1. Question 4.1 page 183. This is for exercise only. Solutions are given on page 185.

2. Question 4.2 page 183.

**Answer:** Express this problem (determining whether a DFA and a regular expression are equivalent) as the following language:

\[ EQ_{DFA,REX} = \{ \langle D, E \rangle \mid D \text{ is a DFA, } E \text{ is a reg. exp. and } L(D) = L(E) \} \]

To prove this language is decidable, we have two different methods.

**Method 1:** we reduce it to \( EQ_{DFA} \) which is known to be decidable. We construct the mapping function \( f : \Sigma^* \rightarrow \Sigma^* \) such that for any input \( \langle D, E \rangle \),

1. \( f(\langle D, E \rangle) = \langle D, D_E \rangle \), where \( D_E \) is a DFA constructed from the regular expression \( E \) such that \( L(D_E) = L(E) \); the construction of \( D_E \) can be done in a finite number of steps based on the proofs of Lemma 1.55 (page 67) and Theorem 1.39 (page 55);
2. \( L(D) = L(E) \) if and only if \( L(D) = L(D_E) \).

So

\[ EQ_{DFA,REX} \leq_m EQ_{DFA} \]

By Theorem 5.22 (page 208) and the fact that \( EQ_{DFA} \) is decidable, \( EQ_{DFA,REX} \) is also decidable.

**Method 2:** Since \( EQ_{DFA} \) is decidable, we assume that is an algorithm \( M \) (i.e., Turing machine that halts on all inputs) that decides if two given DFAs accepts the same language. We now construct another algorithm \( M_1 \) to decide language \( EQ_{DFA,REX} \).
The algorithm $M_1$ inputs $\langle D, E \rangle$, where $D$ is a DFA and $E$ is a regular expression. $M_1$ works as follows:

First, $M_1$ converts $E$ to an equivalent NFA (by Lemma 1.55) and then to an equivalent DFA (by Theorem 1.39). Let it be $D_E$ such that $L(D_E) = L(E)$.

Second, $M_1$ simulates $M$ on input $\langle D, D_E \rangle$.

Finally, $M_1$ accepts $\langle D, E \rangle$ if $M$ accepts $\langle D, D_E \rangle$; $M_1$ rejects $\langle D, E \rangle$ if $M$ rejects $\langle D, D_E \rangle$.

Clearly, $M_1$ halts on all inputs and it decides language $EQ_{DFA,REX}$.

3. Question 4.3 page 183.

Answer: This question also has two different methods, much like Question 4.2. But we only show one here.

Since $EQ_{DFA}$ is decidable, we assume that is an algorithm $M$ (i.e., Turing machine that halts on all inputs) that decides if two given DFAs accepts the same language. We now construct another algorithm $M_1$ to decide language $ALL_{DFA}$.

The algorithm $M_1$ inputs $\langle A \rangle$, where $A$ is a DFA. $M_1$ works as follows:

First, $M_1$ generates a DFA $B$ that accepts language $\Sigma^*$. The construction of such a DFA is easy since it can just be the one consisting of a single state $q$ (which is both the start and accepting state) with transition function $\delta(q, 0) = q; \delta(q, 1) = q$.

Second, $M_1$ simulates $M$ on input $\langle A, B \rangle$.

Finally, $M_1$ accepts $\langle A \rangle$ if $M$ accepts $\langle A, B \rangle$; $M_1$ rejects $\langle A \rangle$ if $M$ rejects $\langle A, B \rangle$.

Clearly, $M_1$ halts on all inputs and it decides language $ALL_{DFA}$.

4. Question 4.4 page 183.

Answer: We construct an algorithm $M$ to decide language $A_{CFG}$.

$M$ works as follows.

Given input $\langle G \rangle$, $M$ first convert the grammar $G$ to Chomsky normal form $G'$ using the steps provided by the proof of Theorem 2.9 (page 107) such that $L(G) = L(G')$. Since variables other than the newly introduced start variable $S_0$ cannot have $\epsilon$-rules in $G'$, $M$ checks if any rule of $S_0$ is an $\epsilon$-rule. If so, $M$ accepts $\langle G \rangle$; if not, $M$ rejects $\langle G \rangle$.

Clearly, $M$ halts on all inputs and it determines a given grammar generates string $\epsilon$, i.e., it decides language $A_{CFG}$. 2
5. Read and understand the proof of Theorem 4.11 (the proof is on pages 179-181). Use succinct logical reasoning statements to rewrite the proof. You can only use succinct logical statements, assumptions, and/or inferences, with a total up to 20 sentences. Each sentence can be at most one line long. Number these sentences and they are connected logically in a sequence.

