Assignment 3 solutions

1. Suppose that $f : \Sigma^* \to \Sigma^*$ is a computable function with the property that

$$x \leq y \;\Rightarrow\; f(x) \leq f(y)$$

for all strings $x, y \in \Sigma^*$. (For two strings $u, v \in \Sigma^*$ we interpret $u \leq v$ to mean that either $u = v$ or u comes before v in the lexicographic ordering of Σ^*.)

Prove that $\mathrm{range}(f) = \{f(x) \,:\, x \in \Sigma^*\}$ is a decidable language.

Solution. (Prepared by Ansis Rosmanis.) Let us distinguish two cases: the case when $\mathrm{range}(f)$ is finite, and the case when $\mathrm{range}(f)$ is infinite. If $\mathrm{range}(f)$ is a finite language, then it is regular, and therefore decidable. Now suppose that $\mathrm{range}(f)$ is infinite, thus for every string x there exists a string y such that $f(y) > x$. Due to the property of f given above, $x \in \mathrm{range}(f)$ if and only if there is $z < y$ such that $f(z) = x$. Therefore, in this case the algorithm deciding whether $x \in \mathrm{range}(f)$ is:

```
1. set $z$ to be the first string in the lexicographic ordering;
2. if $f(z) = x$, accept;
3. if $f(z) > x$, reject;
4. increase $z$ to the next string in the lexicographic ordering, and go back to Step 2.
```

It is easy to see that this algorithm always stops and works correctly. Hence, $\mathrm{range}(f)$ is decidable.

2. Prove that the language

$$\{\langle D, k\rangle \,:\, D \text{ is an DFA, } k \in \mathbb{N}, \text{ and } D \text{ accepts at most } k \text{ strings over its alphabet}\}$$

is decidable.

Solution. (Prepared by Ansis Rosmanis.) Note that, if a DFA D accepts a string $x \in \Gamma^*$ of length at least m, then there exist a state $q_i \in Q$ and strings $u, v, w \in \Gamma^*$ such that $x = uvw$, $\delta^*(q_0, u) = q_i$, $\delta^*(q_i, v) = q_i$, $1 \leq |v| \leq m$, and $\delta^*(q_i, w)$ is an accepting state, where δ is the state transition function of D, and therefore D accepts all the strings in the form uv^*w and $L(D)$ is infinite. This statement also implies that, if D accepts a string $x \in \Gamma^*$ of length at least $2m$, it accepts another string $y \in \Gamma^*$ such that $m \leq |y| < 2m$. And similarly as above, if a DFA D accepts a string (of any length) and visits some state at least twice while doing so, $L(D)$ is infinite. Taking this all into account, the following algorithm decides the language given in the statement of the question, and therefore that language is decidable:

```
1. if the input is not in the form $\langle D, k\rangle$, where $D$ is a DFA and $k \in \mathbb{N}$, reject;
2. initialize a counter $l$ to be 0 (the number of accepted strings);
3. for every string $x \in \Gamma^*$ of length at most $2m - 1$, if $D$ accepts $x$, do:
    (a) increase $l$ by one;
    (b) if $l > k$, reject;
```

(c) if D visits some state twice while accepting x, <u>reject</u>;

4. (if the loop above does not reject) <u>accept</u>.

There are many alternative solution possible. One would be to consider an algorithm which first from the description of a DFA D obtains a regular expression R corresponding to the same language L as D. Then, the language L is finite if and only if R does not contain the Kleene star operation. Finally, if R does not contain the Kleene star, the algorithm can rewrite it in the form $z_1 \cup \ldots \cup z_l$, where $z_i \in \Gamma^*$ and $i \neq j \;\Rightarrow\; z_i \neq z_j$ for all i and j. Now the algorithm accepts if and only if $l \leq k$.

Alternate Solution. (Prepared by John Watrous.) Ansis's solution is fine, but when I prepared the problem I had the following solution in mind. Consider the following DTM M.

On input $\langle D, k\rangle$, where D is a DFA and k is a nonnegative integer:

1. Construct a DFA E with $L(E) = \{x \in \Gamma^* \;:\; |x| \geq m\}$, where Γ is the alphabet of D and m is the number of states of D.
2. Construct a DFA F such that $L(F) = L(D) \cap L(E)$, and <u>reject</u> if $L(F) \neq \varnothing$ (using the DTM for E_{DFA} described in lecture).
3. Count the number of strings over Γ with length less than m that D accepts. <u>Accept</u> if this number is at most k, and <u>reject</u> otherwise.

