} \\ & \Leftrightarrow & \exists y \in \Sigma^*.\ x \cdot y \in A & \{L(M_A) = A\} \\ & \Leftrightarrow & x \in \textbf{pref}\ A & \{\text{def. } \textbf{pref}\ A\} \end{array}$$

10. Consider the following unary operation on languages:

$$\mathit{min}(L) = \{x \in L \mid \text{no proper prefix of } x \text{ is in } L\}$$

   Prove that regular languages are closed under this operation; that is, prove that if language A is regular, then so is $\mathit{min}(A)$.

   Solution: Assume A is regular. (For simplicity, we shall also assume $\epsilon \notin A$.) Then there is a DFA M_A accepting A. We construct another DFA M'_A from M_A by adding two new states q_F and q_G, making

q_F the only accepting state of M'_A, letting $\delta'(q_F, a) = q_G$ and $\delta'(q_G, a) = q_G$ for every $a \in \Sigma$, and by re-directing all edges pointing to a final state in M_A to point to q_F. We can then show $L(M'_A) = \mathit{min}(A)$ as follows:

$$\begin{array}{rll}
x \in L(M'_A) & \Leftrightarrow \ \hat{\delta}'(s, x) = q_F & \{\text{Def. acceptance and } M'_A\} \\
& \Leftrightarrow \ \hat{\delta}(s, x) \in F \text{ and for no proper prefix } y \text{ of } x,\ \hat{\delta}(s, y) \in F & \{\text{Def. } M'_A\} \\
& \Leftrightarrow \ x \in A \text{ and no proper prefix of } x \text{ is in } A & \{\text{Def. acceptance}\} \\
& \Leftrightarrow \ x \in \mathit{min}(A) & \{\text{Def. } \mathit{min}(A)\}
\end{array}$$

thus proving that $\mathit{min}(A)$ is regular.

1.3 DFA State Minimization

11. For the deterministic automaton given below, apply the minimization algorithm of Lecture 14 to compute the equvalence classes of the collapsing relation $\approx$ defined in Lecture 13. Show clearly the computation steps. List the equivalence classes, and apply the quotient construction to derive a minimized automaton. Draw its graph.

		a	b	
$\rightarrow$	q_0	q_0	q_1	
	q_1	q_2	q_3	
	q_2	q_2	q_3	
	q_3	q_2	q_4	F
	q_4	q_0	q_1	

Solution: (incomplete) After applying the minimization algorithm we obtain that $q_0 \approx q_4$ and $q_1 \approx q_2$. Thus we obtain the following minimized automaton (given as a table).

		a	b	
$\rightarrow$	$\{q_0, q_4\}$	$\{q_0, q_4\}$	$\{q_1, q_2\}$	
	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_3\}$	
	$\{q_3\}$	$\{q_1, q_2\}$	$\{q_0, q_4\}$	F

12. For the deterministic automaton given below, apply the minimization algorithm of Lecture 14 to compute the equvalence classes of the collapsing relation $\approx$ defined in Lecture 13. Show clearly the computation steps. List the equivalence classes, and apply the quotient construction to derive a minimized automaton. Draw its graph.

			a	b
$\rightarrow$	q_1		q_3	q_8
	q_2	F	q_3	q_1
	q_3		q_8	q_2
	q_4	F	q_5	q_6
	q_5		q_6	q_2
	q_6		q_7	q_8
	q_7		q_6	q_4
	q_8		q_5	q_8

Solution: With the minimization algorithm we establish that $q_1 \approx q_6 \approx q_8$, $q_2 \approx q_4$ and $q_3 \approx q_5 \approx q_7$. The resulting quotient automaton, presented as a table, is:

			a	b
$\rightarrow$	$\{q_1, q_6, q_8\}$		$\{q_3, q_5, q_7\}$	$\{q_1, q_6, q_8\}$
	$\{q_3, q_5, q_7\}$		$\{q_1, q_6, q_8\}$	$\{q_2, q_4\}$
	$\{q_2, q_4\}$	F	$\{q_3, q_5, q_7\}$	$\{q_1, q_6, q_8\}$

13. For the deterministic automaton given below, apply the minimization algorithm of Lecture 14 of the textbook to compute the equvalence classes of the collapsing relation $\approx$ defined in Lecture 13.

	a	b
$\rightarrow$ q_0 F	q_1	q_3
q_1	q_2	q_3
q_2 F	q_5	q_2
q_3	q_4	q_1
q_4 F	q_5	q_4
q_5	q_5	q_5

(a) Show clearly the computation steps (use tables).
(b) List the computed equivalence classes.
(c) Apply the quotient construction of Lecture 13 to derive the minimized automaton. Draw its graph.

