Solutions: Computability, etc.

1. (a) What is an algorithm?

   (b) Show that there are algorithms for

      i. deciding whether a natural number is prime;

      ii. finding the highest common factor of two natural numbers;

      iii. deciding whether a traditional syllogism (i.e. argument with two premisses and a conclusion, all of A,E, I or O, form) is valid;

      iv. deciding whether a string of symbols of your favourite system of the classical propositional calculus is a wff.

      v. deciding whether a given wff of your favourite system of the classical propositional calculus is a theorem (provable from no assumptions).

   (c) Is there an algorithm for deciding whether an arbitrary real number is greater than one?

(b) i. We can mechanically decide whether n is prime by the brute force method of trying to divide it in turn by every smaller number from 2 up (or better, by every number up to the first natural m such that $m^2 \geq n$), and seeing whether any of these numbers are indeed factors of n.

ii. Given m, n (where $m < n$), the inefficient brute force method would be to work downwards through the numbers from $k = m$ testing at each step to see whether k divides m and n. But, as Euclid knew, we can be more efficient: see http://en.wikipedia.org/wiki/Euclidean_algorithm.

iii. What do you mean,'I don't know about traditional syllogisms'? Pause for a bit of self-education at http://en.wikipedia.org/wiki/Syllogism.

Yes, that isn't very exciting is it? But one thing you will have picked up is that there is a finite number of kinds of traditional syllogism (256 in fact). So a computer can store a finite look-up table which says of each of the 256 kinds whether or not it is valid, giving us a brute-force algorithm to test validity. (Membership of finite sets of finite objects is always decidable.)

iv. Details will depend on the details of your favourite propositional language. But the key routine is going to be *something* like this, as check whether expression E is built up in the right kind of way to be a wff.

> Look at the first symbol in the current string E up for testing.
>
> 1. If it a propositional letter, then test to see if you have an atomic formula of your language: if so, output "ok", if not, output "no".
> 2. If E starts with a negation sign, i.e. is of the form $\neg E_1$, then restart the test on E_1.
> 3. If E starts with a right bracket or a left bracket without a matching right bracket at the end, then output "no".
> 4. If E starts with a left bracket and ends with a right bracket, then do the following:

(a) If E is of the form $(E_1 \circ E_2)$ where $\circ$ is a two-place connective, and E_1 has the same number of left-hand and right-hand brackets (perhaps zero), then start again running the whole test on both E_1 and E_2.

(b) If not, output "no".

Keep on going, running the test as many times as necessary on smaller and smaller chunks of E stopping if you get a "no", showing E isn't a wff: if you only get "ok"s, then E is indeed a wff.

v. By the completeness theorem, S is a theorem of the classical propositional calculus iff it is a tautology. So do a truth-table test to determine whether S is a tautology and hence a theorem.

(c) Real numbers aren't finite objects so can't in general be handled by finite processing rules which will always terminate in a finite amount of time. Thus, the real number r represented as $1.00000\ldots$ is greater than one iff eventually some digit is non-zero: we might have to wait for ever to find that out!

2. Some very easy 'reality checks':
   (a) Which of the following definitions characterize (total) computable functions $f\colon \mathbb{N} \to \mathbb{N}$?
      i. $f(n) = 3^n$.
      ii. $f(n) = n^3 + 6n^2 + 7$.
      iii. $f(n) = \sqrt[3]{n}$.
      iv. $f(n) = 0$ if n is prime, and $f(n) = 1$ otherwise.
      v. $f(n)$ is the $n+1$-th prime number.
   (b) Suppose $f\colon \mathbb{N} \to \mathbb{N}$ and $g\colon \mathbb{N} \to \mathbb{N}$ are computable functions.
      i. Is their product $h(n) =_{\text{def}} f(n) \cdot g(n)$ computable?
      ii. Is their composition $f \circ g$ computable?
   (c) Suppose Δ and Γ are decidable subsets of $\mathbb{N}$. Show that so too are
      i. $\Delta \cup \Gamma$,
      ii. $\Delta \cap \Gamma$,
      iii. $\mathbb{N} \setminus \Delta$.

(a) i. Evidently computable.

ii. Evidently computable – as indeed is the value of any polynomial of the form

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_2 n^2 + a_1 n^1 + a_0$$

(where each $a_j \geq 0$).

iii. Not a total function $f\colon \mathbb{N} \to \mathbb{N}$, so a fortiori not a computable total function.

iv. Evidently computable, since there is an algorithm for deciding whether a number is prime.

v. We know that there is an infinity of primes, so the function is total, defined for every n. To compute $f(n)$, start off on the process of taking each natural number in term and testing to see if it is a prime, and recording successes: output the $n+1$-th success.

(b) 'Yes' to both (i) and (ii). In particular, we compute the value of $f \circ g$ for argument n by first computing the value of $g(n)$, and then feeding that result in as the argument to the computable function f, to get the value $f(g(n))$.

