Solutions to Assignment #2

1. If we have algorithms for computing f and g effectively, then an algorithm for computing their composite could look like:

Given $n \in \mathbb{N}$, use the algorithm for g to compute $g(n)$. Take $g(n)$ and use the algorithm for f to compute $f(g(n))$.

2. One way to think of the real numbers would be as decimal expansions: an integer followed by an infinite string of digits. There is also the issue of sign – is the number positive or negative. This could be given as a single bit. So being ≥ 0 is decidable – if the sign bit is positive then the number is ≥ 0 otherwise not. Being > 0 is a little more problematic: If the number is positive i.e. has sign bit positive, you need to look at the decimal expansion. If the integer part is > 0 then the number is > 0 otherwise you have to look at the decimal part. If you ever see a non-zero number then you know the number is > 0. But in a finite time you will not know you are looking at 0 or not so this isn't decidable.

3. As discussed in class, we consider strings of symbols from L_A and describe an algorithm by induction for recognizing terms.

(1) If the string is of length 1 and is either 0 or a variable then it is a term. Otherwise not.

(2) If the string is of length > 1 and is not of the form $S(\sigma)$ or (σ) where σ is a string then it is not a term.

(3) If it is of the form $S(\sigma)$ then it is a term iff σ is a term which is determined by induction.

(4) We now need to consider the case where the sequence looks like $(\ \delta\ )$. Let's digress and introduce the notion of bracket count for a sequence:

If we have symbols $s_1 s_2 \cdots s_k$ define the bracket count inductively by $n_1 = \begin{cases} 1 & \text{if } s_1 = (\\ -1 & \text{if } s_1 =) \\ 0 & \text{otherwise.} \end{cases}$

$$n_{\ell+1} = \begin{cases} n_\ell + 1 & \text{if } s_{\ell+1} = (\\ n_\ell - 1 & \text{if } s_{\ell+1} =) \\ n_\ell & \text{if } s_{\ell+1} \text{ is anything else.} \end{cases}$$

Now there are many things one could prove: for instance if $s_1 \cdots s_k$ is a term and $n_1 \cdots n_k$ is the bracket count then $n_k = 0$, $n_i \geq 0$ for all i etc.

We are considering the case where we have a sequence

$(\ s_2 s_3 \cdots s_{k-1}\ )$ so the bracket count starts $1 \cdots$ and hopefully ends with 0. We are looking for the main binary operation. To find it, if it exists, look for the first binary operation with a bracket count of 1 - if there is none then the sequence is not a term. If there is one then we have $(\mu\ s\ \tau)$ where s is $+$ or $\times$ and this is a term if μ and τ are terms which we determine recursively.

4. a) $x < y$: $\exists z\,(z \neq 0 \wedge x + z = y)$.

b) x is a perfect square : $\exists y\,(y \times y = x)$

c) x is a power of 2 : Informally you can say either $x = 1$ or (2 divides x and whenever y divides x, 2 divides y or $y = 1$) which can be written formally as

$$x = 1 \vee \Big(\exists z\,(S(S(0)) \times z = x) \wedge \forall y\,\forall z\,(y \times z = x \rightarrow (y = 1 \vee \exists u\, S(S(0)) \times u = y))\Big).$$