MATH 3TP3 Assignment #9 Solutions

1. I'll present this slightly abstractly, with the aim of making it clear that this same trick will work for any function defined in this simple recursive way.

   Consider the following recursive mathematical definition of a function gödelNumeral(x):

   - gödelNumeral$(0) := \ulcorner 0 \urcorner$
   - gödelNumeral$(n+1) :=$ concat$(\ulcorner S \urcorner,$ gödelNumeral$(n))$

   where for strings α and β, concat$(\ulcorner \alpha \urcorner, \ulcorner \beta \urcorner) = \ulcorner \alpha\beta \urcorner$ (this is the function expressed by our formula Concat(x, y, z)). (This is "recursive" in the sense that its value at $n+1$ is defined in terms of its value at n.)

   Then for any natural number n, gödelNumeral$(n) = \ulcorner \overline{n} \urcorner$.

   Now we can implement gödelNumeral in a formula in just the same way as we did for Exp: by using a list to hold the intermediate values. So let GödelNumeral(x, y) be the formula:

   $$\begin{aligned}&\exists z : \langle [z]_0 = \overline{\ulcorner 0 \urcorner} \wedge \langle [z]_x = y \wedge \\ &\quad \forall x' : \langle x' < x \supset \exists y' : \exists y'' : \langle \langle [z]_{x'} = y' \wedge [z]_{Sx'} = y'' \rangle \wedge \\ &\qquad \mathrm{Concat}(\overline{\ulcorner S \urcorner}, y', y'') \rangle \rangle \rangle \rangle .\end{aligned}$$

   To spell out what this means: for $n, m \in \mathbb{N}$, we have $\mathbb{N} \models$ GödelNumeral$(\overline{n}, \overline{m})$ iff there is a list whose first n elements are

   $$(\ulcorner 0 \urcorner, \mathrm{concat}(\ulcorner S \urcorner, \ulcorner 0 \urcorner), \mathrm{concat}(\ulcorner S \urcorner, \mathrm{concat}(\ulcorner S \urcorner, \ulcorner 0 \urcorner)), \ldots),$$

   i.e. the first n elements are

   $$(\text{gödelNumeral}(0), \text{gödelNumeral}(1), \ldots, \text{gödelNumeral}(n)),$$

   and the nth element (numbering from 0) is m. In other words, $m =$ gödelNumeral(n).

2. One rather important thing to stress before I give a solution: a distressing number of you seemed to think that TNT $\vdash \sim \sigma$ is the same as TNT $\nvdash \sigma$. They are not at all the same!

The first implies the second, since TNT is sound for $\mathbb{N}$ so can't prove both σ and $\sim \sigma$, but the second does not imply the first. Indeed, consider the Gödel sentence G_{TNT}. We saw that TNT $\nvdash$ G_{TNT}, because that's what G_{TNT} "says" and (as we showed) G_{TNT} is true (in $\mathbb{N}$). But precisely because G_{TNT} is true, we also have TNT $\nvdash \sim \sigma$, because TNT is sound for $\mathbb{N}$!

Now on to the question.

The idea here was to perform a minor variation of the arithmoquining trick used to create the Gödel sentence given in lectures. I'll go through it in some wordy detail here. "True" will mean "true in $\mathbb{N}$".

Let's find a sentence in English first, then translate to TNT.

We want a sentence which says of itself that it's negation is a TNT-theorem. To get a sentence which talks about itself, we quine a formula which talks about the quine of its free variable. So let p be the "incomplete English sentence":

> The negation of the quine of x is a TNT-theorem.

Then the quine of p is the sentence s:

> The negation of the quine of "The negation of the quine of x is a TNT-theorem" is a TNT-theorem.

So s can be read as saying that "not s" is a TNT-theorem.

Now let's translate this to TNT. Quining becomes arithmoquining, which we know we can express with the formula Arithmoquine(x, y). Saying that a sentence is a TNT-theorem is also something we know how to express - that's Theorem$_{\text{TNT}}(x)$. How about saying that the **negation** of a sentence is a TNT-theorem? Well, all that's missing is a way to express that a sentence is the negation of another sentence - but that's just a matter of having an extra '$\sim$' symbol at the start, which is a simple syntactic feature that Concat can detect: we can read Concat$(\overline{\ulcorner\sim\urcorner}, y, z)$ as saying that z codes for the negation of what y codes for (when y does code for a formula).

So we can translate our English string p to the TNT-formula $\phi(x)$:

$$\exists y : \langle \text{Arithmoquine}(x, y) \wedge \exists z : \langle \text{Concat}(\overline{\ulcorner\sim\urcorner}, y, z) \wedge \text{Theorem}_{\text{TNT}}(z)\rangle\rangle.$$

Now let σ be the arithmoquine of $\phi(x)$,

$$\exists x : \langle x = \overline{\ulcorner \phi(x) \urcorner} \wedge \phi(x) \rangle.$$

Then σ is true iff $\phi(\overline{\ulcorner \phi(x) \urcorner})$ is true, which is iff the negation of the arithmoquine of $\phi(x)$ is a TNT-theorem, which is iff the negation of σ is a TNT-theorem, i.e. iff TNT $\vdash \sim \sigma$.

So σ is as desired.

Is σ true? Well, suppose it is. Then TNT $\vdash \sim \sigma$. But TNT is sound for $\mathbb{N}$, so $\mathbb{N} \models \sim \sigma$, i.e. σ is false. Contradiction.

So σ is false.

Does this give us $\mathbb{N}$-incompleteness of TNT? Well... σ is false, so $TNT \not\vdash \sim \sigma$. But $\sim \sigma$ is true. So $\sim \sigma$ is a true sentence which is not a theorem. So indeed, this shows that TNT is $\mathbb{N}$-incomplete.

The arguments we've gone through here are essentially identical to those in lectures. σ is basically $\sim G_{\text{TNT}}$. I suggest you now reread this section of lectures, and make sure you understand what G_S is why it shows that S can't be both sound and complete for $\mathbb{N}$.