(c) Δ is a decidable subset of $\mathbb{N}$ iff the characteristic function of Δ – i.e. the function c_Δ such that $c_\Delta(n) = 0$ if $n \in \Delta$ and $c_\Delta(n) = 1$ otherwise – is computable. So we are given that the characteristic functions c_Δ and c_Γ are computable. Then

i. Their product $c_\Delta \cdot c_\Gamma$ is computable. But this is the function which is zero for argument n so long as one of $c_\Delta(n)$ and $c_\Gamma(n)$ is 0 (and is 1 otherwise). And this is $c_{\Delta\cup\Gamma}$ (think about it!). So, since its characteristic function is computable, $\Delta \cup \Gamma$ is decidable.

ii. $c_{\Delta\cap\Gamma}(n)$ is 0 if both $c_\Delta(n)$ and $c_\Gamma(n)$ is 0 and is 1 otherwise. So its value is evidently computable if $c_\Delta(n)$ and $c_\Gamma(n)$ are. If you want an equation, we can put $c_{\Delta\cap\Gamma}(n) =_{\text{def}} 1 - \{(1 - c_\Delta(n))(1 - c_\Gamma(n))\}$.

iii. The characteristic function of $\mathbb{N} \setminus \Delta$ is simply $1 - c_\Delta$, so is computable if c_Δ is.

3. (a) Define $j \colon \mathbb{N} \to \mathbb{N}$ as follows: $j(n) = n + 1$ if Julius Caesar ate grapes on his fifth birthday, and $j(n) = 2n$ otherwise. Is j an effectively computable function? [Hint: look carefully at the summary definition of an effectively computable function.]

(b) The function $k \colon \mathbb{N} \to \mathbb{N}$ is defined as follows: $k(n) = 0$ if there are at least n consecutive 7s somewhere in the decimal expansion of π, and $k(n) = 1$ otherwise. Is k effectively computable? [Hint: we don't know whether there is a maximum number m such that there are at least m consecutive 7s somewhere in the decimal expansion of π, so consider the cases where there is and there isn't separately.]

(c) The function $h \colon \mathbb{N} \to \mathbb{N}$ is defined as follows: $h(n) = 0$ if there are *exactly* n consecutive 7s [bounded by some other digits] somewhere in the decimal expansion of π, and $h(n) = 1$ otherwise. Is h effectively computable?

(a) We said a one-place total function $f \colon \mathbb{N} \to \mathbb{N}$ is effectively computable iff there is an algorithm which can be used to calculate, in a finite number of steps, the value of the function for any given natural number input. NB: we only require that there *is* an algorithm for f, not that we know which algorithm computes f. Now, depending on Julius Caesar's grape consumption, j is either the successor function, and there is a trivial algorithmic procedure Π_S to compute that, or it is the double-it function, and there is a trivial algorithmic procedure Π_2 to compute *that*. So either way, the function is computable. (But we just don't know which of the two algorithms to use to compute j.)

(b) Since we don't know whether there a maximum number m such that there are least m consecutive 7s somewhere in the decimal expansion of π, we have to consider cases. If there *isn't* a maximum such number, then $k(n) = 1$ for all n. And that constant function is trivially computable. If there *is* a maximum m, then for all $n \leq m$, $k(n) = 0$, and for $n > m$, $k(n) = 1$. And that step function is trivially computable too (to determine $k(n)$ which just compare n with m). So either way k is computable by some algorithm: we again just don't know which algorithm to use.

(c) For all we know, as n increases $h(n)$ could eventually flip to and fro between 0 and 1 in a hopelessly unpatterned way. We just don't know whether h is computable. It is believed that the pattern of digits in the expansion of π is random, and then we can expect blocks of 7's of arbitrary length, so we always have $h(n) = 1$. But we don't *know* if this is so.

4. (a) Define the function $f\colon \mathbb{N}^2 \to \mathbb{N}$ as follows: f 'codes' the ordered pair of numbers $\langle m, n\rangle$ by mapping it to $2^m(2n+1)$.
   i. Prove f is an effectively computable bijection.
   ii. Show that the functions $fst(k)$ and $snd(k)$ are effectively computable, where these are the functions which 'decode' a number k by finding respectively the m and the n such that f maps $\langle m, n\rangle$ to k.
   iii. Prove f^{-1} is an effectively computable bijection.
   iv. Conclude that $\mathbb{N}^2$ is effectively enumerable.

   i. It is trivial that f is effectively computable. We need to show that f is (1) one-one and (2) onto (is injective and surjective).

   For (1), note that if $\langle m, n\rangle \neq \langle m', n\rangle'$, then either $m \neq m'$ or $n \neq n'$ or both. But if $m \neq m'$, then $2^m(2n+1) \neq 2^{m'}(2n'+1)$ (as they are divisible by 2 a different number of times) and if $m \neq m'$, then $2^m(2n+1) \neq 2^{m'}(2n'+1)$ (as they have different odd divisors).

