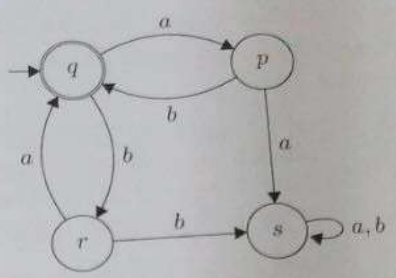
(2 points) Question 1. Consider the following automaton $\mathcal{A}$ over the alphabet $\Sigma = \{a, b\}$.

2.0



For the following questions, just state your answers. You do not need to prove them.

(i) Is $\mathcal{A}$ deterministic or non-deterministic?

(ii) Is aaa accepted by $\mathcal{A}$?

(iii) Is $ababa$ accepted by $\mathcal{A}$?

(iv) Is $baabb$ accepted by $\mathcal{A}$?

Solutions for Question 1.

(i) deterministic

(ii) No

(iii) No

(iv) No

2

(2 points) Question 2. Consider the following grammar $\mathcal{G} = \langle \Sigma, V, R, S \rangle$, where the components Σ, V, R, S are as follows.

- The alphabet $\Sigma = \{a, b\}$.
- The set of variables $V = \{S, A, B\}$.
- S is the starting variable.
- R contains the following rules:

$$\begin{aligned} S &\rightarrow aB \mid bA \\ A &\rightarrow a \mid aS \mid bAA \\ B &\rightarrow b \mid bS \mid aBB \end{aligned}$$

For the following questions, just state your answers. You do not need to prove them.

(i) Is $abba \in L(\mathcal{G})$?

(ii) Is $baaba \in L(\mathcal{G})$?

(iii) Is $aaa \in L(\mathcal{G})$?

(iv) Is $bbba \in L(\mathcal{G})$?

Solutions for Question 2.

(i) Yes

(ii) No

(iii) No

(iv) No

2

(2 points) Question 3. Consider the following Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{acc}, q_{rej}, \delta \rangle$.

- $\Sigma = \{0, 1\}$, $\Gamma = \{\lhd, 0, 1, \sqcup\}$, and $Q = \{q_0, p_0, p_1, s, t, r_0, r_1, q', q_{acc}, q_{rej}\}$.

  As usual, q_0, q_{acc}, q_{rej} are the initial, accepting and rejecting states, respectively.
- δ is defined as follows.

$$
\begin{array}{lll}
(q_0, \sqcup) \to (q_{rej}, \sqcup, \texttt{Stay}) & (p_0, \sqcup) \to (q_{rej}, 0, \texttt{Stay}) & (p_1, \sqcup) \to (s, 1, \texttt{Stay}) \\
(q_0, 0) \to (p_0, \lhd, \texttt{Right}) & (p_0, 0) \to (p_0, 0, \texttt{Right}) & (p_1, 0) \to (p_0, 1, \texttt{Right}) \\
(q_0, 1) \to (p_1, \lhd, \texttt{Right}) & (p_0, 1) \to (p_1, 0, \texttt{Right}) & (p_1, 1) \to (p_1, 1, \texttt{Right}) \\
(q_0, \lhd) \to (q_{rej}, \lhd, \texttt{Stay}) & (p_0, \lhd) \to (q_{rej}, \lhd, \texttt{Stay}) & (p_1, \lhd) \to (q_{rej}, \lhd, \texttt{Stay}) \\
\\
(s, \sqcup) \to (q_{rej}, \sqcup, \texttt{Stay}) & (t, \sqcup) \to (q_{rej}, \sqcup, \texttt{Stay}) & (q', \sqcup) \to (q', 0, \texttt{Left}) \\
(s, 0) \to (t, 1, \texttt{Left}) & (t, 0) \to (t, 0, \texttt{Left}) & (q', 0) \to (r_0, 0, \texttt{Left}) \\
(s, 1) \to (s, 0, \texttt{Left}) & (t, 1) \to (t, 1, \texttt{Left}) & (q', 1) \to (r_1, 0, \texttt{Left}) \\
(s, \lhd) \to (r_1, \lhd, \texttt{Right}) & (t, \lhd) \to (q_{acc}, \lhd, \texttt{Stay}) & (q', \lhd) \to (q_{acc}, \lhd, \texttt{Right}) \\
\\
(r_0, \sqcup) \to (t, 0, \texttt{Left}) & (r_1, \sqcup) \to (t, 1, \texttt{Left}) & \\
(r_0, 0) \to (r_0, 0, \texttt{Right}) & (r_1, 0) \to (r_0, 1, \texttt{Right}) & \\
(r_0, 1) \to (r_1, 0, \texttt{Right}) & (r_1, 1) \to (r_1, 1, \texttt{Right}) & \\
(r_0, \lhd) \to (q_{rej}, \lhd, \texttt{Stay}) & (r_1, \lhd) \to (q_{rej}, \lhd, \texttt{Stay}) &
\end{array}
$$

