Can control theory make software better?

March 19 2013, by Larry Hardesty

The oscillation of a pendulum offers the simplest example of a Lyapunov function, a central concept in control theory. The pendulum's loss of energy with each swing guarantees that it won't lurch into a less stable state.

"Formal verification" is a set of methods for mathematically proving that

a computer program does what it's supposed to do. It's universal in hardware design and in the development of critical control software that can't tolerate bugs; it's common in academic research; and it's beginning to make inroads in commercial software.

In the latest issue of the journal *IEEE Transactions on Automatic Control*, researchers from MIT's Laboratory for Information and Decision Systems (LIDS) and a colleague at Georgia Tech show how to apply principles from control theory—which analyzes dynamical systems ranging from robots to power grids—to formal verification. The result could help computer scientists expand their repertoire of formal-verification techniques, and it could be particularly useful in the area of approximate computation, in which designers of computer systems trade a little bit of computational accuracy for large gains in speed or power efficiency.

In particular, the researchers adapted something called a Lyapunov function, which is a mainstay of control theory. The graph of a standard Lyapunov function slopes everywhere toward its minimum value: It can be thought of as looking kind of like a bowl. If the function characterizes the dynamics of a physical system, and the minimum value represents a stable state of the system, then the curve of the graph guarantees that the system will move toward greater stability.

"The most basic example of a Lyapunov function is a pendulum swinging and its energy decaying," says Mardavij Roozbehani, a principal research scientist in LIDS and lead author on the new paper. "This decay of energy becomes a certificate of stability, or 'good behavior,' of the pendulum system."

Of course, most dynamical systems are more complex than pendulums, and finding Lyapunov functions that characterize them can be difficult. But there's a large literature on Lyapunov functions in control theory,

and Roozbehani and his colleagues are hopeful that much of it will prove applicable to software verification.

Skirting dangers

In their new paper, Roozbehani and his coauthors—MIT professor of electrical engineering Alexandre Megretski and Eric Feron, a professor of aerospace software engineering at Georgia Tech—envision a computer program as a set of rules for navigating a space defined by the variables in the program and the memory locations of the program instructions. Any state of the program—any values for the variables during execution of a particular instruction—constitutes a point in that space. Problems with a program's execution, such as dividing by zero or overloading the memory, can be thought of as regions in the space.

In this context, formal verification is a matter of demonstrating that the program will never steer its variables into any of these danger zones. To do that, the researchers introduce an analogue of Lyapunov functions that they call Lyapunov invariants. If the graph of a Lyapunov invariant is in some sense bowl shaped, then the task is to find a Lyapunov invariant such that the initial values of the program's variables lie in the basin of the bowl, and all of the danger zones lie farther up the bowl's walls. Veering toward the danger zones would then be analogous to a pendulum's suddenly swinging out farther than it did on its previous swing.

In practice, finding a Lyapunov invariant with the desired properties means systematically investigating different classes of functions. There's no general way to predict in advance what type of function it will be—or even that it exists. But Roozbehani imagines that, if his and his colleagues' approach catches on, researchers will begin to identify algorithms that lend themselves to particular types of Lyapunov invariants, as has happened with control problems and Lyapunov

functions.

Fuzzy thinking

Moreover, many of the critical software systems that require formal verification implement control systems designed using Lyapunov functions. "So there are intuitive reasons to believe that, at least for control-system software, these methods will work well," Roozbehani says.

Roozbehani is also enthusiastic about possible applications in approximate computation. As he explains, many control systems are based on mathematical models that can't capture all of the complexity of real dynamical systems. So control theorists have developed analytic methods that can account for model inaccuracies and provide guarantees of stability even in the presence of uncertainty. Those techniques, Roozbehani argues, could be perfectly suited for verifying code that exploits approximate computation.

"Computer scientists are not used to thinking about robustness of software," says George Pappas, chair of the Department of Electrical and Systems Engineering at the University of Pennsylvania. "This is the first work that is formalizing the notion of robustness for software. It's a paradigm shift, from the more exhaustive, combinatorial view of checking for bugs in software, to a view where you try to see how robust your software is to changes in the input or the internal state of computation and so on."

"The idea may not apply in all possible kinds of software," Pappas cautions. "But if you're thinking about software that implements, say, controller or sensor functionality, I think there's no question that these types of ideas will have a lot of impact."

More information: Paper: "Optimization of Lyapunov Invariants in Verification of Software Systems"

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology