Computability and Complexity

Exercises

Decidability

Exercise 1 : $0 - 1$ ***sequences.***
Prove that $\{0,1\}^{\mathbb{N}}$, the set of all infinite sequences over $\{0,1\}$, is uncountable.
Answer: We do a proof by diagonalization. Assume by contradiction that there is a bijective function f from $\mathbb{N}$ to $\{0,1\}^{\mathbb{N}}$. To each positive integer n we have an associated sequence s_n. Denote by $s_n(i)$ the i-th digit in s_n. We construct the sequence s as follows:

$$s(n) = 1 - s_n(n)$$

Thus, the n-th digit in s is 0 if and only if the n-th digit in s_n is 1 (and conversely). It is easy to see that $s \notin \mathrm{Im}(f)$, but $s \in \{0,1\}^{\mathbb{N}}$, a contradiction.

$\square$

Exercise 2 : ***A whole language.***
Prove that $L = \{< A > \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$ is decidable.
Answer: A DFA that accepts every word over Σ has the following properties:

1. *It is complete (meaning that for every $(q,a) \in Q \times \Sigma$ such that q is accessible from the initial state q_0, there is an $r \in Q$ such that $\delta(q,a) = r$);*
2. *Every state is accepting (meaning $F = Q$).*

So we can construct a Turing machine that verifies those two properties: simply do a breadth-first search starting from the initial state and check that each state you reach is accepting and has a successor state through δ with every character in Σ. The Turing machine will necessarily halt since A is finite. Reject at any point if a condition is not verified, and otherwise (if the computation ends without rejecting) accept. Such a Turing machine decides L.

$\square$

Exercise 3 : ***Regexps.***
Consider the problem of deciding whether a DFA A and a regular expression E verify $L(A) = L(E)$. Express this problem as a language and prove that it is decidable.
Answer: The language we consider is $\{< A, E > \mid L(A) = L(E),\ A \text{ is a DFA},\ E \text{ is a regular expression}\}$. To prove that it is decidable, we use the following Turing machine:

1. *Convert E into an equivalent NFA B using the algorithm seen in Chapter 1.*
2. *Convert B into an equivalent DFA B' using the algorithm seen in Chapter 1.*
3. *Run the Turing machine from Theorem 4.5 of Chapter 4 on input $< A, B' >$. If the machine accepts, then accept; otherwise, reject.*

It is easy to see that this machine decides our language.

$\square$

Exercise 4 : ***Towards the infinity.***
Prove that $L = \{< A > \mid A \text{ is a DFA and } L(A) \text{ is infinite}\}$ is decidable.
Hint: Think about the pumping lemma!

Answer: We want to reject the automatas that have a finite language and accept those that have an infinite language. The pumping lemma guarantees that if a regular language contains a word of length at least p (with p being its pumping length), then it will be infinite, since it will be possible to pump the word. Thus, the language we want to decide is equivalent to $\{< A > \mid A \text{ is a DFA and } \exists w \in L(A) \text{ such that } |w| \geq p\}$. *Now, we also know that the pumping length is at most the number of states in A. So we can construct the following Turing machine:*

1. *Let k be the number of states in A.*
2. *Construct a DFA K that accepts all words of length k or more (it is trivial to construct).*
3. *Construct a DFA B that verifies $L(B) = L(A) \cap L(K)$ (it is possible, as seen in the homework on languages).*
4. *Test whether $L(B) = \emptyset$ using the Turing machine constructed in the course (Theorem 4.4). If the machine accepts, then* reject*; otherwise* accept.

It is easy to see that it decides the language, and thus that it decides L.

□

Exercise 5 : *Accepting palindromes.*
Prove that $L = \{< A > \mid A$ is a DFA and A accepts some palindrome$\}$ is decidable.
Hint: Think about a CFG that generates a palindrome, and prove that the intersection between a regular language and a context-free language is context-free.
Answer: First we prove that if L_r is regular and L_c is context-free, then $L_r \cap L_c$ is context-free. Let $A_r = (Q_r, \Sigma, \delta_r, q_r, F_r)$ be a DFA that recognizes L_r, and let $A_c = (Q_c, \Sigma, \Gamma, \delta_c, q_c, F_c)$ be a PDA that recognizes L_c. The automata $(Q, \Sigma, \Gamma, \delta, q_0, F)$ with:

- $Q = Q_r \times Q_c$
- $\delta((q_1, q_2), \ell, a) = ((\delta_r(q_1, a), r), b)$ *with* $(r, b) \in \delta_c(q_2, \ell, a)$
- $q_0 = (q_r, q_c)$
- $F = (q_1, q_2)$ *with* $q_1 \in F_r$ *and* $q_2 \in F_c$

recognizes $L_r \cap L_c$, which proves the result.
The grammar with rule $S \to \epsilon \mid a \mid b \mid aSa \mid bSb$ generates the language of all palindromes. Let P be a pushdown automata that recognizes the same language. We have the following Turing machine:

1. *Construct a PDA B that verifies $L(B) = L(A) \cap L(P)$ (it is possible since the intersection of a context-free language and of a regular language is context-free).*
2. *Construct a CFG G_B such that $L(G_B) = L(B)$ (using the method described in the course).*
3. *Test whether $L(G_B) = \emptyset$ using the Turing machine constructed in the course (Theorem 4.8). If the machine accepts, then* reject*; otherwise* accept.

This Turing machine clearly decides L.

□