Solution: As a table:

	a	b
$\rightarrow$ $\{q_0\}$ F	$\{q_1, q_3\}$	$\{q_1, q_3\}$
$\{q_1, q_3\}$	$\{q_2, q_4\}$	$\{q_1, q_3\}$
$\{q_2, q_4\}$ F	$\{q_5\}$	$\{q_2, q_4\}$
$\{q_5\}$	$\{q_5\}$	$\{q_5\}$

14. For the deterministic automaton given below, apply the minimization algorithm of textbook Lecture 14 to compute the equvalence classes of the collapsing relation $\approx$ defined in textbook Lecture 13.

	a	b
$\rightarrow$ q_0	q_2	q_0
q_1	q_0	q_2
q_2 F	q_4	q_3
q_3	q_0	q_4
q_4 F	q_4	q_1

(a) Show clearly the computation steps (use tables).
(b) List the computed equivalence classes.
Solution: These are $\{q_0\}$, $\{q_1, q_3\}$ and $\{q_2, q_4\}$.
(c) Apply the quotient construction of textbook Lecture 13 to derive the minimized automaton. Draw its graph.
Solution: (as a table)

	a	b
$\rightarrow$ $\{q_0\}$	$\{q_2, q_4\}$	$\{q_0\}$
$\{q_1, q_3\}$	$\{q_0\}$	$\{q_2, q_4\}$
$\{q_2, q_4\}$ F	$\{q_2, q_4\}$	$\{q_1, q_3\}$

1.4 Myhill–Nerode Relations and Theorem

15. Consider the language L defined by the regular expression $(a^* + ba)^*$. Describe the equivalence classes of $\{a, b\}^*$ w.r.t. the Myhill–Nerode relation $\equiv_L$ defined by: (cf. equation (16.1) on page 97)

$$x_1 \equiv_L x_2 \overset{\text{def}}{\Longleftrightarrow} \forall y \in \Sigma^*.(x_1 \cdot y \in L \Leftrightarrow x_2 \cdot y \in L)$$

Present these equivalence classes through regular expressions. Use $\equiv_L$ to construct a minimal automaton $M_{\equiv_L}$ (cf. page 91) for the language L, and draw the graph of the automaton.

Solution: The states of the quotient automaton are:

$$\Sigma^*/_{\equiv_L} = \{L((a + ba)^*),\ L((a + ba)^*b),\ L((a + ba)^*bb(a + b)^*)\}$$

16. Recall the Myhill–Nerode Theorem, textbook Lecture 16, with the equivalence relation $\equiv_A$ on strings defined in equation (16.1). The latter gives rise to the following technique for proving that the minimal DFA for a given regular language A over Σ has *at least* k states:

    Identify k strings $x_1, \ldots, x_k$ over Σ, for which you can show that:
    $x_i \not\equiv_A x_j$ whenever $i \neq j$.

    Then $\equiv_A$ has at least k equivalence classes, and hence the desired result.

    Now, let m be an arbitrary but fixed even number. Consider the language B over $\{a, b\}$ consisting of all strings of length at least m which have an equal number of a's and b's in the last m positions. Use the technique described above to show that the minimal DFA for language B has at least $2^{\frac{m}{2}}$ states.
    Hint: Consider the strings of length $\frac{m}{2}$.

    Solution: Consider the set X of strings of length $\frac{m}{2}$ over $\{a, b\}$. Let $x, y \in X$ so that $x \neq y$. Let x', y' be the shortest suffices of x and y respectively, such that $|x'| = |y'|$ and $x' \neq y'$. Then obviously $\sharp a(x') \neq \sharp a(y')$. Now let $z = a^r b^s$, where $r = \frac{m}{2} - \sharp a(x')$ and $s = m - (|x'| + r)$. Then $x \cdot z \in B$ while $y \cdot z \notin B$, and hence $x' \not\equiv_B y'$. Since X has $2^{\frac{m}{2}}$ elements, $\equiv_B$ has at least $2^{\frac{m}{2}}$ equivalence classes, and therefore the minimal DFA for language B has at least $2^{\frac{m}{2}}$ states.

1.5 Proving Non–regularity of a Language

17. Apply the Pumping Lemma – in contra–positive form, as a game with the Demon – to show that the following language:

$$A = \{a^n \mid n \text{ is a power of } 2\}$$

    is not regular.

    Solution: One possible solution is:

    (D) Demon picks k.
    (W) We pick $x = \epsilon$, $y = a^{2^k}$ and $z = a^{2^k}$. Then $xyz = a^{2^{k+1}}$ and $|y| > k$.
    (D) Demon picks u, v and w so that $uvw = y = a^{2^k}$ and $v \neq \epsilon$.
    (W) We pick $i = 2$.

    Then $xuv^iwz = a^{2^{k+1}+l}$ for some $0 < l \leq 2^k$. Since $l < 2^{k+1}$ we have $2^{k+1} < 2^{k+1} + l < 2^{k+2}$, and hence $xuv^iwz \notin A$. We have a winning strategy, and A is therefore not regular.

2 Pushdown Automata and Context-Free Languages

2.1 Combined Problems

1. Consider the language:

$$L \overset{\text{def}}{=} \left\{ x \cdot y \in \{a,b\}^+ \mid y = \textbf{rev } x \right\}$$

where **rev** x denotes the reverse string of x (cf. HW 2.2, page 302).

(a) Use the Pumping Lemma to prove that L is *not* regular.

Solution: As a game with the Demon (cf. Lecture 11):

- Demon picks $k > 0$.
- We pick for example $x = \epsilon$, $y = a^k$, $z = bba^k$, and we have $xyz = a^k bba^k \in L$ and $|y| \geq k$.
- Demon picks $uvw = y = a^k$, $v \neq \epsilon$.
- We pick for example $i = 0$.

