Show that a language is decidable iff some enumerator enumerates the language in lexicographic order

The proof is given in the below:

> If A is decidable, the enumerator operates by generating the strings in lexicographic order and testing each in turn for membership in A using the decider. Those strings which are found to be in A are printed.
>
> If A is enumerable in lexicographic order, we consider two cases. If A is finite, it is decidable because all finite languages are decidable. If A is infinite, a decider for A operates as follows. On receiving input w, the decider enumerates all strings in A in order until some string lexicographically after w appears. That must occur eventually because A is infinite. If w has appeared in the enumeration already then *accept*, but if it hasn't appeared yet, it never will, so *reject*.
>
> Note: Breaking into two cases is necessary to handle the possibility that the enumerator may loop without producing additional output when it is enumerating a finite language. As a result, we end up showing that the language is decidable but we do not (and cannot) algorithmically construct the decider for the language from the enumerator for the language. This subtle point is the reason for the star on the problem.

1. I could not understand the rejection case in the last line in the second paragraph. The author said "if it has not appeared yet, it never will". Why, could anyone explain it for me in a simpler way?
2. Also I could not understand the last paragraph at all, could anyone explain it to me in a simpler way?

computability undecidability

edited Mar 12 at 3:54

Yuval Filmus
178k 12 166 325

asked Mar 11 at 17:55

Idonotknow
64 7

1 Don't use images as main content of your post. This makes your question impossible to search and inaccessible to the visually impaired; we don't like that. Please transcribe text and mathematics (note that you can use LaTeX) and don't forget to give proper attribution to your sources! – D.W. ♦ Mar 11 at 19:24

1 What exactly couldn't you understand? Can you make your question more precise? I worry that if someone else explains you might just say you don't understand that either. Have you tried checking a different textbook/source to see what proof they give for this fact? Reading a different proof might help you understand whatever you don't understand about this one. – D.W. ♦ Mar 11 at 19:24

2 By the way, what is meant here by "lexicographic order"? Usually this order causes $a < aa < aaa < aaaa < \ldots < b < bb < \ldots$. If so, the first part looks quite wrong to me: an enumerator can't start by enumerating string in lexicographic order since it will never reach b in finite time. I guess here it actually means something like "order strings by length first, and then use lexicographic order" (aka shortlex order). – chi Mar 12 at 12:15

1 Answer

It's easier to think of A as a list of natural numbers. If A is decidable, then we can list all numbers in A in increasing order by just testing all of them in order – Is $0 \in A$? Is $1 \in A$? And so on.

Now let us show that if A can be enumerated in increasing order, then we can decide A. Let us try to use the following algorithm:

> On input n, enumerate all integers in A in increasing order until a number $m \geq n$ appears. If n has appeared, then clearly $n \in A$. If n has not appeared, then $n \notin A$.

To see why the last point holds, suppose that $n = 5$, and the enumeration starts $2, 3, 6$. At the point when we see 6, we know that 5 is never going to appear.

There is one problem with this algorithm – if A is a finite language, then a number $m \geq n$ might never appear. Indeed, suppose for example that $A = \{2, 3, 5\}$ and that $n = 6$. We see $2, 3, 5$, and then wait forever for a larger number to appear. We fix this problem in the following way:

- If A is finite, it is clearly decidable.
- If A is infinite, then the algorithm described above always terminates (why?).

Unfortunately, given just an enumerator for A, we (provably) cannot know whether A is infinite or not, and so we don't know which of the two cases to use. Fortunately, this does not matter for the proof. All we need to do is show that if A has an enumerator, then it has a decider. There is no requirement to turn an enumerator for A into a decider for A in a uniform fashion.

answered Mar 12 at 3:59

Yuval Filmus