

## WRONG AGAIN!

By Francis Sullivan



EVERYONE WHO'S EVER WRITTEN A PROGRAM MUST HAVE WONDERED AT SOME POINT HOW SO VERY MANY REALLY SNEAKY BUGS MANAGED TO CREEP INTO THE CODE. SOME ERRORS, OF COURSE, ARE MERELY TYPING MISTAKES—EVEN A MODERATE-SIZED PROGRAM CONSISTS OF SEVERAL THOUSAND

characters, each of which must be correct. But others are of a much different and more subtle character and can persist for years before being exposed.

I recall working on a code for a molecular dynamics simulation. Every few time steps of integration of the governing differential equations required a recalculation of each particle's local neighborhood because the particles moved as the simulation proceeded. Because this recalculation was so frequent, not too much was expected to change between updates. I tried to take advantage of this fact by using Edsger Dijkstra's elegant smoothsort algorithm, which runs in linear time on lists that are almost sorted. After the program we'd written had been in use for several years and served as the basis of a few publications, I noticed that I'd made an error by coding an "improved" smoothsort! The bug remained hidden because my implementation was correct unless the number of particles grew quite large. When I explained this to the users, I learned that they'd quietly replaced my version of smoothsort with a correct quicksort years earlier because they didn't understand how smoothsort was supposed to work.


Ever since this strange episode, I've wondered how and why such things can happen. Of course, if the pseudocode is well structured, it's possible to write a correct program without having *any* understanding of the algorithm. It reminds me of the song "Lobachevsky" by Tom Lehrer:

I am never forget the day I am given first original paper to write.  
It was on Analytic and Algebraic Topology of Locally Euclidean  
Metritzation of Infinitely Differentiable Riemannian Manifold.  
Bozhe moi! This I know from nothing. But I think of great  
Lobachevsky and I get idea—haha!

Lobachevsky's idea, of course, was to plagiarize. However, in-

stead of merely plagiarizing from smoothsort's pseudocode, I decided to do something more dangerous: I decided to think. The error was typical of what can happen when you "almost" understand an algorithm and then make a small change to express things differently. In other words, the mistake was in my interpretation of what I thought was smoothsort logic.

Philosophers have written extensively about logic errors, focusing on the question of how they could occur if everything is done according to logic. These authors usually mean "true" errors rather than something like a typing mistake, but it's difficult to tell if they assume that, for every proposition, either it or its negation could be proved. In the case of computer programs, Dijkstra and his colleagues strongly advocated a technology of formal verification, which can be useful in small cases because it provides clarity, discipline, and structure. If I'd done a formal analysis of my modification, I might have found the flaw. But then, of course, I'd have to somehow check the formal verification, perhaps by doing a higher-level even more formal verification of the first formal verification, and so on. It begins to feel like the halting problem.

So what is to be done? I suspect a completely reliable, formal, and rigorous self-checking method for expressing algorithms is impossible. This doesn't mean we shouldn't try for clarity of expression, but in the end, the acid test is to run the program on real data—lots and lots of real data. During the Punic wars, the Roman senator Cato said *Carthago delenda est!* (Carthage must be destroyed!) With apologies to all readers expert in Latin, I'm adopting as a motto, *data vera utenda sunt!* (Real data must be used!) 

Francis Sullivan is the director of the IDA Center for Computing Sciences in Bowie, Maryland. Contact him at [fran@super.org](mailto:fran@super.org).