Assignment 6

Aleksandr Salo

Due October 28, 2014

1 Textbook exercises

5.9 Let $T = \{< M > \mid$ M is a TM that accepts $w^{\mathcal{R}}$ whenever it accepts w$\}$. Show that T is undecidable. Use Reduction.

Proof (by reduction to $A_{TM} \leq_m T$)).

1. Language A is **mapping reducible** to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w : w \in A \Leftrightarrow f(w) \in B$ *(Sipser Def. 5.20)*.
2. The following machine F computes a reduction f that takes input of the form $< M, w >$ and returns output of the form $< M' >$.
   $F =$ "on input $< M, w >$,
   (a) Construct the following machine M':
   ```
   M' = "On input $x$:
   1. Run M on $x^{\mathcal{R}}$
   2. If M accepts,
   2.1 Run M on $x$ and ACCEPT if M does
   3. REJECT"
   ```
   (b) Output $< M' >$."
   Note the following about the language of the machine M':

M, w	$L(M')$
M accepts w	$L(M') = \{L(M), L(M)^{\mathcal{R}}\}$
M rejects w	$L(M') = \emptyset$
M loops w	$L(M') = \emptyset$

   Thus, only if M accepts w, that M' would accept w and $w^{\mathcal{R}}$.
   Formally: $< M, w > \in A_{TM}$ iff $< M' > \in T$
3. The existence of a valid computable (trivially we can construct and simulate TM) reduction function proves that A_{TM} is mapping reducible to T. That in turn proves that T is **undecidable** language. *(Sipser corollary 5.23)*.

6.23 Show that the function K(x) is not a computable function. Hint: you could use a reduction for this problem (as usual), but it is much easier to prove if you consider what you would be able to do if K(x) was computable, along with the idea that some strings of every length are incompressible (Theorem 6.29).

1. Assume that computable function $K(x)$ exists (for the sake of following contradiction), and M is a TM that computes it.
2. Let us now construct the following machine M':

```
$M'$ = "on input $n$,
0. interpret $n$ as a natural number.
1. for i = $1, 2, 3 ... \infty$:
1.1. for each string s of length i
1.1.1. compute $k = K(s)$
1.1.2. if $k \geq |s| = n$, HALT"
```

3. Note, that incompressible strings of every length exist. *Sipser, theorem 6.29*
4. Thus we know that M' would eventually halt when finds s s.t. $K(s) \geq n$.
5. To describe that construction we need to implicitly characterize:
   (a) The machine itself.
   (b) n for the halting condition.
6. Note that $< M', n >$ is a description of some string s that we can derive by simulating M' on n on the universal Turing Machine.
7. By *Sipser theorem 6.24*: $K(s) \leq | < M', n > | + c$
8. Note that the length of $< M', n >$ can be no more than: $< M' > + log_2(n) + c_1$, where c_1 is a constant due to the separation issue. Yet $< M' >$ is constant in any reasonable programming language (or as an TM description), thus the total length of the construction can not exceed $log_2(n) + c_2$.
9. Now we have the following inequality: $n \leq K(w) \leq | < M', n > | = log_2(n) + c_3$, or, refining: $n \leq log_2(n) + c_3$, which is obviously not true for some sufficiently large n. That is a contradiction. We described the machine computing string of complexity greater than the description of the machine.
10. This logical contradiction allows us to conclude that Kolmogorov complexity is uncomputable function.

6.21 Show how to compute the descriptive complexity of strings K(x) with an oracle for $HALT_{TM}$.

1. Let us construct machine M that computes $K(x)$ with the oracle for $HALT_{TM}$:

```
$M$ = "On input x,
1. for i = 1 ... $|x|$+1:
1.1. for every string of length $i$ try parse it as $< M, w >$ and if succeed:
1.1.1. ask oracle whether $< M, w > \in HALT_{TM}$ and if "yes":
1.1.1.1. run $M$ on $w$ until it halts
1.1.1.1.1. if $x$ is left on the tape, output $i$"
```

   Note, that we can use the lexicographic order to iterate over all the strings of given length.

2. Thus this TM will compute Kolmogorov complexity for any input string x.

4 Consider the languages $HALT_{TM}$ (from Theorem 5.1) and E_{TM} (from Theorem 5.2). Prove the following statement: $E_{TM} \leq_T HALT_{TM}$

Solution 1

1. A is Turing reducible to language B, if A is decidable relative to B.
2. By *Sipser example 6.19*, we know that $E_{TM} \leq_T A_{TM}$.
3. By *Sipser example 5.24*, we know that $A_{TM} \leq_m HALT_{TM}$.
4. By *Sipser definition 6.20*, we know that Turing reducibility is more general than mapping reducibility.
5. Considering all facts together we easily derive that $E_{TM} \leq_T HALT_{TM}$.

Solution 2

1. A is Turing reducible to language B, if A is decidable relative to B.
2. Let us first prove that $A_{TM} \leq_T HALT_{TM}$.
3. Let us construct the procedure for solving the A_{TM} with the help of oracle for $HALT_{TM}$.
   $T^{HALT_{TM}}$ = "On input $< M, w >$,
   1. Ask oracle for $HALT_{TM}$ if M halts on w.
   2. If NO, REJECT;
   3. If YES, run M on w and do what it does."
   This routine clearly decides $A_T M$ because it accepts w iff M halts and accepts, and we know for sure that M would halt if we run it because we previously consulted with the oracle.
4. Now, by *Sipser example 6.19*, we know that $E_{TM} \leq_T A_{TM}$.
5. Taking into account those two statements we derive that $E_{TM} \leq_T HALT_{TM}$.

Solution 3

1. A is Turing reducible to language B, if A is decidable relative to B.
2. Let us construct the procedure for solving the E_{TM} with the help of oracle for $HALT_{TM}$.
   $T^{HALT_{TM}}$ = "On input $< M >$,
   1. Construct TM N:
   N = "On any input:
   (a) Run M in parallel on all strings in Σ^*.
   (b) If M accepts any of these strings, HALT"
   2. Ask oracle whether $< N, \epsilon > \in HALT_{TM}$
   3. If answer is NO, ACCEPT; 4. If YES, REJECT"