Cpt S 516 Solutions to Homework # 3

1. (standard, 10pt) Show that the following function f is not computable: for all natural number n, if M_n (the n-th Turing machine) accepts a regular language, then $f(n) = 1$, else $f(n) = 0$.

Suppose that f is computable, by a TM M_f. Then, one can construct a TM M_R as follows:
Input a TM M;
compute the index n of M;
Run M_f on n;
M_f outputs 1 then say yes;
M_f outputs 0 then say no.
This TM M_R is able to answer (yes/no) whether a TM accepts a regular language. This is a contradiction, since it is undecidable whether a TM accepts a regular language.

2. (standard, 10pt) Show that there is a totally computable function e such that for any $x, y, z \in N$, $f_{e(x,y)}(z) = f_x(z) + f_y(z)$. Hint: f_i denotes the function computed by the i-th TM.

Define $F(x, y, z) = f_x(z) + f_y(z)$. F is a computable function (why?) Using s-m-n.

3. (standard, 10pt) Show that if $f(x, y)$ is a computable function, then so is $f(f(x, y), y)$.

Assume M_f computes $f(x, y)$. Then $f(f(x, y), y)$ is computed by:
input x, y;
Run M_f on x, y;
Collect the output (denoted by z);
Run M_f again on z and y;
Collect the output and print it out.

4. (standard, 10pt) Let $f(x)$ be a totally computable function. Define g as follows:
$g(0) = 1$;
$g(n + 1) = f(g(n))$.

Show that g is also a totally computable function.

Let M_f totally compute f. (i.e., M_f always halts) Then build M_g that totally computes g as follows:

input n;
if $n == 0$ then output 1 and done.
Let $z := 0$;
(*) Run M_f on z;
Assign z to be the output of M_f;
n–;
if $n == 0$ then output z and done.
goto (*);

5. (standard, 10pt) A word is a *double word* if it is in the form of ww for some w. Let L be a recursive language. Let N be natural numbers. Define a function $f : N \to N$ such that, for each $n \in N$, if there is a double word in L of length n, then $f(n) = 1$, else $f(n) = 0$. Show that f is a totally computable function.

Let M_L recognize L (since L is recursive). Construct the folowing M_f to totally compute f:

input n;
if n is odd then output 0 and done.
Let $n = 2k$ for some k.
For each word w with length k do:
  Run M_L on ww;
  If M_L says yes, then output 1 and done.
Output 0 and done.

6. (a little tricky, 10pt) A word is a *double word* if it is in the form of ww for some w. Let L be a regular language. Let N be natural numbers. Define a function $f : N \to N$ such that, for each $n \in N$, if there is a double word in L of length $\geq n$, then $f(n) = 1$, else $f(n) = 0$. Show that f is a totally computable function.

Let M_L be a FA accepting L. For each n, we con construct a FA M_n accepting $\{w : ww \in L, |ww| \geq n\}$:

input w;

make two copies of M_L: M_1 and M_2;

scan the w from the initial state of M_L while running M_1; in parallel to this scanning,

scan the w from a guesed state of M_L while running M_2; in parallel to this scanning,

scan the w to make sure that the length is $\geq n/2$ (store n in the finite control).

At the end of w, make sure that M_1 is at the guessed state and M_2 is at the final state of M_L (otherwise M_n crashes)

We know that it is decidable to test whether a FA M_n accepts a nonempty language – denote the algorithm by A.

Now, below is a TM M_f that totally computes f:

input n;

construct M_n in above;

run A on M_n;

A says yes (i.e., $L(M_n)$ nonempty) then output 1;

A says no then output 0.

7. (not hard, 10pt) Let $F(x, y, z, x', y', z')$ be any Presburger formula over six free variables. Define $G(n, x, y, z, x', y', z')$ be a formula in the following form:

$$\exists x_1, \cdots, x_{n-1}, y_1, \cdots, y_{n-1}, z_1, \cdots, z_{n-1}.$$

$$F(x, y, z, x_1, y_1, z_1) \wedge F(x_1, y_1, z_1, x_2, y_2, z_2) \wedge \cdots \wedge F(x_{n-1}, y_{n-1}, z_{n-1}, x', y', z').$$

