24.118: Paradox and Infinity
PS 5: Suggested solutions

November 19, 2001

[Note (In case this wasn't obvious): If a set is effectively enumerable, there are many ways of effectively listing it. So the procedures I describe are each just one of many possible ways of showing that the set in question is effectively enumerable.]

1a.

(3pts)

TRUE. Suppose S and S' are effectively enumerable, algorithm A1 enumerates S and algorithm A2 enumerates S'. To enumerate $S \cup S'$ do this: Run A1 for one step. If it outputs an element, list it. Run A2 for one step . If it outputs an element, list it. Run A1 for another step. If it outputs an element, list it. Run A2 for another step. If it outputs an element, list it. Run A1 for another step ...

Some of you said we could list the sets 'in succession'. However, that may not work. Remember that in order to effectively enumerate a set you have to make sure that each member of the set gets listed after a finite number of steps. If you try listing the union of two sets by listing the two sets in succession and the first set you try listing is infinite, then none of the elements of the second set ever get listed (except for those in the intersection of te two sets). Because of this difficulty, just saying that we can use the two algorithms associated with the two sets to list the union of the sets is not enough. You need to spell out, however briefly, how they are to be used together.

1b.

(3pts)

TRUE. Suppose S and S' are effectively enumerable, algorithm A1 enumerates S and algorithm A2 enumerates S'. To enumerate $S \cap S'$ on List L do this: List the 1st elements of S and S' using A1 and A2. List the 2nd element of S. If this is equal to the 1st element listed by A1, place it in L. List the 2nd element of S. If it is equal to the 1st or second element listed by A1, place it in L. . . .
In general: List the nth element of S. If it is the same as one of the 1st to n-1st element listed by A2, place it in L. List the nth element of S'. If it is the same as one of the 1st to nth element listed by A1, place it in L.

1c.

(3pts)

FALSE. Counterexample: The set of numbers that code theorems of PA is effectively enumerable, but its complement is not effectively enumerable (because it it were, then PA would be decidable, which we know it isn't).

Some of you proposed something like the following: Suppose you want to list the complement of S. Take a list L of all the natural numbers. List S. For each element e of S listed, erase e from L. At the end of that process, you have a list of the complement of S. This doesn't work. What you need to show is that there is an effective enumeration of the complement S' of S, ie a listing procedure such that each element of S' gets listed after finitely many steps. Suppose, using your procedure, you want to list the complement of S. Which element do you list first? Only after listing *all of* S, will you know, using your procedure, what elements of the original list L are in S'. But that means you will never know what elements to list for S'.

1d.

(4pts)

FALSE. If this were true, then *every* set of natural numbers would be effectively enumerable. This is how: Without loss of generality, assume S is an infinite set. Now consider the collection C=$\{\{n\}|n \in S\}$, that is the collection of singleton sets each of which contains exactly one member of S. C contains infinitely many effectively enumerable sets (since S is infinite and each set in C is finite). If 1d were true, then the union of the sets in C would be effectively enumerable and since the union of the sets in C is just S, S would be effectively enumerable. But we know that there are many sets that are not effectively enumerable.

So what's wrong with the procedure that many of you proposed to enumerate S, viz: List the 1st element of the 1st set, the 1st element of the 2nd set, the 2nd element of the 1st set, the 2nd element of the 2nd set, the 1st element of the 3rd set, etc.?
The problem is that in order to list the elements of the nth set of the sequence, you need to know what the nth algorithm is. And to know this, you need to assume that the set of listing algorithms is itself effectively enumerable.[1]

1e.

(4pts)

TRUE. Suppose A and B are effectively enumerable, enumerated by algorithms AA and AB, respectively. We need to find algorithms AC and AD listing sets C and D (with the desired

[1]If you want to effectively list the union of *finitely* many effectively enumerable sets, you can just use all the the finitely many listing algorithms as subroutines in your algorithm (which will still be finite). But if you want to effectively list the union of *infinitely* many effectively enumerable sets, you have to find some other way of invoking the infinitely many listing algorithms. Now, an algorithm can just be characterized by a finite number of symbols (say a C-program that implements it), which in turn can be coded as a single number using some Gödel-coding. So an algorithm can efffectively be accessed through its code-number under some fixed coding-scheme. Thus, a collection of infinitely many algorithms can be coded as a set of numbers (which code algorithms). If this set is effectively enumerable, you can effectively list all the members of all the sets.

properties). The idea is this: Put all elements which are in A but not in B into C, put all elements in B but not in A into D and 'split up' the overlap (elements in both A and B) in some systematic, effective fashion.
One way to do this is the following: Start with the C-list and D-list empty. At step n, run AA for another step. If there is an output o, check if o is already on the D-list. If not, put o on the C-list. Then run AB for another step. If there is an output o, check if o is already on the C-list. If not, place o on the D-list.

