Bigger and Bigger gets Harder and Harder

Let $P(f,x)$ be a nontrivial semantic property of programs f and inputs x, like the halting property ("Does f halt on input x?"), and let $p_n(f,x)$ be a program purporting to determine P, but only when the sizes of both f and x are no greater than n. Clearly such programs exist for solving any property P (semantic or not) and for every size-bound n, since there are only finitely many possible programs and inputs of that limited size and a table lookup would fit the bill.

By "nontrivial" we mean that there are f,x for which the property holds and others for which it does not. By "semantic" we mean that any two programs with the same input-output behavior either both have the property or neither does.

Since the property is nontrivial, there must be values Y, y, Z, and z, such that $P(Y,y)$ is FALSE while $P(Z,z)$ is TRUE. For the halting problem, Y is any program that diverges on y and Z is one that converges for z.

Consider, now, the program

$$c(x) \ := \begin{cases} Y(y) & \text{if } p_n(x,x) \\ Z(z) & \text{otherwise} \end{cases}$$

It it straightforward to check that $p_n(c,c)$ cannot give the correct answer:

- If $p_n(c,c)$ returns TRUE, then $c(c)$ behaves like $Y(y)$. Because $P(Y,y)$ is FALSE and is a semantic property, $P(c,c)$ must be FALSE, and so the answer given by $p_n(c,c)$ should also have been FALSE.
- If, on the other hand, $p_n(c,c)$ returns FALSE, then $c(c)$ behaves like $Z(z)$. It follows that both $P(Z,z)$ and $P(c,c)$ are TRUE and the answer should have been TRUE.
- Finally, $p_n(c,c)$ is patently wrong if it returns an invalid answer (neither TRUE nor FALSE) or diverges and gives no answer at all.

So, assuming that the program p_n works as advertized, it must be that the counterexample c is too big for it. Now the size of c is

$$|c| = |p_n| + k,$$

where k is the combined sizes of Y, y, Z, and z, plus a few (language dependent) characters for the conditional test and the call to p_n.

The moral of all this is that any correct program p_n must be almost as large as its maximum legal input (or even larger):

$$|p_n| = |c| - k > n - k.$$

It follows immediately that there can be no (finite) program that correctly decides any (nontrivial semantic) property P for *all* programs and *all* inputs. Moreover, there exists some constant k (that depends only on the property$P$ and on the specific programming language) such that *no* program p correctly decides P for all programs and inputs of size up to $|p| + k$.

Notes:

1. The only programming features that were used in the above argument are calls to functions and conditionals (and copying input).
2. How size is measured is not criticial.
3. The program P might execute or interpret its input f, but only on values it is certain that f terminates, or else it could not guarantee its own termination.
4. Most modern programming languages allow programs to be passed as parameters, as done here. If not, then one can pass instead some computable code or identification number for the program. By "computable", we mean that it is possible to decompose the program based on the code and look at its component parts and also interpret it and execute the referenced program on any given input.
5. Essentially the same argument applies to properties of programs with more than one argument (but not when there is no input). In that case, both programs and tuples of inputs can be enumerated and assigned numerical values.