We say that F *terminates* iff $\exists n.G(n, 0, 0, 0, 1, 1, 1)$ holds. Show that it is undecidable whether F terminates. (From this result, one can show that G is not Presburger)

Let M be a two-counter machine. A configuration of M is a triple (s, y, z) of a state, counter y and counter z. M can be represented as a graph where each edge denotes a counter instruction. For instance, an instruction like:

s : if $y == 0$ then goto s'

can be recorded as a Presburger formula over x, y, z, x', y', z':

$$x = s \wedge x' = s' \wedge y = 0 \wedge y' = y \wedge z' = z$$

We use $F(x, y, z, x', y', z')$ to denote the disjunction over all of the Presburger formulas obtained from the instructions. One can assume that M starts with configuration $(0, 0, 0)$ and ends with configuration $(1, 1, 1)$. Clearly, $\exists n.G(n, 0, 0, 0, 1, 1, 1)$ iff M halts. The undecidability follows.

8. (not hard, 10pt) Let n be a natural number. A *linear function* is a total function $f : Z^n \to Z$ (Z denotes the integers) such that there is a Presburger formula F satisfying for all $x_1, \cdots, x_n, y \in Z$,

$F(x_1, \cdots, x_n, y)$ holds iff $f(x_1, \cdots, x_n) = y$.

A LP problem is defined as below.

Given: a number n, a linear function f, and a Presburger formula $C(x_1, \cdots, x_n)$.
Question: Is there a number K $(< +\infty)$ such that (1). for all $x_1, \cdots, x_n \in Z$, $K \geq f(x_1, \cdots, x_n)$, and (2). for some $x_1, \cdots, x_n \in Z$, $K = f(x_1, \cdots, x_n)$.

Show that the LP problem is decidable.

We rewrite the Questionpart into a Presburger formula as follows:
$\exists K.$
$\forall x_1, \cdots, x_n \in Z, \exists z, K \geq z \wedge P(x_1, \cdots, x_n, z)$
$\wedge$
$\exists x_1, \cdots, x_n \in Z, P(x_1, \cdots, x_n, K).$
Then, decidability follows from the fact that Presburger's satisfiability is decidable.

9. (hard, 20pt) We use x and its subscripts $x_1, x_2, \cdots$ to denote integer variables. A *linear constraint* is a formula in the following form:

$$\sum_{1 \leq i \leq n} a_i x_i > a$$

where n is a natural number and the a's are integers. A *mod constraint* is a formula in the following form:

$$x \bmod a = b$$

where $a \neq 0$ and b are integers. A *linear formula* F is defined by the following grammar:

$$F ::= C | F \wedge F | \neg F$$

where C is a linear constraint or a mod constraint. Clearly (why?), a linear formula is also a Presburger formula. We say that F is a linear formula over $x_1, \cdots, x_n$ if $x_1, \cdots, x_n$ are all the integer variables appearing in F. Show that for any Presburger formula $P(x_1, \cdots, x_n)$ over free variables $x_1, \cdots, x_n$, there is a linear formula F over $x_1, \cdots, x_n$ such that

for all integers $x_1, \cdots, x_n$, $P(x_1, \cdots, x_n)$ holds iff $F(x_1, \cdots, x_n)$ holds.

First, observe that

$$\sum_{1 \le i \le n} a_i x_i \# a$$

with $\# \in \{=, \neq, \geq, \leq, >, <\}$ is expressible as a linear formula. Example: $2x - y = 8$ can be expressed as

$$\neg(2x - y > 8 \vee -2x + y > -8).$$

Second, observe that

$$\sum_{1 \le i \le n} a_i x_i \bmod d = a$$

is expressible as a linear formula. Example: $2x - y \bmod 2 = 1$ can be expressed as $((x \bmod 2 = 0 \wedge y \bmod 2 = 0 \wedge false) \vee (x \bmod 2 = 0 \wedge y \bmod 2 = 1 \wedge true) \vee (x \bmod 2 = 1 \wedge y \bmod 2 = 0 \wedge false) \vee (x \bmod 2 = 1 \wedge y \bmod 2 = 1 \wedge true))$.