The reason this DTM correctly decides the language described in the question is as follows. We know from the proof of the pumping lemma for regular languages that if D is a DFA having m states, and D accepts any string with length m or greater, then D must accept infinitely many strings. The converse of this statement is obviously true: if D accepts infinitely many strings, then D must accept some string with length at least m. Therefore, M rejects on step 2 if and only if $L(D)$ is infinite. If $L(D)$ is finite, then M proceeds to step 3 and counts the number of strings accepted by D, all of which must have length less than m, compares this number with k and accepts or rejects appropriately.

3. Prove that the language

$$\text{INF}_{\text{CFG}} = \{\langle G\rangle \;:\; G \text{ is a context-free grammar and } L(G) \text{ is infinite}\}$$

is decidable.

Solution. (Prepared by Ansis Rosmanis.) Consider the following algorithm.

1. if the input does not correspond to a context-free grammar G, <u>reject</u>;
2. convert the grammar into Chomsky normal form, and let V_1 be the new set of variables;
3. using the following steps remove from the grammar all variables from which we cannot derive any string in Γ^*:
   (a) initialize $V_2 := \emptyset$ (the set of all variables from which we can derive a sting in Γ^*);
   (b) repeat the following till $|V_2|$ is not increased per iteration:
   update $V_2 := \{A \in V_1 \;:\; \exists B, C \in V_2 : (A \rightarrow BC) \;\text{ or }\; \exists a \in \Gamma : (A \rightarrow a)\}$;

(c) if $S \to \varepsilon$, update $V_2 := V_2 \cup \{S\}$;

(d) delete from the grammar all the rules which contains variables from $V_1 \setminus V_2$ (note that $V_2 = \emptyset$ if and only if $L(G) = \emptyset$);

4. using the following steps remove from the grammar all variables which are not 'reachable':

   (a) initialize $V_3 := \{S\}$ (the set of all 'reachable' variables);

   (b) repeat the following till $|V_3|$ is not increased per iteration:
   update $V_3 := V_3 \cup \{B \in V_2 \ : \ \exists A \in V_3 \, \exists C \in V_2 : (A \to BC \text{ or } A \to CB)\}$;

   (c) delete from the grammar all the rules which contains variables from $V_2 \setminus V_3$;

5. for every $A \in V_3 \setminus \{S\}$ do:

   (a) initialize $V_3^A := \{B \in V_3 \ : \ \exists C \in V_3 : (A \to BC \text{ or } A \to CB)\}$ (the set of all variables 'reachable' from A);

   (b) repeat the following till $\left|V_3^A\right|$ is not increased per iteration:
   update $V_3^A := V_3^A \cup \{B \in V_3 \ : \ \exists D \in V_3^A \, \exists C \in V_3 : (D \to BC \text{ or } D \to CB)\}$;

   (c) if $A \in V_3^A$, <u>accept</u>;

6. (if the loop above does not accept) <u>reject</u>.

We claim that the algorithm above decides the language INF_{CFG}. First, notice that everything we do before Step 5 is just to convert the grammar into Chomsky normal form and get rid of 'useless' variables, that is, variables which are either not 'reachable' or from which we cannot derive terminal strings. Second, if for any variable $A \in V_3$ there exist strings $x, y \in (\Gamma \cup V_3)^*$ such that $xy \neq \varepsilon$ and from A we can derive xAy, then all of the following holds:

- $A \neq S$ and neither x nor y contains S due to the properties of Chomsky normal forms,
- $A \in V_3^A$ (i.e., A can be 'reached' from itself),
- the language accepted by G is infinite because from S we can reach A and from any variable in V_3 other than S we can derive a non-empty string in Γ^*.

One also can see quite easily that if there is no such variable A, then the language accepted by G is finite. Hence, the algorithm above indeed does what we claim, and the language INF_{CFG} is decidable.

Alternate Solution. (Prepared by John Watrous.) Again, I had a slightly different solution in mind from the one Ansis has given. Consider the following DTM M.

