CS3 Computability and Intractability (2001-2002)
Exercise Sheet 2

The deadline for this coursework is noon on Thursday 21 February. Please submit your solutions directly to your tutor. Solutions submitted after the deadline, but before noon on Monday 25 February, will have their credit reduced by one third. No credit will be given for solutions submitted after that date. Please note that multiple submissions are not allowed. The marks for questions are not always related to their length or difficulty. Answers that are not completely correct but show relevant reasoning will be awarded partial credit (depending on how much progress is shown).

1. The Turing machine M_{sort} has states $Q = \{q_0, q_1, q_2, q_3\}$, initial state q_0, input alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$, tape alphabet $\Gamma = \{\texttt{0}, \texttt{1}, \textvisiblespace\}$, and transition function given by the tuples

$$\begin{array}{l}\{\ (q_0, \texttt{0}, q_0, \texttt{0}, R),\ (q_0, \texttt{1}, q_0, \texttt{1}, R),\ (q_0, \textvisiblespace, q_1, \textvisiblespace, L),\\ \ \ (q_1, \texttt{0}, q_2, \texttt{0}, L),\ (q_1, \texttt{1}, q_1, \texttt{1}, L),\\ \ \ (q_2, \texttt{0}, q_2, \texttt{0}, L),\ (q_2, \texttt{1}, q_3, \texttt{0}, R),\\ \ \ (q_3, \texttt{0}, q_0, \texttt{1}, R)\ \}\end{array}$$

   Use the encoding of Note 6 to code M_{sort} as a word over $\{\texttt{0}, \texttt{1}, \texttt{B}, \texttt{*}\}$.

   The file `/home/cs3/ci/tm/univ.tm` contains the universal Turing machine M_u, as described in Note 6, in a form suitable for input to the Simulator. Use the Simulator to follow the progress of M_u simulating the computation of M_{sort} on input `0101`. Submit a listing of the output produced by the Simulator. Make sure you select a relatively fast simulation speed. Note that the simulation will always end (if it ends at all) with the input being rejected. [*6 marks*]

2. For this exercise let $x_1, x_2, \ldots, x_n$ be variables over $\mathbb{N}$ (i.e., the natural numbers $0, 1, 2, \ldots$). We will consider polynomial functions $P(x_1, x_2, \ldots, x_n)$ with integer coefficients (e.g., $3x_1^3x_2 - 5x_1x_2x_3^2 + 25x_1x_2 + 42$). We say that $P(x_1, x_2, \ldots, x_n)$ has a root in $\mathbb{N}^n$ if and only if $P(b_1, b_2, \ldots, b_n) = 0$ for some $(b_1, b_2, \ldots, b_n) \in \mathbb{N}^n$. For example $G(x_1, x_2) = x_1 - 2x_2^2 - 3$ has infinitely many such roots, just consider $(2m^2 + 3, m)$ for any natural number m. As another example $G(x_1, x_2, x_3) = 2x_1^2x_2 - 10x_1x_3^3 + 1$ has no roots since $2x_1^2x_2 - 10x_1x_3^3$ is even no matter what values we choose for the variables. *Hilbert's tenth problem*[1] is:

   Input: A polynomial $P(x_1, x_2, \ldots, x_n)$ with integer coefficients.
   Output: 'Yes' if $P(x_1, x_2, \ldots, x_n)$ has a root in $\mathbb{N}^n$ and 'no' otherwise.

[1] So called because it was the tenth of a list of twenty three problems that Hibert proposed in his address to the International Congress of Mathematicians in 1900. For a very interesting account see J.J. Gray, *The Hilbert Challenge*, Oxford University Press (2000).

It was proved on 1972 that Hilbert's tenth problem is unsolvable[2], i.e., there is no algorithm for it, contrary to the expectations of Hilbert and his contemporaries.

