PHYS.ORG

New mathematical framework formalizes oddball programming techniques

May 23 2012, by Larry Hardesty

Two years ago, Martin Rinard's group at MIT's Computer Science and Artificial Intelligence Laboratory proposed a surprisingly simple way to make some computer procedures more efficient: Just skip a bunch of steps. Although the researchers demonstrated several practical applications of the technique, dubbed loop perforation, they realized it would be a hard sell. "The main impediment to adoption of this technique," Imperial College London's Cristian Cadar commented at the time, "is that developers are reluctant to adopt a technique where they don't exactly understand what it does to the program."

Loop perforation is just one example of a way in which computer scientists are looking to trade a little bit of accuracy for substantial gains in performance. Others include high-speed chips that yield slightly inaccurate answers to arithmetic problems and low-power memory circuits that don't guarantee perfectly faithful storage. But all of these approaches provoke skepticism among engineers: If a computing system is intrinsically unreliable, how do we know it won't fail catastrophically?

At the Association for Computing Machinery's Conference on Programming Language Design and Implementation in June, Rinard's group will present a new mathematical framework that allows computer scientists to reason rigorously about sloppy computation. The framework can provide mathematical guarantees that if a computer program behaves as intended, so will a fast-but-inaccurate modification of it.

"Loop perforation shares with a lot of the research we've done this kind of happy-go-lucky, let's-give-it-a-go-and-see-what-happens approach," says Rinard, a professor in MIT's Department of Electrical Engineering and Computer Science. "But once you observe a phenomenon, it helps to understand why you see what you see and to put a formal framework around it."

Incentive structure

The new research, which also involved lead author Michael Carbin and his fellow graduate students Deokhwan Kim and Sasa Misailovic, fits into the general category of formal verification. Verification is a method for mathematically proving that a program does what it's supposed to. It's used in hardware design, in academic work on algorithms and in the development of critical software that can't tolerate bugs. But because it's labor intensive, it's rarely used in the development of commercial applications.

That's starting to change, however, as automated verification tools become more reliable and accessible. Carbin hopes that the performance gains promised by techniques such as loop perforation will give programmers an incentive to adopt formal verification techniques. "We're identifying all these opportunities where programmers can get much bigger benefits than they could before if they're willing to do a little verification," Carbin says. "If you have these large performance gains that just don't come about otherwise, then you can incentivize people to actually go about doing these things."

As its name might imply, loop perforation involves the computer routines known as loops, in which the same operation is performed over and over on a series of data; a perforated loop is one in which iterations of the operation are omitted at regular intervals. Like other techniques that trade accuracy for performance, Rinard explains, loop perforation works well with tasks that are, in the jargon of computer science, nondeterministic: They don't have to yield a single, inevitable answer. A few pixels in a frame of Internet video might be improperly decoded without the viewer's noticing; similarly, the precise order of the first three results of a Web search may not matter as much as returning the results in a timely fashion.

Drawing boundaries

With the researchers' framework, the programmer must specify "acceptability properties" for each procedure in a program. Those properties can be the types of things that formal verification typically ensures: that the result of a computation falls within a range of values, for instance, or that the output of a function adheres to a particular file format. But with the MIT framework, the programmer can also specify acceptability properties by reference to the normal execution of the program: The output of the modified procedure must be within 10 percent of the output of the unmodified procedure, say, or it must yield the same values, but not necessarily in the same order.

One advantage of the framework is that it allows developers who have already verified their programs to reuse much of their previous reasoning. In many cases, the programmer can define an acceptability property in such a way that, if it's met, the relaxed program will also preserve any other properties that the programmer has verified.

In the framework described in the new paper, the programmer must also describe how the procedure is to be modified, whether through loop perforation or some other technique. But Carbin says the group is already working on a computer system that allows the programmer to simply state acceptability properties; the system then automatically determines which modifications preserve those properties, with what types of performance gains.

"This idea of relaxation — trading the traditional notion that a computer has to do every part of a computation exactly correctly for huge energy savings, or performance savings, or ease of implementation — is not a new idea," says Dan Grossman, an associate professor of computer science and engineering at the University of Washington, who also works on program relaxation. "But what this paper does is put in on a much firmer footing."

The paper also, Grossman says, shows "how you can formally verify software. But what it's doing by doing that is explaining what relaxed software actually is, what it means, what exactly it's relaxing."

More information: Paper: “Proving Acceptability Properties of Relaxed Nondeterministic Approximate Programs” (PDF)

Provided by Massachusetts Institute of Technology

Citation: New mathematical framework formalizes oddball programming techniques (2012, May 23) retrieved 20 June 2024 from https://phys.org/news/2012-05-mathematical-framework-formalizes-oddball-techniques.html