On input $\langle G \rangle$, where G is a context-free grammar:

1. Convert G to an equivalent context-free grammar H in Chomsky Normal Form. Let k denote the number of variables in H.
2. Construct a DFA D such that $L(D) = \left\{x \in \Gamma^* \ : \ |x| \geq 2^{k+1}\right\}$, where Γ is the alphabet of G.
3. Construct a context-free grammar K for the language $L(H) \cap L(D)$.
4. Test if $L(K)$ is empty or not, using the algorithm described in lecture. If $L(K)$ is empty then <u>reject</u>, otherwise <u>accept</u>.

Along the same lines as the solution to the previous problem, we have that $L(G) = L(H)$ is infinite if and only if H generates a string having length at least 2^{k+1} (this time due to the pumping lemma for context-free languages), and therefore M decides INF_{CFG}.

The construction of K from H and D can be performed by first constructing a PDA for H, modifying it to obtain a PDA for $L(H) \cap L(D)$ using a Cartesian product construction, and converting the resulting PDA back to a CFG.

Alternately, it is not too difficult to modify H directly to obtain a CFG for $L(H) \cap L(D)$. If the set of variables of H is W, and the set of states of D is Q, then take $W \times Q \times Q$ to be the set of variables of the new grammar. See if you can fill in the rest!

4. Suppose that $A \subseteq \Sigma^*$ is a Turing-recognizable language. Define

$$B = \{x \in \Sigma^* \ : \ \text{there exists a string } y \in \Sigma^* \text{ such that } xy \in A\}.$$

Prove that B is Turing-recognizable.

Solution. (Prepared by Ansis Rosmanis.) Let us define an algorithm that recognizes B, using a DTM M_A recognizing A as a subroutine:

```
1. initialize $n := 1$;
2. for all strings $y \in \Sigma^*$ of length at most $n$, do:
    (a) run $M_A$ on $xy$ for $n$ steps (or less if $M_A$ already stops earlier);
    (b) if $M_A$ accepts $xy$ in at most $n$ steps, accept;
3. set $n := n + 1$, and go back to Step 2.
```

Now let us show that this algorithm indeed recognizes B. If $x \in B$ then there exist $n_1, n_2 \in \mathbb{N}$ such that for some $y \in \Sigma^{n_1}$ the DTM M_A accepts xy in n_2 steps. Therefore, our algorithm above accepts x using at most $\max(n_1, n_2)$ iterations of Steps 2 and 3. If $x \notin B$, then the algorithm clearly runs forever, and therefore never accepts x.

5. Assume that A is an infinite Turing-recognizable language. Prove that there exists an infinite decidable language $B \subseteq A$.

Solution. (Prepared by Ansis Rosmanis.) For every $n \in \mathbb{N}$ let $l(n)$ be the n-th string in the lexicographical order, and correspondingly for every $x \in \Sigma^*$ let $l^{-1}(x)$ be the number of the string x in the lexicographical order. Let M_A be a DTM recognizing the language M_A. Consider the following algorithm, where $x \in \Sigma^*$ denotes the input.

```
1. initialize $m := 1$;
2. run $M_A$ on $x$ for $m$ steps (or less if $M_A$ already stops earlier);
       if $M_A$ accepts $x$ in at most $m$ steps, accept;
3. initialize $n := l^{-1}(x) + 1$;
4. while $n < l^{-1}(x) + m$ do:
    (a) run $M_A$ on $l(n)$ for $m$ steps (or less if $M_A$ already stops earlier);
            if $M_A$ accepts $l(n)$ in at most $m$ steps, reject;
```

(b) set $n := n + 1$;

5. set $m := m + 1$, and go back to Step 2.

First of all, because A is infinite, for every $x \in \Sigma^*$ there is $m \in \mathbb{N}$ such that M_A accepts x or one of the next $m - 1$ following strings (lexicographically) in m steps, therefore the algorithm above always stops. Also, it is easy to see that it always rejects all the strings outside A. Therefore it decides some language B such that $B \subseteq A$. All that is left to show is that B is infinite.