   For (2), note that any number is equal to $2^m(2n+1)$ for some m, n (divide by two as often as possible – counting how often we can do it to get m – and that leaves us with an odd number $r = 2n+1$).

   ii. $fst(k)$ is computed by counting how many times k is divisible by 2. $snd(k)$ is computed by dividing k by 2 as many times as possible until left with an odd number r, then computing the n such that $r = 2n+1$.

   iii. Since f is a bijection, we know it has an inverse which is a bijection (see the exercise sheet 'Functions', Qn. 2). $f^{-1}(k) = \langle fst(k), snd(k)\rangle$, so f^{-1} can be evaluated by a computer.

   iv. f^{-1} effectively enumerates $\mathbb{N}^2$.

   (b) Suppose g maps the pair $\langle m, n\rangle$ to the code number $\{(m+n)^2 + 3m + n\}/2$.
   i. Show g is an effectively computable bijection from $\mathbb{N}^2$ to $\mathbb{N}$. [Hint, put $j = m + n$, and express the function g in terms of j and m.]
   ii. Show informally that the functions $fst(k)$ and $snd(k)$ are effectively computable, where these are the functions which 'decode' a number k by finding respectively the m and the n such that g maps $\langle m, n\rangle$ to k.
   iii. Show that that g's inverse is the effectively computable function which maps k to the k-th pair counting from 0 along the zig-zag route graphically displayed in *IGT2*, §2.4.
   iv. Conclude again that $\mathbb{N}^2$ is effectively enumerable.

   i. Put $j = m + n$. Then g maps $\langle m, n\rangle$ to $\frac{j(j+1)}{2} + m$, i.e. to $0 + 1 + \ldots + j + m$, where $0 \leq m \leq j$.

   But put the latter way, it is clear that g is a bijection. For every number can be expressed in the form $0 + 1 + \ldots + j + m$, where $0 \leq m \leq j$ in exactly one way (think about it!).

ii. Given a number k, it is a computational matter to put k into the form $\frac{j(j+1)}{2}+m$, where $0 \leq m \leq j$, and $fst(k) = m$, and $snd(k) = j - m$.

iii. Look at the north-east to south-west diagonals. Observe that for successive values of $j \geq 0$, each diagonal consists of $j+1$ pairs $\langle m, n \rangle$ where $m + n = j$. So when we get to the k-th pair counting from zero, where $k = \frac{j(j+1)}{2} + m$ with $m \leq j$, we'll be running down the diagonal where the pairs sum to j, at position m along the diagonal (counting them from 0), i.e. we encounter the pair $g^{-1}(k)$.

iv. $g^{-1}\colon \mathbb{N} \to \mathbb{N}^2$ serves as another enumerating map.

5. (a) Show that $\Sigma \subseteq \mathbb{N}$ is a decidable infinite set of numbers iff the members of Σ can be effectively enumerated in ascending order of size.

   (b) Show that a non-empty set $\Sigma \subseteq \mathbb{N}$ is decidable iff Σ is the range of a non-decreasing computable function $f\colon \mathbb{N} \to \mathbb{N}$. [A function is non-decreasing iff $m < n \to f(m) \leq f(n)$.]

   (c) Can your proof of (b) be adapted to show that any effectively enumerable set of numbers is decidable?

(a) Suppose Σ is decidable and infinite. Evaluate $f(n)$ by testing whether $0 \in \Sigma$, $1 \in \Sigma$, $2 \in \Sigma, \ldots$ in turn and outputting the $n{+}1$-th success. Then f is effectively computable (since by assumption membership of Σ is decidable), increasing, and enumerates Σ.

For the other direction, suppose f effectively enumerates Σ in increasing order. Then to decide whether $n \in \Sigma$, compute $f(0)$, just compute the values of $f(0)$, $f(1)$, $f(2), \ldots$ in turn. Then either for some k, $f(k) = n$ (so $n \in \Sigma$). Or else – because f by hypothesis enumerates an infinite set – we eventually get to some k where $f(k) > n$ and for no $j < k$, $f(j) = n$. And then $n \notin \Sigma$ (because f never decreases, if n were in Σ, then n *would* be the value of some $f(j)$ for an earlier $j < k$). That effectively decides whether $n \in \Sigma$.

(b) Suppose Σ is infinite. Then we can recycle the argument we've just used to show that it is decidable iff effectively enumerable by a non-decreasing function.

But if Σ is finite and non-empty it is (i) trivially decidable, and (ii) trivially the range of a non-decreasing computable function (take the enumerating function to be the function that lists the finite number of members in Σ in increasing order, repeating the final, maximum, member for ever: that's computable). So that establishes the (material) biconditional: Σ is decidable iff the range of a non-decreasing computable function.

So either way, Σ is decidable iff it is the range of a non-decreasing computable function.

(c) We know from Theorem 3.9 that some effectively enumerable sets of numbers are not decidable. In other words, some sets of numbers Σ which are the range of a computable function are not decidable. So we know that the 'non-decreasing' clause in the statement of the previous result has to be crucial. And it is. Look at the pivotal role in our earlier proof of the claim "And then $n \notin \Sigma$ (because f never decreases, if n were in Σ, then n *would* be the value of some $f(j)$ for an earlier $j < k$)." If f might decrease, this step fails.