MATH 3TP3 Assignment #10 Solutions

In this question, we show that the set of TNT-sentences proven by TNT is, although computably enumerable, not computable. There is, in other words, no decision procedure for TNT-theoremhood.

This is a consequence of the fact that TNT proves all true sentences of the form $\text{Theorem}_S(\overline{\ulcorner n \urcorner})$. Actually, although I didn't mention this in lectures so as to keep the presentation of G2T clean, TNT' already proves such sentences. So we're really showing that no system which strengthens TNT' and is sound for $\mathbb{N}$ has computable theory. But I'll stick to talking about TNT in these solutions.

In outline, the idea is this: the halting problem yields a set H which is computably enumerable but not computable (namely: the set of codes for programs which halt when fed their own code as input). Now TNT "implements" arbitrary Post formal systems, via $\text{Theorem}_S(x)$, and Post formal systems are Turing complete; we deduce that $n \in H$ is true iff $\text{Theorem}_S(\overline{\ulcorner f(n) \urcorner})$ is a TNT-theorem, where S is an appropriate Post system and f is a computable translation map (which below will be Dashify). Hence if the set of theorems were computable, so would be H.

The question led you through this argument, asking you to fill in the details. So here I'll reproduce the text of the question, and insert answers in italics to the bold bits where you were asked to do something.

Suppose we have fixed a Gödel numbering for an alphabet which contains the alphabet of TNT and also the dash symbol "-".

Define the function Dashify : $\mathbb{N} \to \mathbb{N}$ by

$$\text{Dashify}(n) = \ulcorner \text{-}^n \urcorner,$$

where "-^n" is the string consisting of n dashes.

Show that Dashify **is a (total) computable function, by describing informally an algorithm to calculate it.**

To calculate $\text{Dashify}(n)$*, write* $\ulcorner \text{-} \urcorner$ *in decimal* n *times, concatenated together into a single decimal string, then read the result as a decimal number. In more detail (paralleling the definition of* Concat*):* $\text{Dashify}(0) = 0$*, and to calculate* $\text{Dashify}(n+1)$ *first calculate* $\text{Dashify}(n)$ *then add it to* $\ulcorner \text{-} \urcorner \cdot 10^l$*, where* l *is the length of the decimal representation of* $\ulcorner \text{-} \urcorner$*.*

Since string manipulations and finding free variables correspond to computable operations on Gödel numbers, it follows that for any TNT-wff with one free variable $\phi(x)$, the function $\text{DashSub}_\phi : \mathbb{N} \to \mathbb{N}$ defined by

$$\text{DashSub}_\phi(n) = \ulcorner \phi(\overline{\ulcorner \texttt{-}^n \urcorner}) \urcorner$$

is computable. You should ponder this, but you don't need to write anything.

I ponder by writing. Simplest way to explain how to do this: first expand ϕ into its complete parse tree, then replace each free instances of x (i.e. those which aren't below any $\exists x :$ or $\forall x :$) with $\texttt{-}^n$, then push the parse tree back into a single string (it folds up smoothly, like closing a pop-up book), and take its Gödel number. Parsing, string substitution and unparsing are all simple algorithmic operations.

Recall that we showed, as part of our discussion of the Halting problem, that there exists a set H which is c.e. but not computable.

Consider Dashify(H), the image of H under the dashification map Dashify. **Show that Dashify(H) is c.e. but not computable.**

Dashify(H) *is c.e.: given n, calculate* Dashify(0), Dashify(1),... *in order, testing for equality with n; if we find that some* Dashify$(m) = n$, *then run our semi-decision procedure for H with input m. If it finds that $m \in H$, then $n \in$* Dashify(H) *and we return True. Otherwise, $n \notin$* Dashify(H), *and we never return anything.*

Dashify(H) *is not computable: suppose it were; then H would also be computable, contradicting the choice of H. Indeed: given n, calculate* Dashify(n) *and run our hypothesised decision procedure to determine whether* Dashify$(n) \in$ Dashify(H). *Since* Dashify *is injective, $n \in H$ iff* Dashify$(n) \in$ Dashify(H).

Deduce that there is a Post formal system S for which $\texttt{-}^n$ is an S-theorem iff $n \in H$.

Post's theorem on Turing completeness of Post formal systems we stated in lectures tells us that if a set Σ of Gödel numbers of strings in an alphabet A is c.e. then there is a system S, in an alphabet $A' \supseteq A$ which may extend A by adding some auxiliary symbols, such that Σ is precisely the set of A-strings which are S-theorems. Applying this to the c.e. set Dashify(H) *of Gödel numbers of strings in the alphabet $A = \{\texttt{-}\}$ whose only symbol is $\texttt{-}$, we obtain a system S which is as required.*

(Note that it's quite clear that we'll need auxiliary symbols in a case such as this - systems in one-symbol alphabets can't do very much at all)

Without loss of generality, assume that we can extend our Gödel numbering to include the alphabet of S.

Now

$$n \in H \iff \mathbb{N} \vdash \text{Theorem}_S(\overline{\ulcorner \text{-}^n \urcorner}); \tag{1}$$

as we saw (/will see) in our discussion of Gödel's Second Incompletness Theorem, it follows:

$$n \in H \iff \text{TNT} \vdash \text{Theorem}_S(\overline{\ulcorner \text{-}^n \urcorner}). \tag{2}$$

(By the way: note that we're using $\mathbb{N}$*-soundness of* TNT *to get the right-to-left direction. If* TNT *is actually inconsistent, for example, then it proves everything and so in particular proves* $\text{Theorem}_S(\overline{\ulcorner \text{-}^n \urcorner})$ *for every n!)*

By considering DashSub_ϕ for an appropriate ϕ, **show that**

$$\{\ulcorner \text{Theorem}_S(\overline{\ulcorner \text{-}^n \urcorner})\urcorner \mid n \notin H\}$$

is *not* c.e.

First, note that H's complement $\mathbb{N} \setminus H$ *is not c.e., since H is c.e. but not computable, and a set is computable iff both it and its complement are c.e.. Now let* $\phi(x)$ *be* $\text{Theorem}_S(x)$*. Then* $\text{DashSub}_\phi(n) = \ulcorner \text{Theorem}_S(\overline{\ulcorner \text{-}^n \urcorner})\urcorner$*. So we want to see that* $\{\text{DashSub}_\phi(n) \mid n \notin H\} = \text{DashSub}_\phi(\mathbb{N} \setminus H)$ *is not c.e.. But this follows, just as in the case of* $\text{Dashify}(H)$ *above, from computability and injectivity of* DashSub_ϕ *and the fact that* $\mathbb{N} \setminus H$ *is not c.e..*

Deduce from this and (2) that the set of Gödel numbers of TNT-sentences which are not TNT-theorems is not c.e.

Suppose it were. Then we can semidecide $\text{DashSub}_\phi(\mathbb{N} \setminus H)$ *as follows: given m, first run through n to see if* $m = \text{DashSub}_\phi(n)$ *for some n. If it is, then run our hypothesised semidecision procedure to check if m is the Gödel number of a non-theorem, returning True if it does.*

Conclude that there does not exist an "anti-TNT", a formal system which proves precisely those TNT-sentences which TNT does not prove.

Being a TNT-sentence is a decidable property of a string (we just have to check well-formedness and that there are no free variables). So the set of TNT-sentences which are theorems of a formal system is a c.e. set.