So, we may assume that linear constrains and mod constraits are defined in the above general forms

$$\sum_{1 \le i \le n} a_i x_i \# a$$

and

$$\sum_{1 \le i \le n} a_i x_i \bmod d = a$$

Next, we prove by an induction on the structure of a Presburger formula P; in below, we shall use $[P]$ to denote the representation of P in terms of a linear formula (i.e., the F):

case 1. if P is atomic; i.e., $t + t = t$, then, P is already a linear formula – in this case, take $[P]$ as P;

case 2. if P is $P_1 \wedge P_2$, then take $[P]$ as $[P_1] \wedge [P_2]$ which is still a linear formula.

case 3. if P is $\neg P_1$, then take $[P]$ as $\neg[P_1]$ which is still a linear formula.

case 3. if P is $\exists x.P_1$, then a lot of things need to do. Notice that $[P_1]$ being a linear formula can always written into a CNF – a disjunction

$Q_1 \vee \cdots \vee Q_m$

where each Q_i is a conjunction

$l_1 \wedge \cdots \wedge l_n$

where each l_j is either a linear constraint or a mod constraint. Hence, since $\exists x.P_1$ is equivalent to $\exists x.Q_1 \vee \cdots \vee \exists x.Q_m$, we can take $[P]$ as $[\exists x.Q_1] \vee \cdots \vee [\exists x.Q_m]$. In below, we need only to show how to obtain $[\exists x.Q]$ where the Q is int he form of $l_1 \wedge \cdots \wedge l_n$. Example: consider a Q in the form of

$2x - y \geq 5 \wedge 3x + 2y < 10.$

we first change it into:

$6x \geq 3y + 15 \wedge 6x < -4y + 20$

Then, $\exists x.Q$ is true iff $-4y + 20 > 3y + 15$ and one of the followings is true:

- $-4y + 20 - (3y + 15) \geq 6,$
- $-4y+20-(3y+15) < 6$ and for some $0 \leq j < 6$, $3y+15+j \bmod 6 = 0$.

It is not hard to express $\exists x.Q$ into a linear formula. Hence, each $[\exists x.Q]$ can be obtained.

10. (10pt, standard) Let $L = ((abc)^*(ab)^*)^*$. Show me a Presburger formula that defines $\#(L)$ (and hence L is semilinear).

Let $P(n_a, n_b, n_c)$ be a Presburger formula defined as below:

$$\exists k_1 \geq 0 \exists k_2 \geq 0. n_a = k_1 + k_2 \wedge n_b = k_1 + k_2 \wedge n_c = k_1.$$

Then, $\#(L) = \{(n_a, n_b, n_c) : P(n_a, n_b, n_c)\}$.

11. (10pt, hard) Keep in mind that context free languages are semilinear, and also the result in the previous problem. Now, let us look at a more difficult problem. Consider a PDA M that never reads input. So M starts from the initial state with an empty stack can, following its instructions, push/pop symbols to/from the stack. I have a meter that does the following. Whenever M pushes an a the meter charges two dollars, whenever M pushes a b the meter gives a credit of three dollars. We say that M is stable if during any time of any execution, the meter (initially 0) stores money x satisfying

$x \geq 0$ and $3x \geq n + m$ where n and m are the number of symbols a and b in the stack respectively at the time. Show that it is decidable whether M is stable.

Build a PDA M' as follows. M' faithfully simulates M. Additionally, whenever M pushes (resp. pops) a, M' reads an input symbol p_a (resp. q_a). Whenever M pushes (resp. pops) b, M' reads an input symbol p_b (resp. q_b). M' accepts if it is at the end of the input. We use L' to denote the context-free language accepted by M'. L' is also a semilinear language. Define $L = \{w : -2\#_{p_a} + 3\#_{p_b} < 0 \vee 3(-2\#_{p_a} + 3\#_{p_b}) < (\#_{p_a} - \#_{q_a}) + (\#_{p_b} - \#_{q_b})\}$. (Notice that, in the problem statement, money x is $-2\#_{p_a} + 3\#_{p_b}$, n is $\#_{p_a} - \#_{q_a}$, m is $\#_{p_b} - \#_{q_b}$.) Notice that L is a commutative semilinear language. So, $L \cap L'$ is also a semilinear language. Notice that M is not stable iff $L \cap L' \neq \emptyset$. The result follows from the decidability of testing emptiness of a semilinear set $L \cap L'$.