Theory of Computation — CSE 105

Computability Theory
Solution to Homework 3

1. Let $A = \{\langle M\rangle |\ M$ is a DFA which doesn't accept any string containing an odd number of 1s$\}$. Show that A is decidable.

Solution: We can prove that this language is decidable by constructing a Turing machine M that decides A.

M will try to accept all strings of length less than or equal to $2k$, where k is the number of states in the DFA. If none of these strings has an odd number of 1s, then we claim that the DFA doesn't accept any strings with an odd number of 1s. The reason is as follows.

By restricting the size of the input to be less than or equal to twice the number of states in the DFA, the only strings we are eliminating from consideration are those for which there is a state which is entered more than twice. (That is, if s is a string with more than $2k$ symbols, then there must be some state q which appears more than twice in the execution of A on s).

The reason we may exclude such strings from consideration is as follows: let s be the smallest string which has length greater than or equal to $2k$. Then there must be some state q which is entered three or more times (see figure 1). Let $s = abcd$, where:

- a is the sequence of symbols accepted between the start state and the first occurence of state q,
- b is the sequence of symbols accepted between the the first occurence of state q and the second occurence of state q,
- c is the sequence of symbols accepted between the the second occurence of state q and the third occurence of state q, and
- d is the sequence of symbols accepted between the the third occurence of state q and the accept state.

Let A, B, C, and D be the number of 1s in a, b, c, and d, respectively. Then, the following claim holds.

Claim: There exist a string t such that $|t| < |s|$, t contains an odd number of 1s, and $t \in L(A)$.

Proof: $A + B + C + D$ is equal to the number of 1s in s. So, $A + B + C + D$ is odd. Then, there are four cases to consider depending whether B and C are odd or even numbers:

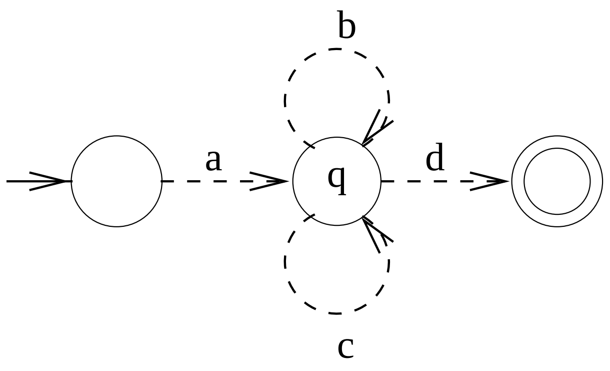

Figure 1: Example showing how the strings a, b, c, and d are related to s.

- B and C are both odd. In this case, $B + C$ is even. So $A + D$ must be odd. So, let $t = ad$.
- B is odd and C is even. In this case, $B + C$ is odd. So $A + D$ must be even. So, let $t = abd$.
- B is even and C is odd. In this case, $B + C$ is odd. So $A + D$ must be even. So, let $t = acd$.
- B and C are both even. In this case, $B + C$ is even. So $A + D$ must be odd. So, let $t = ad$.

In all cases, t has an odd number of 1s and $|t| < |s|$.

Theorem 1 *If s is a string accepted by a DFA G such that $s \geq 2k$ and s contains an odd number of 1s, then there is a string t such that $t \in L(G)$, $|t| < 2k$, and t contains an odd number of 1s.*

Proof: It follows easily from claim above.

Now that we have proved theorem 1, we are finally in a position to give a description of a Turing machine T_A that decides A.

T_A = "Given a string, s,

1. Check if s is of the form $\langle G \rangle$ for some DFA G and em reject if it's not.
2. For every string $s' \in \sum^\star$ which has length less than or equal to twice the number of states in G and which contains an odd number of 1's.
   (a) Run G on s'. If G accepts s', then *reject* s.
3. *Accept* s.

2. Let $L_{sub} = \{\langle T_1, T_2\rangle |\ T_1$ and T_2 are Turing Machines and $L(T_1) \subseteq L(T_2)\}$. Show that L_{sub} is undecidable.

Solution: We will proceed by contradiction. Assume that L_{sub} is recursive. Let T_{sub} be a TM that decides L_{sub}. We will now show that the undecidable halting language $L_H = \{\langle T, x\rangle |\ T$ halts on input $w\}$ is decidable to obtain the desired contradiction.

With the help of T_{sub}, we will now design a Turing machine T_H to decide L_H. The description of T_H is given below.