Then $xuv^iwz = uwz = a^j bba^k$ for some $j < k$, and hence $xuv^iwz \notin A$. We have a winning strategy, and L is therefore not regular.

(b) Give a context–free grammar G for L.

Solution: $S \to aa \mid bb \mid aSa \mid bSb$

(c) Prove your grammar correct (cf. Lecture 20): that is, prove $L = L(G)$.

Solution: We have to prove:

$$\forall x \in \{a,b\}^+.\ (S \xrightarrow{*}_G x \Leftrightarrow x \in L)$$

We show the two directions of the equivalence separately.

($\Rightarrow$) By induction on the length of the derivation of x.
Basis Holds vacuously, since $S \xrightarrow{0}_G x$ is false: $x = S$ is impossible since $S \notin \{a,b\}^+$.
Induction Step Assume $x' \in L$ for all x' such that $S \xrightarrow{n}_G x'$ (induction hypothesis).
Let $S \xrightarrow{n+1}_G x$. Then, we must have $S \xrightarrow{1}_G \gamma$ and $\gamma \xrightarrow{n}_G x$ for some $\gamma \in \{a,b,S\}^+$. But then γ can only be aa or bb or aSa or bSb. The first two cases imply $n = 0$ and $x = \gamma$, and then obviously $x \in L$. In the case $\gamma = aSa$, it must be that $x = ax'a$ and $S \xrightarrow{n}_G x'$ for some x'. From the induction hypothesis, we have $x' \in L$. But $x = ax'a$, and therefore also $x \in L$.
The case $\gamma = bSb$ is similar.

($\Leftarrow$) By induction on $|x|$, which is even and positive.
Basis $|x| = 2$, then $x \in L$ implies that x is either aa or bb. In both cases $S \xrightarrow{1}_G x$ and therefore $S \xrightarrow{*}_G x$.
Induction Step Assume $S \xrightarrow{*}_G x'$ for all $x' \in L$ such that $|x'| = n$ (induction hypothesis).
Let $|x| = n + 2$, and let $x \in L$. It must be that either $x = ax'a$ or $x = bx'b$ for some x' such that $x' \in L$ and $|x'| = n$. From the induction hypothesis, $S \xrightarrow{*}_G x'$. Then, in the case $x = ax'a$ we also have $aSa \xrightarrow{*}_G ax'a = x$, and since $S \xrightarrow{1}_G aSa$, then $S \xrightarrow{*}_G x$.
The case $x = bx'b$ is similar.

(d) Give an NPDA for L.

Solution: One possibility is to put G in GNF:

$$S \to aA \mid bB \mid aSA \mid bSB$$
$$A \to a$$
$$B \to b$$

and then construct the NPDA canonically (cf. Lecture 24):

$$Q \overset{\text{def}}{=} \{q\}$$
$$\Sigma \overset{\text{def}}{=} \{a,b\}$$
$$\Gamma \overset{\text{def}}{=} \{S,A,B\}$$

$$\delta \overset{\text{def}}{=} \begin{array}{l} \langle q, S\rangle \overset{a}{\hookrightarrow} \langle q, SA\rangle \\ \langle q, S\rangle \overset{a}{\hookrightarrow} \langle q, A\rangle \\ \langle q, S\rangle \overset{b}{\hookrightarrow} \langle q, SB\rangle \\ \langle q, S\rangle \overset{b}{\hookrightarrow} \langle q, B\rangle \\ \langle q, A\rangle \overset{a}{\hookrightarrow} \langle q, \epsilon\rangle \\ \langle q, B\rangle \overset{b}{\hookrightarrow} \langle q, \epsilon\rangle \end{array}$$

$$s \overset{\text{def}}{=} q$$
$$\perp \overset{\text{def}}{=} S$$

2. Consider the language:

$$A = \left\{a^k b^l a^m \mid m = k + l\right\}$$

(a) Use the closure properties of regular languages to show that A is not regular.

Solution: The language $L(b^*a^*)$ is regular, but $A \cap L(b^*a^*) = \{b^n a^n \mid n \geq 0\}$, as we already know, is not regular. Since regular languages are closed under intersection, A is not regular.

(b) Give a context–free grammar G generating A.

Solution: One possibility is:

$$\begin{array}{lcl} S & \rightarrow & aSa \mid B \\ B & \rightarrow & \epsilon \mid bBa \end{array}$$

(c) Prove your grammar correct; that is, prove $S \xrightarrow{+}_G x \Leftrightarrow x \in A$.

Solution: The proof is standard, and is made easy by the fact that we already know that $B \xrightarrow{+}_G x \Leftrightarrow x \in \{b^n a^n \mid n \geq 0\}$.

(d) Construct an NPDA accepting $A - \{\epsilon\}$ on empty stack. Explain your choice of productions.