State which of the following words accepted by the Turing machine above. Here you also do not need to prove your answers.

(a) 00.

(b) 01.

(c) 10.

(d) 11.

Note that this is exactly the same Turing machine as in HW 3.

Solutions for question 3.

(b) (d) are the words accepted by the TM

The TM above accepts only "odd numbers" and will reject "even numbers"

1

(1 point) Question 4. Prove that for every two languages L_1 and L_2:

If L_1 is decidable and L_2 is undecidable, then $L_1 \leq_m L_2$.

1

Solution for question 4.

Since L_2 is undecidable, I can find two words w_1, w_2, such that $w_1 \in L_2$, and $w_2 \notin L_2$. (If I can't find w_1, then $L_2 = \emptyset$, but $\emptyset$ is decidable. If I can't find w_2, then $L_2 = \Sigma^*$, but Σ^* is decidable.)

I will construct a computable function F as follows:

Input: a word $w \in \Sigma^*$

Output: If $w \in L_1$, output w_1.

If $w \notin L_1$, output w_2.

Since L_1 is decidable, I can check $w \in L_1$, $w \notin L_1$ by running w on the TM which decides L.

By the computable function F constructed above, I can show that $w \in L_1$ iff $F(w) \in L_2$, thus $L_1 \leq_m L_2$

- $w \in L_1 \Rightarrow F(w) \in L_2$

  If $w \in L_1$, then $F(w)$ output w_1, by the definition of w_1, $w_1 \in L_2 \Rightarrow F(w) \in L_2$

  If $w \notin L_1$, then $F(w)$ output w_2, by the definition of w_2, $w_2 \notin L_2 \Rightarrow F(w) \notin L_2$

- $F(w) \in L_2 \Rightarrow w \in L_1$

  If $F(w) \in L_2$, then $F(w)$ must be w_1 by the construction of F, and since $F(w)$ is $w_1 \Rightarrow w \in L_1$

  If $F(w) \notin L_2$, then $F(w)$ must be w_2 by the construction of F, and since $F(w)$ is $w_2 \Rightarrow w \notin L$.

$\Rightarrow L_1 \leq_m L_2$

(1 point) Question 5. Is the statement below still true for every two languages L_1 and L_2?

If L_1 and L_2 are undecidable, then $L_1 \leqslant_m L_2$.

1

Solution for question 5.

No, since $HALT_0$, $HALT_0'$ are both undecidable language, but it is impossible to show that $HALT_0 \leq HALT_0'$ => disprove it.

prove it that $\not\leq$

(1 point) Question 6. Prove that the following problem CFL-Difference is undecidable.

CFL-Difference	
Input:	Two CFG $\mathcal{G}_1$ and $\mathcal{G}_2$.
Task:	Output True, if $L(\mathcal{G}_1) - L(\mathcal{G}_2) = \emptyset$. Otherwise, output False.

1

Solution for question 6.

I will show that CFL-Subset $\leq_m$ CFL-Difference, and since CFL-Subset is undecidable $\Rightarrow$ CFL-Difference is undecidable.

Let g_{1s}, g_{2s} be the input of CFL-Subset problem, I will construct $F(g_{1s}, g_{2s})$, such that output g_{1s} as the g_1 in CFL-Difference, and output g_{2s} as the g_2 in CFL-Difference.

Now, I want to show that $(g_{1s}, g_{2s}) \in$ CFL-Subset iff $F(g_{1s}, g_{2s}) \in$ CFL-Difference

- $(g_{1s}, g_{2s}) \in$ CFL-Subset $\Rightarrow F(g_{1s}, g_{2s}) \in$ CFL-Difference.

