University of Nevada, Las Vegas Computer Science 456/656 Spring 2020

Answers to Assignment 4: Due Tuesday March 24, 2020

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known to science at this time.

   (a) **T** Every language accepted by an NFA is accepted by some DFA.

   (b) **F** Every language accepted by an NPDA is accepted by some DPDA.

   (c) **T** Every language accepted by an NTM is accepted by some TM.

   (d) **T** The class of Turing machines which allow movement the head to not move during a step is equivalent to the class of Turing machines with require that the head move at each step.

   (e) **T** The class of PDAs which accept by final state is equivalent to the class of PDAs which accept by empty stack.

   (f) **F** The class of 2-PDAs, that is, automata with 2 stacks, is equivalent to the class of PDAs with just one stack.

   (g) **T** The class of C++ programs is equivalent to the class of Turing machines.

   (h) **T** The class of Turing machines with a 2-way infinite tape is equivalent to the class of Turing machines with a semi-infinite tape.

   (i) **T** The complement of any recursive language is recursive.

   (j) **F** The complement of any recursively enumerable language is recursively enumerable.

2. Prove that a language L is recursive if and only if there is a machine which enumerates L is canonical order.

   Suppose $L \subseteq \Sigma^*$ is recursive. Let $w_1, w_2, \ldots$ be the canonical order of Σ^*, which can be easily generated by a program. The following program enumerates L in canonical order. Note that the condition $w \in L$ can be evaluated because L is decidable.

```
for(int i = 1; true; i++) // that makes it an infinite loop
 if (w_i ∈ L)
  cout<< w_i
```

   Conversely, suppose there is a machine M which enumerates L in canonical order. If L is finite, we are done, since every finite language is decidable. If L is infinite, let $u_1, u_2, \ldots$ be the members of L in canonical order. The following program decides whether a given string w is a member of L.

```
read(w)
for u_i for all i in increasing order // obtained by emulation of M
 if(u_i == w) {return 1; halt;}
 else if(u_i > w) {return 0; halt;}
```

Let $w_1, w_2, \ldots$ be the canonical order of Σ^*, which can be easily generated by a program. Let M be a machine which accepts L. The following program enumerates L, but not necessarily in canonical order.

```
for(int t = 1; true; t++) // that makes it an infinite loop
  for(int i = 1; i ≤ t; i++)
    if (M accepts w_i in at most t steps)
      cout<< w_i
```

If w_i is accepted by M it is accepted in k steps for some finite k. Let $t = \max i, k$. Then w_i will be written during the t^{th} iteration of the outer loop.

Conversely, suppose there is a machine M which enumerates L. Let $u_1, u_1, \ldots$ be that enumeration. The following program accepts L.

```
read(w)
for u_i for all i // obtained by emulation of M
  if(u_i == w) halt
```

Note that the program will never halt if $w \notin L$.

3. Give an unrestricted grammar which generates $L = \left\{a^{n^2} \ : \ n \geq 0\right\}$.

   I thought I would find a solution on the internet, but I didn't. I believe the following grammar generates L. The variables are S, F, R, A, B, D and the only terminal is a.
   $S \to \lambda$
   $S \to FABR$
   $F \to FABB$
   $AB \to aBA$
   $Aa \to aA$
   $AR \to R$
   $F \to D$
   $Da \to aD$
   $DB \to D$
   $DR \to \lambda$

   Examples. Note that each blue string has $2n-1$ $B's$ and n^2 $a's$ for some n.

   $S \Rightarrow \lambda$
   $S \Rightarrow FABR \Rightarrow FaBAR \Rightarrow FaBR \Rightarrow DaBR \Rightarrow aDBR \Rightarrow aDR \Rightarrow a$
   $S \Rightarrow FABR \Rightarrow FaBAR \Rightarrow FaBR \Rightarrow FABBaBR \Rightarrow FaBABaBR$
   $\Rightarrow FaBaBAaBR \Rightarrow FaBaBaABR \Rightarrow FaBaBaaBAR \Rightarrow FaBaBaaBR$
   $\Rightarrow DaBaBaaBR \Rightarrow aDBaBaaBR \Rightarrow aDaBaaBR \Rightarrow aaDBaaBR$
   $\Rightarrow aaDaaBR \Rightarrow aaaDaBR \Rightarrow aaaaDBR \Rightarrow aaaaDR \Rightarrow aaaa$

   Do you see how this works? It is based on the fact that the sum of consecutive odd numbers starting at 1 is always a square. For example $1 = 1^2$, $1 + 3 = 2^2$, and $1 + 3 + 5 = 3^2$. We can also generate a^9:

   $S \Rightarrow \cdots \Rightarrow FaBR \Rightarrow \cdots \Rightarrow FaBaBaaBR \Rightarrow \cdots \Rightarrow FaBaBaaBaaBaaaBR \Rightarrow \cdots \Rightarrow aaaaaaaaa$

4. Prove that the halting problem is undecidable. Hint: the proof given in our textbook looks different from the proof I gave in class, but it is essentially the same. You might find yet another proof in another textbook or on the internet.

   First let's look at the following "fallacious contradiction."

   (a) Every unicorn has a horn.

   (b) No unicorn has a horn.

   (a) holds because having a horn is part of the definition of a unicorn, while (b) holds by exhaustive search, resulting in no unicorn without a horn. Thus, there is a contradiction.

   Resolution: the two statements only contradict each other if at least one unicorn exists! Do you see that?

   Now, we're ready for the proof that HALTis undecidable.

   Recall that $\langle M \rangle$ is a string which names a TM M, and that HALT $= \{\langle M \rangle w : M$ halts with input $w\}$ We define the diagonal language $L_d = \{\langle M \rangle : \langle M \rangle \langle M \rangle \notin \text{HALT}\}$.

   Claim: L_d is not decidable. The proof of the claim is by contradiction. Assume that L_d is decidable. Then L_d is accepted by some TM M_d. Then, for any TM M,

$$\langle M \rangle \in L_d \iff \langle M \rangle \langle M \rangle \notin \text{HALT by definition of } L_d \quad (1)$$

$$\langle M \rangle \in L_d \iff \langle M_d \rangle \langle M \rangle \in \text{HALT by definition of } M_d \quad (2)$$

$$\langle M_d \rangle \in L_d \iff \langle M_d \rangle \langle M_d \rangle \notin \text{HALT by universal instantiation of (1)} \quad (3)$$

$$\langle M_d \rangle \in L_d \iff \langle M_d \rangle \langle M_d \rangle \in \text{HALT by universal instantiation of (2)} \quad (4)$$

   Since M_d exists, equations (3) and (4) contradict each other. We conclude that L_d is not decidable.

   We now reduce L_d to the complement of HALT. Let $R(\langle M \rangle) = \langle M \rangle \langle M \rangle$ for every Turing machine M. R is a reduction of L_d to the complement of HALT. Since L_d is not decidable, the complement of HALT is not decidable. Since the complement of any decidable language is decidable, HALT is not decidable.