Solution: One possibility is to build an NPDA with three states, having the following productions:

$$\begin{array}{lll} \langle q_0, \perp\rangle \overset{a}{\hookrightarrow} \langle q_0, C\rangle & \langle q_1, C\rangle \overset{a}{\hookrightarrow} \langle q_2, \varepsilon\rangle & \langle q_2, C\rangle \overset{a}{\hookrightarrow} \langle q_2, \varepsilon\rangle \\ \langle q_0, \perp\rangle \overset{a}{\hookrightarrow} \langle q_2, C\rangle & \langle q_1, C\rangle \overset{b}{\hookrightarrow} \langle q_1, CC\rangle & \\ \langle q_0, \perp\rangle \overset{b}{\hookrightarrow} \langle q_1, C\rangle & & \\ \langle q_0, C\rangle \overset{a}{\hookrightarrow} \langle q_0, CC\rangle & & \\ \langle q_0, C\rangle \overset{a}{\hookrightarrow} \langle q_2, CC\rangle & & \\ \langle q_0, C\rangle \overset{b}{\hookrightarrow} \langle q_1, CC\rangle & & \end{array}$$

The first state counts the initial a's, the second state counts the b's which follow, and the third state checks for the sum. In addition, the first state can nondeterministically decide that no b's are going to come and that exactly half of the a's have been read.

3. Consider the language:

$$A = \{x \in \{a, b\}^* \mid \sharp a(x) < \sharp b(x)\}$$

(a) Give a context–free grammar G for A. Explain your choice of productions.

Solution: There are many possible solutions. One way of looking at the strings of the language is to devide these into the ones which have exactly one occurrence of b more than occurrences of a, and those that have more. A string is in the first group exactly when it can be represented as a string of the shape $e_1 \cdot b \cdot e_2$, where e_1 and e_2 are strings with an equal number of a's and b's. Thus, e_1 and e_2 can be produced by the grammar $E \rightarrow \epsilon \mid aEb \mid bEa \mid EE$. A string is in the second group exactly when it is the concatenation of two strings of A. So we arrive at the following grammar:

$$\begin{array}{lcl} S & \rightarrow & EbE \mid SS \\ E & \rightarrow & \epsilon \mid aEb \mid bEa \mid EE \end{array}$$

(b) Construct an NPDA accepting A on empty stack. Explain its workings.

Solution: Again, there is a number of good solutions. One elegant solution using ε-transitions (proposed by one of the students at the exam) is based on the observation that if we use the "standard" productions for comparing occurrences:

$$\begin{array}{ll}
\langle q, \perp\rangle \overset{a}{\hookrightarrow} \langle q, A\perp\rangle & \langle q, \perp\rangle \overset{b}{\hookrightarrow} \langle q, B\perp\rangle \\
\langle q, A\rangle \overset{a}{\hookrightarrow} \langle q, AA\rangle & \langle q, B\rangle \overset{b}{\hookrightarrow} \langle q, BB\rangle \\
\langle q, B\rangle \overset{a}{\hookrightarrow} \langle q, \varepsilon\rangle & \langle q, A\rangle \overset{b}{\hookrightarrow} \langle q, \varepsilon\rangle
\end{array}$$

then a string is in A exactly when after reading it the stack contains only B's (on top of $\perp$). Note that there must be at least one such B. So, we can obtain the desired behaviour by adding two more productions:

$$\begin{array}{l}
\langle q, \perp\rangle \overset{b}{\hookrightarrow} \langle q, B\rangle \\
\langle q, B\rangle \overset{\varepsilon}{\hookrightarrow} \langle q, \varepsilon\rangle
\end{array}$$

4. Consider the language:

$$A = \{a^m b^n \mid m \leq n\}$$

(a) Use the Pumping Lemma for regular languages (as a game with a Deamon) to prove that A is not regular.

(b) Refer to the closure properties of context–free languages to argue that A is context–free. That is, represent A as the result of some operation(s) on context–free languages (which we already know to be context–free) under which CFLs are closed.

Solution: $A = B \cdot C$ for $B \overset{\text{def}}{=} \{a^m b^m \mid m \geq 0\}$ and $C \overset{\text{def}}{=} \{b^n \mid n \geq 0\}$, both of which we know to be context–free, and we know CFLs to be closed under language concatenation.

(c) Give a context–free grammar G generating A.

Solution: One possibility is the grammar $S \to \epsilon \mid aSb \mid Sb$.

(d) Prove your grammar correct. You are allowed to reuse results proved in class.

(e) Construct an NPDA accepting $A - \{\epsilon\}$ on empty stack. Explain your choice of productions.

Solution: One solution is a NPDA with one control state q and productions:

$$\begin{array}{l}
\langle q, S\rangle \overset{a}{\hookrightarrow} \langle q, SA\rangle \\
\langle q, S\rangle \overset{a}{\hookrightarrow} \langle q, A\rangle \\
\langle q, A\rangle \overset{b}{\hookrightarrow} \langle q, A\rangle \\
\langle q, A\rangle \overset{b}{\hookrightarrow} \langle q, \epsilon\rangle
\end{array}$$

The automaton uses the first production for reading all initial a's but the last, then guesses the last a and uses the second production to get rid of the S at the top of the stack. The stack now has as many A's as the a's read so far. The automaton guesses the number of b's in excess of a's in the string, and uses the third production that many times. The stack now has as many A's as there are remaining (unread) b's. The automaton uses production four to empty the stack upon reading the whole string.

5. Consider the language:

$$A = \left\{a^k b^l a^m \mid l = k + m\right\}$$

(a) Refer to the closure properties of regular languages to argue that A is not regular.

Solution: The regular lanuguages are closed under intersection, but $A \cap L(a^* b^*)$ equals $\{a^n b^n \mid n \geq 0\}$ which is not regular, hence A cannot be regular.