Suppose M_A accepts $x_1 \in A$ in m_1 steps. Then either $x_1 \in B$ or there is a string x_2 in the next $m_1 - 2$ strings following x_1 lexicographically such that M_A accepts x_2 in $m_2 < m_1$ steps. Again, either $x_2 \in B$ or there is a string x_3 in the next $m_2 - 2$ strings following x_2 lexicographically such that M_A accepts x_3 in $m_3 < m_2$ steps, and so on. Because $(m - 2) + (m - 3) + \ldots + 1 < m^2$, for every $x \in A$ which is accepted by M_A in m steps, either $x \in B$ or one of the m^2 strings following x lexicographically is in B. Hence, A being infinite implies B being infinite as well.

Alternate Solution. (Prepared by John Watrous.) By a theorem we proved in lecture, there must exist a computable function $f : \Sigma^* \to \Sigma^*$ such that $\mathrm{range}(f) = A$. Consider the following DTM M that makes use of such a function.

```
On input $x$:
  1. Set $z = \varepsilon$.
  2. Compute $y = f(z)$.
  3. If $y = x$ then accept and if $y > x$ (in lex-order) then reject.
  4. Increment $z$ (with respect to lex-ordering) and goto 2.
```

It is clear that M halts on all inputs, given that $A = \mathrm{range}(f)$ is infinite. It is clear that $L(M) \subseteq A$, because M only accepts inputs x in the range of f. Finally, for any string z such that $f(z) > f(w)$ for all $w < z$, it holds that M accepts $f(z)$. There are infinitely many such strings $f(z)$, and so M accepts infinitely many strings. Taking $B = L(M)$ completes the solution.

6. [Bonus question] Suppose we have agreed on a reasonable encoding scheme for DTMs having the input alphabet Σ, whereby (i) every such DTM is encoded by at least one string over Σ, and (ii) every string over Σ is an encoding of some such DTM.

   Prove that there exists a DTM M with input alphabet Σ, and an encoding $\langle M \rangle \in \Sigma^*$ of M, with the following property: for M^{R} being the DTM encoded by the string $\langle M \rangle^{\mathrm{R}}$ it holds that $L(M^{\mathrm{R}}) = L(M)^{\mathrm{R}}$. In other words, by reversing the encoding $\langle M \rangle$, we obtain an encoding of a DTM accepting the reverse of the language $L(M)$.

Solution. (Solution prepared by John Watrous.) Define a DTM as follows.

```
On input $\langle \langle K \rangle, x \rangle$, where $K$ is a DTM and $x \in \Sigma^*$:
  1. Let $K^{\mathrm{R}}$ be the DTM encoded by $\langle K \rangle^{\mathrm{R}}$.
  2. Simulate $K^{\mathrm{R}}$ on input $x^{\mathrm{R}}$. If $K^{\mathrm{R}}$ accepts $x^{\mathrm{R}}$, then output $\varepsilon$ (and accept), and if $K^{\mathrm{R}}$ rejects
     $x^{\mathrm{R}}$, then enter an infinite loop.
```

This DTM partially computes the function

$$f(\langle K \rangle, x) = \begin{cases} \varepsilon & \text{if } x^{\mathrm{R}} \in L(K^{\mathrm{R}}) \\ \uparrow & \text{otherwise.} \end{cases}$$

By the recursion theorem, there exist a DTM M and an encoding $\langle M\rangle$ of M, such that M computes the function

$$g(x) = f(\langle M\rangle, x) = \begin{cases} \varepsilon & \text{if } x^{\mathrm{R}} \in L(M^{R}) \\ \uparrow & \text{otherwise.} \end{cases}$$

This means that M accepts x (as it outputs ε) if and only if M^{R} accepts x^{R}, or in other words $L(M)^{\mathrm{R}} = L(M^{\mathrm{R}})$.

Another way to describe the DTM M, along the lines of how the text would do it, is as follows:

On input x:

1. Obtain a description $\langle M\rangle$ of the DTM now being defined (which is possible by the recursion theorem), and let M^{R} be the DTM encoded by $\langle M\rangle^{\mathrm{R}}$.
2. Run M^{R} on input x^{R}.

By the same reasoning as above, it holds that $L(M)^{\mathrm{R}} = L(M^{\mathrm{R}})$. (If you prefer this style of self-referential DTM description, you are free to use it as you like in this course.)