Some of you drew some arbitrary line between the overlap and said that since the resulting sets C and D are subsets of effectively enumerable sets (A and B respectively), C and D are themselves effectively enumerable. But that doesn't follow. There are many effectively enumerable sets with subsets that are not effectively enumerable. In fact, most subsets of the set of natural numbers are not effectively enumerable as discussed in class.

1f.

(4pts)

FALSE. The reason p is calculable is that for every number n we can determine whether n codes an arithmetical sentence φ, and if it does, inspect whether there is a PA-proof of either φ or $\neg\varphi$. (a) If we find a proof of φ we output 1, (b) if we find a proof of $\neg\varphi$. (c) If we find no proof for either, we keep searching for ever and never output anything.
It is easy to extend p to those numbers which do not code PA-sentences at all (because we can determine in finitely many steps whether a number codes a PA-sentence or not), but in order to extend p to a *total* function we need to have a mechanical way of finding out whether or not the proof-search-procedure described above every yields a result (the problem being that if we are in situation (c), we will never know this). But there is no mechanical way for doing that, so p cannot be extended to a total calculable function.

1g.

(4pts)

TRUE. Suppose algorithm A enumerates R. Define f as follows

$f(n) = m$ such that $\langle n, m\rangle$ is the first pair listed by A whose first member is n.

Then f is calculable (to calculate $f(n)$ it just list R and wait for the first pair with n as first member, then extract the second member and output it), total (since for every n there is an m with $R(n,m)$) and for every n: $R(n, f(n))$

1h.

(4pts)

TRUE. If the set S of Gödel-numbers of PA-decidable sentences was decidable, then PA itself would be decidable. This is how you would PA: Suppose A is a decision-procedure for S. Then to determine if PA proves ϕ run A on k, the Gödel-number of ϕ. If A returns 0, then ϕ is not PA-theorem. If A returns 1, then search for a proof (eg check the natural numbers

one by one for whether they code a PA-proof of either ϕ or $\neg\phi$. We know there is a proof of one of them. If you find a proof of ϕ, ϕ is a PA-theorem, if you find a proof of $\neg\phi$, ϕ is not PA-theorem).
On the other hand, S is clearly effectively enumerable: Check the natural numbers one by one for whether they code PA-proofs. If the number you check codes a proof, list both the last line of the proof (it is provable in PA) and the negation of the last line of the proof (it is refutable in PA).

1i.

(4pts)

TRUE. $\mathrm{Bew}_{PA}([[2]+[2]=[4]])$ is provable in PA. This is because there is a PA-proof of $[[2]+[2]=[4]]$. Since there is a proof, we can effectively find it.

Some of you said that since '2+2=4' is true, PA can prove it, so $\mathrm{Bew}_{PA}([[2]+[2]=[4]])$ is provable in PA. That argument doesn't work because there are true sentences that PA cannot prove (eg the Gödel-sentence).

1j.

(4pts)

FALSE. $\neg\mathrm{Bew}_{PA}([[2]+[2]=[5]])$ is a *consistency-statement* for PA and since PA cannot prove its own consistency (by the 2nd incompleteness theorem), PA cannot prove $\neg\mathrm{Bew}_{PA}([[2]+[2]=[5]])$. Also, PA cannot prove $\mathrm{Bew}_{PA}([[2]+[2]=[5]])$, because there is no proof of $[[2]+[2]=[5]]$ and so $\mathrm{Bew}_{PA}([[2]+[2]=[5]])$ would be false (and PA proves no false statements).
Recall that a theory which is inconsistent can prove *every* statement. So a statement that says that there is some statement the theory cannot prove amounts to the claim that the theory is consistent.

2.

(6pts)

Claim If Goldbach's conjecture (GC) is false, then PA can prove that GC is false.

Thus, if GC is undecidable (i.e. PA can neither prove nor refute GC), GC is true.

To see that the claim is true, consider the negation of GC:

$$\exists n > 2 \exists p < n \exists p' < n (n \text{ is even } \wedge (p \text{ is prime } \wedge p' \text{ is prime}) \wedge n = p + p')$$

(that is: there is an even number $n > 2$ such that it is the sum of two primes (both primes less than n))
Suppose n is such a number. Then PA proves

$$\exists p < n \exists p' < n (n \text{ is even } \wedge (p \text{ is prime } \wedge p' \text{ is prime}) \wedge n = p + p')$$

This is because there is an effective way of finding n.