(b) Refer to the closure properties of context–free languages to argue that A is context–free.

Solution: The context–free languages are closed under language concatenation, and since A can be represented as the concatenation of the context–free languages $\left\{a^k b^k \mid k \geq 0\right\}$ and $\{b^m a^m \mid m \geq 0\}$, A must be context–free.

(c) Give a context–free grammar G generating A.
Solution: From the previous observation, and the construction on grammars we used to show this closure property, we directly obtain the grammar:
$S \to AB$
$A \to \epsilon \mid aAb$
$B \to \epsilon \mid bBa$

(d) Construct an NPDA accepting $A - \{\epsilon\}$ on empty stack. Explain your choice of productions.

6. Consider the language:

$$L \overset{\text{def}}{=} \{x \in \{a,b,c\}^* \mid \sharp a(x) + \sharp b(x) = \sharp c(x)\}$$

(a) Show that L is not regular.
(b) Give a context–free grammar for L. Prove your grammar correct.
(c) Construct an NPDA accepting L (on empty stack).

7. Consider the language:

$$L = \{a^m b^n \mid m \neq n\}$$

(a) Give a simple argument for L being context–free.
(Hint: you could use the closure properties of CFLs.)
(b) Construct an NPDA (possibly with ϵ–transitions) accepting L on empty stack. Explain its workings.

2.2 Chomsky–Schützenberger Theorem

8. Recall the Chomsky–Schützenberger Theorem, textbook Supplementary Lecture G. Show how this Theorem applies to the context–free language PAL of even–length palindromes over $\{a,b\}$, by identifying a suitable number n, a regular language R, and a homomorphism (renaming) h.

Solution: PAL is equal to the language $h(\text{PAREN}_n \cap R)$ for $n = 2$, $R = L(([_1 + [_2)^* (]_1 +]_2)^*)$ and h renaming $[_1$ and $]_1$ to a and $[_2$ and $]_2$ to b.

9. Recall the Chomsky–Schützenberger Theorem (textbook Supplementary Lecture G). Show how this theorem applies to the context–free language

$$A \overset{\text{def}}{=} \{x \in \{a,b\}^* \mid \sharp a(x) = \sharp b(x)\}$$

over the alphabet $\Sigma = \{a,b\}$, by identifying:

- a suitable natural number n,
- a regular language R over the alphabet Σ_n of the n–th balanced parentheses language, and
- a homomorphism $h : \Sigma_n \to \Sigma^*$,

for which you argue that $A = h(\text{PAREN}_n \cap R)$ holds.

Solution: Recalling that A is generated by the grammar $S \to \epsilon \mid aSb \mid bSa \mid SS$ it is easy to see that $A = h(\text{PAREN}_n \cap R)$ holds for $n = 2$, $R = (\Sigma_n)^*$ and h defined by $h([_1) = a$, $h(]_1) = b$, $h([_2) = b$ and $h(]_2) = a$.

2.3 Proving Non–context–freeness of a Language

10. Apply the Pumping Lemma for context–free languages (as a game with the Demon) to show that the language:

$$A = \left\{a^n b^n a^j \mid j \leq n\right\}$$

is not context–free.

Solution:

– Demon picks an arbitrary $k \geq 0$.

+ We pick $z = a^k b^k a^k$ which is in A.

– Demon picks u, v, w, x, y such that $z = uvwxy$, $|vx| > 0$, $|vwx| \leq k$.

+ If $vwx = a^l b^m$ for some $l, m \geq 0$, we pick $i = 0$. Otherwise we pick $i = 2$.

Since $|vwx| \leq k$, vwx has either the shape $a^l b^m$ or $b^m a^l$ for some l, m such that $l + m \leq k$. In the first case xv^0wx^0y must be of the shape $a^p b^q a^k$ for some p, q such that $p + q < 2k$, and thus is not in A. In the second case, xv^2wx^2y will either not be in $L(a^*b^*a^*)$ at all, and thus not in A, or else xv^2wx^2y must be of the shape $a^k b^p a^q$ for some p, q such that $p + q > 2k$, and thus not in A. Hence, we win the game in all cases, which shows that A is not context–free.

2.4 Other Problems

11. A context–free grammar $G = (N, \Sigma, P, S)$ is called *strongly right–linear* (or SRLG for short) if all its productions are of shape $A \to aB$ or $A \to \epsilon$. Prove that SRLGs generate precisely the regular languages.

Hint: Define appropriate transformations between SRLGs and Finite Automata – one for each direction! – and prove that these transformations are language preserving. State and prove appropriate lemmas where needed to structure the proofs.