(a) Given a polynomial function $P(x_1, x_2, \ldots, x_n)$ we call a natural number B a *root bound* for the polynomial provided that $P(x_1, x_2, \ldots, x_n)$ has a root in $\mathbb{N}^n$ if and only if it has a root $(b_1, b_2, \ldots, b_n) \in \mathbb{N}^n$ with $b_i \leq B$, for $1 \leq i \leq n$. For example, 3 is a root bound for $F(x_1, x_2)$ as defined above while 0 is a root bound for $G(x_1, x_2, x_3)$.

Is there a Turing machine transducer that takes as input a polynomial function (in any reasonable format) and computes a root bound for the input? Prove your claim. [*5 marks*]

(b) It can be shown that if we are willing to accept roots from $\mathbb{R}^n$ (real numbers) rather than insisting on $\mathbb{N}^n$ then there is an algorithm for deciding if a given polynomial has a root. (One possible algorithm is a generalization of Sturm's algorithm mentioned in NOTE 1.) We can easily apply such an algorithm to a system of several polynomials (requiring common roots) by replacing the polynomials with a single one consisting of the sum of their squares.

Many subsets of the real numbers can be characterized as the set of values that one of the variables (say x_1) takes over all the solutions to a system of polynomial equations. For example, the non-negative real numbers are all the x-values to the solutions of $x - y^2 = 0$ while the set of real numbers between 0 and 1 inclusive (i.e., the interval $[0, 1]$) is given by all the x-values of $x^2 + y^2 - 1 = 0$.

Is it possible to characterize the natural numbers by a method as described in the preceding paragraph? Justify your answer. [*5 marks*]

3. Let L_1, L_2, and L' be languages over the same alphabet Σ, and suppose that L_1 and L_2 are recursive, and that L' is recursively enumerable (r.e.). Which of assertions (i)–(vi) are true? If the assertion is true, prove that it is true; if it is false, provide a counterexample. Note that there are not six *independent* questions here; for example, an affirmative answer to (ii) would immediately imply an affirmative answer to (i).

   (i) $L_1 - L'$ is r.e.;
   (ii) $L_1 - L'$ is recursive;
   (iii) $L' - L_1$ is r.e.;
   (iv) $L' - L_1$ is recursive;
   (v) $L_1 - L_2$ is r.e.;
   (vi) $L_1 - L_2$ is recursive.

   [*12 marks*]

[2] Y. V. Matiyasevič building on the work of many others; see M. Davis, 'Hilbert's tenth problem is unsolvable', *American Mathematical Monthly*, **80** (1973), pp. 233-269. Shorter proofs are now known.

4. (a) The *Blank-tape halting problem* is defined by

$$L_{\text{bt}} = \{\langle M \rangle \mid M \text{ halts on input } \epsilon\},$$

i.e., this is the language of encodings of binary Turing machines that halt on the blank tape. It can be shown that L_{halt} reduces to L_{bt}. Assuming the last fact, prove that the language

$$\widehat{L}_{\text{bt}} = \{\langle M \rangle \mid M \text{ does } \textit{not} \text{ halt on input } \epsilon\}$$

is not recursively enumerable. (Recall that $\langle M \rangle$ represents the encoding of the binary Turing machine M.) [*2 marks*]

(b) Prove that the *Turing machine equivalence problem* is undecidable. That is, show that

$$L_{\text{eq}} = \{\langle M_1 \rangle \$ \langle M_2 \rangle \mid M_1 \text{ and } M_2 \text{ accept the same language}\},$$

the language of encodings of pairs of equivalent binary Turing machines, is not recursive. [*5 marks*]

(c) Now let

$$L_{\text{ineq}} = \{\langle M_1 \rangle \$ \langle M_2 \rangle \mid M_1 \text{ and } M_2 \text{ accept different languages}\}.$$

Prove that neither L_{eq} nor L_{ineq} is recursively enumerable. [*5 marks*]

Kyriakos Kalorkoti, January 2002