If $(g_{1s}, g_{2s}) \in$ CFL-Subset, then $\forall w \in L(g_{1s})$, $w \in L(g_{2s})$. so by the definition of "$-$" ($L(g_1) - L(g_2) = \{w \mid w \in L(g_1)$ and $w \notin L(g_2)\}$), $L(g_1) - L(g_2) = \emptyset \Rightarrow F(g_1, g_2) \in$ CFL-Difference.

If $(g_{1s}, g_{2s}) \notin$ CFL-Subset, then there exist a w, such that $w \in L(g_{1s})$, but $w \notin L(g_{2s})$. so by the definition of "$-$", $L(g_1) - L(g_2) \neq \emptyset \Rightarrow F(g_1, g_2) \notin$ CFL-Difference

- $F(g_{1s}, g_{2s}) \in$ CFL-Difference $\Rightarrow (g_{1s}, g_{2s}) \in$ CFL-Subset

If $F(g_{1s}, g_{2s}) \in$ CFL-Difference, then by definition of "$-$", there's no w, such that $w \in L(g_1)$ and $w \notin L(g_2)$, so $L(g_{1s}) \subseteq L(g_{2s}) \Rightarrow (g_{1s}, g_{2s}) \in$ CFL-Subset

If $F(g_{1s}, g_{2s}) \notin$ CFL-Difference, then by the definition of "$-$", there exist a w, such that $w \in L(g_1)$ and $w \notin L(g_2)$, According to that, I can write $L(g_{1s}) \nsubseteq L(g_{2s}) \Rightarrow (g_{1s}, g_{2s}) \notin$ CFL-Subset

$\Rightarrow (g_{1s}, g_{2s}) \in$ CFL-Subset iff $F(g_{1s}, g_{2s}) \in$ CFL-Difference

$\Rightarrow$ CFL-Subset $\leq_m$ CFL-Difference, and since CFL-Subset is undecidable $\Rightarrow$ CFL-Difference is undecidable, (1 point) **Question 7.** Prove that the following problem CFL-Complement is undecidable.

CFL-Complement	
Input:	A CFG $\mathcal{G} = (\Sigma, V, R, S)$.
Task:	Output True, if $\Sigma^* - L(\mathcal{G})$ is a CFL. Otherwise, output False.

1

Solution for question 7.

I want to show that $L_\infty \leq_T$ CFL-Complement, and since L_∞ is undecidable $\Rightarrow$ CFL-Complement is not decidable.

First, I want to prove one thing: let M be a TM, the words that are not accepting runs on M is a CFL. I will use a non-deterministic PDA to prove it, the construction of the PDA goes as follows:

- check every configuration is valid: it's trivial that one can construct a PDA to do that. If it goes to an invalid configuration, ACCEPT this word.
- check the run ends at an accepting configuration. If the last configuration is not q_{acc} configuration, or the accepting configuration is not in the end of run, ACCEPT this word. It's obvious that there's a PDA can do this.
- check every transition is valid: I can construct a PDA, such that if the stack is empty, push the current configuration is the stack, if the stack is not empty, check the current configuration and the configuration in the stack, such that whether there's a rule can do the transfer. After that, put the current configuration in the stack again. It can be shown that there's a PDA can do this. If any invalid transition occured, ACCEPT the word.

Solution for question 7 (continued).

So, from the above construction of 3 different PDA that checks whether a run is accepted by the TM, the runs that is NOT accepted by the TM can be generated by a PDA, thus it is a CFG.

So, if there's a turing machine M that can solve CFL-Complement, then I can construct F as follows:

Input: a $w \in L_\infty$

Output: the corresponding CFG that generates all the words which are not accepted by the TM.

By the resultes that the accepting run of a TM which can generate infinitely many words is not a CFL, Thus $\Sigma^* - F(w)$ is always false. (Since $w \in L_\infty$ mean that turing machine w can accept infinitely many words,

So $L_\infty \leq_m$ CFL-Complement, and since L_∞ is an undecidable language $\Rightarrow$ CFL-Complement is an undecidable language, too!