Solution: In two parts: the first part shows that the languages generated by SRLGs are regular, while the second shows that every regular language is the language of some SRLG.

a) Let $G = (N, \Sigma, P, S)$ be a SRLG. Define the NFA $N_G \overset{\text{def}}{=} (N, \Sigma, \Delta, \{S\}, F)$ where we define $\Delta(X, a) \overset{\text{def}}{=} \{Y \in N \mid (X \to aY) \in P\}$ and $F \overset{\text{def}}{=} \{X \in N \mid (X \to \epsilon) \in P\}$. Then:

$$\begin{array}{rcll} x \in L(N_G) & \Leftrightarrow & \hat{\Delta}(\{S\}, x) \cap F \neq \emptyset & \{\text{Def. } L(N)\} \\ & \Leftrightarrow & \{X \in N \mid S \to_G^* xX\} \cap F \neq \emptyset & \{\text{Lemma A}\} \\ & \Leftrightarrow & \exists X \in N.\, (S \to_G^* xX \wedge X \to_G \epsilon) & \{\text{Def. } F\} \\ & \Leftrightarrow & S \to_G^+ x & \{\text{Def. } \to_G^*\} \\ & \Leftrightarrow & x \in L(G) & \{\text{Def. } L(G)\} \end{array}$$

So $L(G) = L(N_G)$ and hence $L(G)$ is regular. In the proof, Lemma A states that

$$\hat{\Delta}(\{Y\}, x) = \{X \in N \mid Y \to_G^* xX\}$$

which is proved by induction on the structure of x as follows.
Base case.

$$\begin{array}{rcll} \hat{\Delta}(\{Y\}, \epsilon) & = & \{Y\} & \left\{\text{Def. } \hat{\Delta}\right\} \\ & = & \{X \in N \mid Y \to_G^* X\} & \{\text{Def. SRLG}\} \end{array}$$

Induction. Assume the Lemma holds for x (the ind. hyp.); we show that it then also holds for xa.

$$\begin{array}{rcll} \hat{\Delta}(\{Y\}, xa) & = & \bigcup_{X \in \hat{\Delta}(\{Y\}, x)} \Delta(X, a) & \left\{\text{Def. } \hat{\Delta}\right\} \\ & = & \bigcup_{X \in \{X \in N \mid Y \to_G^* xX\}} \Delta(X, a) & \{\text{Ind. hyp.}\} \\ & = & \bigcup_{X \in \{X \in N \mid Y \to_G^* xX\}} \{Z \in N \mid (X \to aZ) \in P\} & \{\text{Def. } \Delta\} \\ & = & \{X \in N \mid Y \to_G^* xaX\} & \{\text{Def. } \to_G^*\} \end{array}$$

b) This part is similar to the previous one and is only sketched here. Let A be a regular language. Then there is a DFA $M_A = (Q, \Sigma, \delta, s, F)$ such that $L(M_A) = A$. We define the SLRG $G_A \overset{\text{def}}{=} (Q, \Sigma, P, s)$ where $P \overset{\text{def}}{=} \{q_1 \to aq_2 \mid \delta(q_1, a) = q_2\} \cup \{q \to \epsilon \mid q \in F\}$. We then show $x \in L(G_A) \Leftrightarrow x \in L(M_A) = A$, the proof of which is best structured by proving and using Lemma B:

$$q_1 \to_{G_A}^* xq_2 \Leftrightarrow \hat{\delta}(q_1, x) = q_2$$

12. A context–free grammar $G = (N, \Sigma, P, S)$ is called *strongly right–linear* (or SRLG for short) if all its productions are of shape $A \to aB$ or $A \to \epsilon$. Let us call a SRLG *deterministic* if for each non–terminal $A \in N$ and terminal $a \in \Sigma$ there is exactly one production of shape $A \to aB$ (while productions of shape $A \to \epsilon$ are optional).

    Define a *complement construction* on deterministic SRLGs. That is, for deterministic SRLGs G define the complement $\overline{G}$ as a deterministic SRLG for which you show that $L(\overline{G}) = \overline{L(G)}$.

    Solution: Recall that there is a one–to–one corespondence beween DFAs and SRLGs, and recall the complement construction on DFAs. Let $G = (N, \Sigma, P, S)$ be a deterministic SRLG. We define the complement of G as the deterministic SRLG $\overline{G} \stackrel{\text{def}}{=} (N, \Sigma, \overline{P}, S)$ where $\overline{P} \stackrel{\text{def}}{=} \{A \to aB \mid A \to aB \in P\} \cup \{A \to \epsilon \mid A \to \epsilon \notin P\}$ is the set of productions of $\overline{G}$. Then,

$$\begin{array}{rcll}
x \in L(\overline{G}) & \Leftrightarrow & S \to^{+}_{\overline{G}} x & \{\text{Def. } L(G)\} \\
& \Leftrightarrow & \exists! A \in N.\,(S \to^{+}_{\overline{G}} xA \wedge A \to \epsilon \in \overline{P}) & \left\{\overline{G} \text{ a deterministic SRLG}\right\} \\
& \Leftrightarrow & \exists! A \in N.\,(S \to^{+}_{G} xA \wedge A \to \epsilon \notin P) & \left\{\text{Def. } \overline{G}\right\} \\
& \Leftrightarrow & \text{not } S \to^{+}_{G} x & \{G \text{ a deterministic SRLG}\} \\
& \Leftrightarrow & x \notin L(G) & \{\text{Def. } L(G)\} \\
& \Leftrightarrow & x \in \overline{L(G)} & \{\text{Set theory}\}
\end{array}$$

    and therefore $L(\overline{G}) = \overline{L(G)}$.

3 Turing Machines and Effective Computability

3.1 Constructing Turing Machines

1. Give a detailed description of a total Turing machine accepting the palindromes over $\{a, b\}$: that is, all strings $x \in \{a, b\}^*$ such that $x = \textbf{rev}\ x$.