T$_H$ = "Given a string, x,

1. Check if x is of the form $\langle T, w\rangle$ for some TM T and input $w \in \Sigma^*$.
2. If not, reject and halt.
3. If true, construct the description of a TM T_1 given T and w such that

   T$_1$ = "Given a string, y,

   (a) Simulate (or run) T on w

   (b) Accept y if and only if T halts on w."

   Comment: The description of T_1 depends on T and w which in turn depend on x. Also observe that $L(T_1) = \Sigma^*$ if T halts on w and $L(T_1) = \emptyset$ if T does not halt on w. Also notice that T_H is only constructing the description of T_1 rather than running it.
4. Construct a TM T_2 such that T_2 accepts no input string.

   Comment: $L(T_2) = \emptyset$.
5. Input $\langle T_1, T_2\rangle$ to the TM T_{sub}.
6. If T_{sub} accepts, reject and halt.
7. If T_{sub} rejects, accept and halt."

Claim: T_H decides L_H.

Proof: We will show that T_H will accept for inputs $x \in L_H$ and will reject for inputs $x \notin L_H$.

- **case** $x \in L_H$**:** Since $x \in L_H$, x must be of the form $\langle T, w\rangle$ for some TM T and input w. Moreover T must halt on w. When such a x is given as input to T_H, x will pass the test in step 1 and T_H will proceed to step 3. In step 3, the TM T_1 constructed by T_H must be such that $L(T_1) = \Sigma^*$ since T halts on w. In step 4, T_2 will be constructed such that $L(T_2) = \emptyset$. In step 5, since $L(T_1) = \Sigma^*$ and $L(T_2) = \emptyset$, T_{sub} will reject and in step 7 T_H will accept as required.
- **case** $x \notin L_H$**:** Since $x \notin L_H$, either x is not of the form $\langle T, w\rangle$ for any TM T and input w or x is of the form $\langle T, w\rangle$ for some TM T and input w and T does not halt on w. In the first case, T_H will reject since x fails the test in step 1. If x is of the form $\langle T, w\rangle$ for some

TM T and input w and T does not halt on w, then x will pass the test in step 1 of T_H and the control passes to step 3. In step 3, a TM T_1 will be constructed such that $L(T_1) = \emptyset$ since T does not halt on w. The TM T_2 constructed in step 4 is such that $L(T_2) = \emptyset$. Since $L(T_1) \subseteq L(T_2)$, T_{sub} will accept in step 5. Finally, in step 6, T_H will reject as required.

Thus, T_H decides L_H. This is a contradiction to the fact that L_H is undecidable. Hence, L_{sub} is undecidable.

3. Give an example in the spirit of the recursion theorem of a program in a real programming language that prints itself out.

Solution: Here is a program in C:

```
#include <stdio.h>
char*a = ";                                              \n\
void                                                     \n\
main()                                                   \n\
{                                                        \n\
char* b;                                                 \n\
                                                         \n\
printf(\"#include <stdio.h>\\nchar*a = \\\"\");          \n\
for (b=a;*b;b++)                                         \n\
 {                                                       \n\
    if (*b=='\\\"' || *b=='\\\\')                        \n\
        putchar('\\\\');                                 \n\
    if (*b=='\\n')                                       \n\
    {                                                    \n\
        printf(\"\\\\n\\\\\");                           \n\
    }                                                    \n\
    putchar(*b);                                         \n\
  }                                                      \n\
  putchar('\\\"');                                       \n\
  puts(a);                                               \n\
}                                                        \n\
";
void
main()
{
char* b;

printf("#include <stdio.h>\nchar*a = \"");
for (b=a;*b;b++)
 {
    if (*b=='\"' || *b=='\\')
        putchar('\\');
    if (*b=='\n')
    {
        printf("\\n\\");
    }
    putchar(*b);
  }
  putchar('\"');
  puts(a);
}
```

How this program works:

1. Create a string, a, whose contents are described below.
2. Print out the code before the declaration of a.
3. Print out the string a, placing a slash before each quote in a, and a slash before each carriage return in a, and a slash before each slash in a.
4. Print out a single quote.
5. Print a.

   Comment: a contains the code which performs instructions 2,3,4,5.

This program prints the string, a, twice. The first time it prints a, it prints it as though it were a string. The second time, it prints it normally. In this sense, the program is roughly equivalent to the following english instructions:

"Print the preceding sentence twice, the first time in quotes."

Print the preceding sentence twice, the first time in quotes.

The interested reader may wish to consider how to modify this program in order to make it equivalent to the following English instructions:

Print the following sentence twice, the second time in quotes:

"Print the following sentence twice, the second time in quotes."