Scooping the Loop Snooper

A proof that the Halting Problem is undecidable

Written by: Geoffrey K. Pullum
No perfect procedure for bug checks will do

No I won’t just assert it, I’ll prove it to you

I will prove that although you might work til you drop

You cannot tell if comp-u-ta-tion will stop
For imagine we have a procedure called P

That for specified input permits you to see

Whether specified source code, with all of its faults,

Defines a routine that e-vent-u-ally halts.
You feed in your program, with suitable data,

And P gets to work, and a little bit lata

In finite compute time correctly infers

Whether infinite looping behavior occurs.
If there is no looping, then P prints out Good
That means on this input it halts, as it should.

But if it detects an unstoppable loop,
then P reports Bad! and you’re *in the soup*.
Well, the truth is that P cannot possibly be,
Because if you wrote it and gave it to me,
I could use it to set up a logical bind
That would *shatter your reason* and scramble your mind.
Here’s the trick that I’ll use and it’s simple to do.

I’ll define a procedure, which I will call Q,

That will use P’s predictions of halting success

To stir up a terrible logical mess.
For a specified program, say A, one supplies,

The first step of this program called Q I devise

Is to find out from P what’s the right thing to say

Of the looping behavior of \( A \) run on \( A \).
If the answer is Bad Q will suddenly stop.

But otherwise, Q will go back to the top,
And start off again, *looping endlessly back*,
Till the universe dies…
and turns frozen and black.
And this program called Q
wouldn’t stay on the shelf;

I would ask it to forecast its run on itself.

When it reads its own source code,
just what will it do?

What’s the looping behavior of Q run on Q?
If P warns of loops then Q has to quit;  
Yet P is supposed to speak truly of it!

And if Q’s going to quit, then P should say Good!  
Which makes Q *start to loop*! – P denied that it would!
No matter how P might perform, Q will *scoop it*: Q uses P’s output to make P look stupid.

Whatever P says, it cannot predict Q: P is right when it’s wrong, and is false when it’s true!
I’ve created a paradox, neat as can be
And simply by using your putative P.

When you posited P you stepped into a snare;
Your assumption has led you right into my lair.
So where can this argument possibly go?
I don’t have to tell you;
I’m sure you must know.

We now know there cannot possibly be
A procedure that acts like the *mythical* $P$. 
So you can’t ever find a mechanical means
For predicting the acts of computing machines;
It’s something that cannot be done. So we users
Must find our own bugs. Our computers are losers!
Now I know that HALT can’t be done
This is an arg-ue-ment that you have won
But what of the problems that I care about
They can be solved fast,
of that there’s no doubt