   Solution: We build a machine which repeatedly scans the tape from left to right, trying to match the first input symbol (which is directly replaced by the blank symbol) with the last input symbol (also directly replaced by the blank symbol).

2. Give a detailed description of a Turing machine with input alphabet $\{a, \sharp\}$ that on input $a^m \sharp a^n$ halts with $a^{(m \bmod n)}$ written on its tape. Explain the underlying algorithm.

   Solution: Here is one possible algorithm, consisting of three phases. It implements modulo division by repeated subtraction.

   Preparatory phase: scan right to first blank symbol and replace it with $\dashv$.

   Main phase: repeat in rounds, in each round performing:
   repeatedly scan from right to left, matching the rightmost a on the right of $\sharp$ with the rightmost a on the left of $\sharp$. The matching is done by replacing the corresponding a's with $\dot{a}$'s.
   A round terminates in one of two possible ways:
   (a) if there are no more a's on the right of $\sharp$, then delete (that is, replace with the blank symbol) all $\dot{a}$'s on the left of $\sharp$, and restore all $\dot{a}$'s on the right of $\sharp$ to a's. Start new round.
   (b) if there is no matching a on the left of $\sharp$, then go to the next phase.

   Finalizing phase:
   - replace all $\dot{a}$'s by a's on the left of $\sharp$, and
   - delete all other symbols on the tape.

3. Give a detailed description (preferably as a graph) of a total Turing machine accepting the language:

   $$A = \left\{ a^n b^{\frac{n(n+1)}{2}} \mid n \geq 0 \right\}$$

   Explain the underlying algorithm.

   Solution (Idea) We use the well–known equation $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$. We proceed in rounds, in each round marking an a and then marking as many b's as there are marked a's (which equals the number of the current round).

4. Give a detailed description (preferably as a graph) of a Turing machine with input alphabet $\{a, b\}$, that on any input x halts with string $y \in L(a^*b^*)$ written on its tape, where $\sharp a(x) = \sharp a(y)$ and $\sharp b(x) = \sharp b(y)$. Explain how your Turing machine achieves its task.

   Solution: (Idea) There are many possible solutions, but one simple idea is to use insertion sort: Iteratively "swap" the left–most b with the first following a (using renaming), until there are no more such a's. (6 states suffice!)

5. Let M range over deterministic finite automata (DFA).

   (a) Describe a uniform, injective encoding of DFAs into strings over the alphabet $\{0, 1\}$.
   Solution: Such an encoding was discussed in class.

   (b) Let $\hat{M} \in \{0, 1\}^*$ denote the encoding of M. As we know from Cantor's theorem, the set

   $$A \stackrel{\text{def}}{=} \left\{ \hat{M} \in \{0, 1\}^* \mid \hat{M} \notin L(M) \right\}$$

   is not regular (since it is the inverted diagonal set). Argue, however, that A is recursive by giving a (reasonably detailed) description of a total Turing machine accepting A.
   Solution: (Sketch) The Turing machine T starts by modifying the input $\hat{M}$ to $\hat{M}\sharp\hat{q_0}\sharp\hat{M}$ where q_0 is the initial state of M. The added part $\hat{q_0}\sharp\hat{M}$ represents the initial configuration of M (where a configuration of a DFA is understood as a pair consisting of a state and the suffix of the input

string that remains to be read). T continues by simulating M on $\hat{M}$, by iteratively computing and updating the current state of M until the input string $\hat{M}$ has been completely consumed. This is achieved by reading (and consuming) the next symbol of the input string $\hat{M}$ of M, and looking up from $\hat{M}$ (always available in the first segment of the tape) the next state of M. Finally, T checks whether the state in the final configuration of M is an accepting state, and rejects resp. accepts accordingly. Thus, T is a total Turing machine accepting language A.

6. Consider the language:
$$A = \left\{a^l b^m a^n \mid n = \mathbf{max}(l, m)\right\}$$
   (a) Apply the Pumping Lemma for context–free languages (as a game with the Demon) to show that A is not context–free.
   (b) Give a detailed description (preferably as a graph) of a total Turing machine accepting A.

7. Consider the language:
$$L \stackrel{\text{def}}{=} \{ab,\ ababb,\ ababbabbb,\ ababbabbbabbbb,\ \ldots\}$$
   (a) Show that L is not context–free.
   (b) Describe a total Turing machine accepting L. Explain the workings of your machine/algorithm. (If possible, provide a graph of its control automaton.)

3.2 Proving Undecidability using Reduction

8. Argue that acceptance is not decidable: that is, that there is no total Turing machine M_A accepting the language $L_A \stackrel{\text{def}}{=} \left\{\hat{M}\sharp\hat{x} \mid M \text{ accepts } x\right\}$, by reducing from the Halting problem.

   Solution: A TM halts on input x if it either accepts x or otherwise rejects x. We use this observation to show that the Halting problem (cf. Lecture 31) can be reduced to the above acceptance problem.

