Speak math, not code

March 2 2020

Eminent computer scientist Leslie Lamport, winner of 2013 Turing Award, speaking at the dialogue held in conjunction with the SMU-Global Young Scientists Summit 2020. Credit: Rebecca Tan

Have you ever followed a recipe to bake some bread? If you have, congratulations; you have executed an algorithm. The algorithms that follow us around the internet to suggest items we might like, and those

that control what shows up in our Facebook feeds may seem mysterious and uncanny at times. Yet, an algorithm is simply a set of instructions to be completed in a specified sequence, whether by human bakers or computer programs.

The difference, however, lies in how the algorithm is expressed. Recipes are written in English or other spoken languages while computer programs are written in programming languages or code. According to Leslie Lamport, winner of the 2013 Turing Award, thinking mathematically can be a useful step to specifying the algorithm for computer programmes, as it can help programmers clarify their thinking and make programs more efficient.

"Most programmers just start writing code; they don't even know what the algorithm is. It's like starting to build without a blueprint," said Dr. Lamport, speaking at an exclusive dialogue at the Singapore Management University (SMU) on 14 January 2020, held in conjunction with the SMU-Global Young Scientists Summit 2020.

"And the result? The program is hard to debug and inefficient because you would be trying to optimise at the code level rather than at the algorithm level. We should do what almost every other field of science and engineering does: initially describe the problem with math instead."

Why math is better than code

Using Euclid's algorithm as an example, Dr. Lamport walked the audience through how an algorithm can be expressed precisely yet simply with mathematics. Described by ancient Greek mathematician Euclid in 300 BC, Euclid's algorithm is a method for identifying the greatest common divisor (GCD) of two numbers, that is, the largest number that can divide the two numbers without leaving a remainder. For example, the GCD of the numbers 15 and 12 is 3.

The method is simple: subtract the smaller number from the larger number, then repeat this till both numbers are the same; the resulting number is the GCD. The entire procedure can be described in a single mathematical formula, said Dr. Lamport, who is recognised for developing the widely used LaTex file format, in addition to his pioneering work on distributed computing systems.

In contrast, writing Euclid's algorithm in code is more time consuming and cumbersome, and therefore harder to debug if it is not working correctly. "Euclid's program would have to contain a lot of lower level details, like what you should do if either number is less than or equal to zero," Dr. Lamport said. "You would have to decide that if you are writing a computer program but it's not the algorithm's problem."

How much more efficient would using math instead of code be? When engineers used TLA+, a high level formal specification language based on mathematics developed by Dr. Lamport to model, document and verify concurrent computing systems, they were able to dramatically reduce the size of an operating system originally used to control some experiments on the Rosetta spacecraft. "One of the results of specifying the software logic with TLA+ was that the code size was able to be reduced to about ten times less than the original," Dr. Lamport said. "You don't reduce the code size by ten times by better coding; you do it by cleaner architecture, which is just another word for a better algorithm ."

On top of being more efficient, taking a mathematical approach has the additional benefit of making de-bugging easier. Amazon Web Services and Microsoft Azure engineers use TLA+ for their cloud services, Dr. Lamport said, and through it have found bugs in their system designs that could not be found via any other technique.

Get comfortable with math

Although math is both powerful and elegant when it comes to describing algorithms, many people—including computer programmers and engineers—are intimidated by it and shy away from using it. "Some students have asked us when can they stop doing and reviewing the math and start the software programming," said Professor Steven Miller, Vice Provost (Research) at SMU and formerly the Founding Dean of the School of Information Systems.

Dr. Lamport believes that getting used to 'speaking' in mathematics is a matter of exposure. "Why is 'two plus two equals four' considered simple but a logical operation like 'an element of' is hard to understand for most people? Logical operations such as "element of" simply means that something is part of a bunch of other things. That concept doesn't require you to learn any complicated thing like counting, as counting is actually quite complicated," he said.

"Why should 'element of' seem frightening when 'plus' seems so easy? It's just a matter of not being familiar with it, and this is not all your own fault—mathematicians are terrible at teaching it."

For Dr. Lamport, becoming fluent in mathematics is the first step, but for mathematical thinking to truly impact the way algorithms are written, it has to change the way we think. "I want to emphasise that mathematics doesn't solve the problem for you; you have to solve the problem," he said. "Thinking mathematically will help you solve the problem; and mathematics helps to ensure that the solution was right."

Provided by Singapore Managment University