**Answer:** We prove by contradiction using the following sequence of logical statements

(a) Assume \( A_{TM} \) is decided by algorithm \( U \) that halts on all inputs.
(b) \( U \) accepts \( \langle M, \omega \rangle \) if \( M \) accepts \( \omega \); \( U \) rejects \( \langle M, \omega \rangle \) otherwise.
(c) Let \( D \) be a TM that, given \( \langle M \rangle \), reverses the answer of \( U \) on input \( \langle M, \langle M \rangle \rangle \).
(d) \( D \) halts on all inputs because \( U \) halts on all its inputs.
(e) When \( D \) is given the input \( \langle D \rangle \) (i.e., itself), assume \( D \) accepts \( \langle D \rangle \).
(f) Above implies \( U \) rejects \( \langle D, \langle D \rangle \rangle \).
(g) Above means \( D \) does not accept \( \langle D \rangle \).
(h) Contradict.
(i) When \( D \) is given the input \( \langle D \rangle \) (i.e., itself), assume \( D \) rejects \( \langle D \rangle \).
(j) Above implies \( U \) accepts \( \langle D, \langle D \rangle \rangle \).
(k) Above means \( D \) accepts \( \langle D \rangle \).
(l) Contradict.
(m) Paradox.
(n) \( U \) does not exist.
(o) \( A_{TM} \) is not decidable.

6. Question 5.1 page 211. (Hint: reduce language \( ALL_{CFG} \) to language \( EQ_{CFG} \). \( ALL_{CFG} \) is proved to be undecidable in Theorem 5.13).

**Answer:** Based on the hint, we reduce language \( ALL_{CFG} \) to language \( EQ_{CFG} \) using function \( f \) defined as follows.

\[
ALL_{CFG} = \{ \langle G \rangle | G \text{ is a CFG, } L(G) = \Sigma^* \}
\]
f maps instance \( \langle G \rangle \) to instance \( \langle G, G_1 \rangle \), where \( G_1 \) is a context-free grammar that generates \( \Sigma^* \) with rules:

\[
S \rightarrow 0S \mid 1S \mid \epsilon
\]

Clearly, \( f \) can be computed by an algorithm (i.e., a Turing machine that halts on all inputs). Furthermore,

\( \langle G \rangle \in ALL_{CFG} \) if and only if \( \langle G, G_1 \rangle \in EQ_{CFG} \)

So \( ALL_{CFG} \leq_m EQ_{CFG} \). Because \( ALL_{CFG} \) is undecidable (Theorem 5.13), by Theorem 5.23 \( EQ_{CFG} \) is also undecidable.

7. Question 5.4 page 211.

**Answer:** If \( A \leq_m B \) and \( B \) is regular, it does NOT necessarily imply that \( A \) is also regular. This is because the reduction function can be more than the “power” of a DFA. In fact, reduction functions are required only to halt on all inputs but can be perform very sophisticated computations.

For example, Let \( A = \{0^n1^n \mid n \geq 0\} \), which is not a regular language. We reduce \( A \) to \( B = \{11\} \) using function \( f \) that is computed by the following algorithm \( M \). Note that \( B \) consists of only one string 11 so \( B \) is regular.

\( M \) uses a stack to recognize every input string \( \omega \) and accepts it if and only if it is in the form of \( 0^n1^n \) for some \( n \geq 0 \). So \( M \) recognizes \( A \). Moreover, \( M \) outputs 11 if it accepts the input string \( \omega \) and outputs 00 if it rejects the input string \( \omega \). So the function \( f \) computed by \( M \) is actually the following function:

for any \( \omega \in \Sigma^* \),

\[
f(\omega) = 11 \text{ if } \omega = 0^n1^n \text{ for some } n \geq 0
\]

\[
f(\omega) = 00 \text{ if } \omega \neq 0^n1^n \text{ for any } n \geq 0
\]

So function \( f \) maps instances of \( A \) to the only instance of \( B \), i.e., 11; and \( f \) maps instances of \( \overline{A} \) to 00 which is in \( \overline{B} \). That is for any \( \omega \in \Sigma^* \),

\[
\omega \in A \text{ if and only if } f(\omega) \in B
\]
Clearly, $M$ that computes $f$ halts on all inputs. So $A \leq_m B$.

But $B$ is regular, while $A$ is not.

8. Questions 5.6 and 5.7 (exercise only, you do not need to turn in your answers; solutions are given on page 214)