   Assume that there is a total Turing machine M_A accepting L_A. We can then build a machine M_R which, on any input $\hat{M}\sharp\hat{x}$, first swaps the values of t and r in $\hat{M}$ (that is, swaps the accepting and the rejecting states of M), and then behaves exactly like M_A. Hence M_R is a total Turing machine deciding rejection. We can now combine M_A and M_R to produce a total Turing machine M_H deciding the Halting problem: for example, on any input $\hat{M}\sharp\hat{x}$, let M_H first run as M_A and accept if M_A accepts, but continue as M_R if M_A rejects. M_H will thus accept $\hat{M}\sharp\hat{x}$ if M halts on x, and will reject $\hat{M}\sharp\hat{x}$ otherwise.

   But the Halting problem is undecidable, and therefore there is no total Turing machine M_A accepting L_A. The acceptance problem is therefore undecidable.

9. Show that the problem of whether a Turing machine, when started on a blank tape, ever writes a given symbol (say a) of its input alphabet on its tape is not decidable.

   Hint: You could reduce the undecidable problem of acceptance of the null string (problem (f), page 235) to the problem above.

   Solution: Assume the problem was decidable. Then there must be a total Turing machine M_a deciding it. We shall use M_a to build a new total Turing machine M_ϵ deciding the problem of acceptance of ϵ. (Since the latter is known to be undecidable, we shall conclude that the present problem is also undecidable.)

   We construct M_ϵ which, on input $\hat{M}$, converts $\hat{M}$ to $\hat{M}'$ such that M' is like M but:

   - a is renamed to a new letter which is added to the alphabet of $\hat{M}'$,
   - a new state q is added, which becomes the accepting state of $\hat{M}'$, and
   - transitions $\delta(t, b) = (q, a, R)$ are added for every $b \in \Gamma$.

Then rewind and run as M_a on $\hat{M}'$.

We can now deduce:

$$\begin{array}{rcll} M_\epsilon \text{ accepts } \hat{M} & \Leftrightarrow & M_a \text{ accepts } \hat{M}' & \left\{\text{Def. } M_\epsilon \text{ and } \hat{M}\right\} \\ & \Leftrightarrow & M' \text{ reaches } t \text{ starting from } \epsilon & \left\{\text{Def. } M_a \text{ and } \hat{M}'\right\} \\ & \Leftrightarrow & M \text{ accepts } \epsilon & \left\{\text{Def. } \hat{M}'\right\} \end{array}$$

So, M_ϵ decides the problem of acceptance of ϵ. Since the latter is known to be undecidable, we conclude that the present problem is also undecidable.

10. Consider the language

$$VTP \stackrel{\text{def}}{=} \left\{\hat{M}\sharp\hat{x}\sharp\hat{q} \mid M \text{ run on } x \text{ visits } q \text{ twice}\right\}$$

where "M visits q" means that Turing machine M is at a configuration with control state q. Show that language VTP is not recursive by reducing from undecidability of the Membership Problem. That is, given a total Turing machine M_{VTP} deciding VTP, construct a total Turing machine M_{MP} deciding MP, where:

$$MP \stackrel{\text{def}}{=} \left\{\hat{M}\sharp\hat{x} \mid M \text{ accepts } x\right\}$$

Solution: Assume there is a total Turing machine M_{VTP} accepting VTP. Then we can construct a Turing machine M_{MP} as follows.

On input $\hat{M}\sharp\hat{x}$, M_{MP} modifies the input to $\widehat{M'}\sharp\hat{x}\sharp\hat{v}$ where M' is like M but is modified as follows: two new control states u and v are added, the latter of which is made the new accept state of M', and the transition function δ of M is extended to δ' so that $\delta'(t, a) = (u, \natural, R)$ and $\delta'(u, a) = (t, a, L)$ for all a in the input alphabet of M, and $\delta'(t, \natural) = (v, \natural, R)$, where t is the original accept state of M and $\natural$ is a tape symbol of M_{MP} not used elsewhere. Notice that M' visits state t twice on input x exactly when M visits t on x, that is, when M accepts x. M_{MP} then continues as M_{VTP}.

Then:

$$\begin{array}{rcl} M_{MP} \text{ accepts } \hat{M}\sharp\hat{x} & \Leftrightarrow & M_{VTP} \text{ accepts } \widehat{M'}\sharp\hat{x}\sharp\hat{v} \\ & \Leftrightarrow & M' \text{ run on } x \text{ visits } v \text{ twice} \\ & \Leftrightarrow & M \text{ accepts } x \end{array}$$

Since M_{VTP} is total, so is M_{MP}, and so M_{MP} decides MP which we know to be undecidable. Hence there is no total Turing machine M_{VTP} accepting VTP.

3.3 Rice's Theorem

11. Recall Rice's Theorem, textbook Lecture 34. Explain why the trivial properties of the recursively enumerable sets are decidable, by suggesting suitable total Turing machines for these properties.

Solution: There are exactly 2 trivial properties: the empty set (of r.e. sets) and the set of all r.e. sets. Since we represent every r.e. set by some (encoding of a) Turing machine accepting this set, the two trivial properties can be represented, by using some fixed encoding of Turing machines, as the languages $\left\{\hat{M} \mid \text{false}\right\}$, which is the empty set, and $\left\{\hat{M} \mid \text{true}\right\}$, which is the set of all legal Turing machine encodings.

A total Turing machine M_F accepting the first language is simply one that upon reading $\vdash$ immediately enters its reject state, while a total Turing machine M_T accepting the second language is one which decides whether the input string is a legal encoding of some Turing machine according to the chosen encoding scheme.