OPEN ACCESS GOVERNMENT

CLICK the TAB
to go to the section you want

HEALTH & SOCIAL CARE
SCIENCE & RESEARCH
FINLAND FOCUS
EDUCATION, TRAINING & HR
CULTURE & HERITAGE
FINANCE
THE BUILT ENVIRONMENT
ENVIRONMENT
ENERGY
TRANSPORT
CYBERSECURITY
ICT

ISSN 2055-7612 • May 2017

Image: © OHB

92

SPACE STRATEGY PUSHES THE BOUNDARIES OF KNOWLEDGE IN EUROPE

MAROŠ ŠEFČOVIČ, VICE PRESIDENT OF THE EUROPEAN COMMISSION EXPLAINS HOW THE SPACE SECTOR MAKES LIVES BETTER, SAFER AND HEALTHIER

IN THIS ISSUE

Supported by

UNIVERSITÀ DEGLI STUDI DI MILANO · VTT · The Balgrist · varberg · Center for Eldercare & Rehabilitation Technology, University of Missouri · ADJACENT DIGITAL POLITICS LIMITED

Improving trust in computer systems using formal proofs

Dale Miller, Inria, Université Paris-Saclay, LIX, École Polytechnique, and CNRS
highlights the role formal proofs can have in ensuring trust in computer systems

Computer systems are everywhere in our society and their integration with most parts of our lives is constantly increasing. This wide scale integration of computerised systems is accompanied by an increasing need to deal with their correctness in not only safety critical systems, such as those in cars, airplanes, missiles, and hospital equipment, but also general infrastructure systems such as financial databases, power grids, and telecommunication. Even in the area of consumer electronics there are growing concerns about many aspects of their correctness. For example, years ago, establishing the correctness of, say, desktop PCs, music players, and telephones was not urgent since rebooting such systems to recover from errors or bugs was mostly just a nuisance. But today, these same devices are tightly integrated into networks and, hence, must deal correctly and securely with their user's confidential information.

Security

As it is painfully clear today, connecting your computer to the internet is similar to submerging a submarine into the depth of the sea: if there is a crack in your system, it will be exploited quickly and with serious consequences. Attempting to establish some formal guarantees about software systems is no longer an academic curiosity. If we cannot provide basic guarantees of the correctness and security of our computer systems, our future could resemble William Gibson's world in his novel Virtual Light, where network security was so bad that important data was transferred by bikers carrying hard-disks. In such a world, the development of all the new features and services arising from networking and sharing, and the concomitant increases in efficiency and productivity, that we all hope to see unfold would be greatly delayed.

Whilst it is possible to increase one's trust in software by employing various management techniques during the construction and maintenance of computer systems or by making use of standard, open source systems, current experience with these approaches still results in insecure software. There is, however, a growing interest in taking the extreme position of treating computer systems as mathematical structures and formally proving some correctness and security properties for them. While achieving mathematical certainty about some aspects of computer systems can go a long way to increasing trust in the correctness of such systems, obtaining formal proofs is a complex and difficult process with many moving parts, some of which must also be trusted. For example, programming languages, compilers, and machine architectures all need to have mathematical descriptions. One may need to prove properties (invariants) about the execution of any program. One needs to prove that a compiler transforms a program in a high-level language to a block of machine code in such a way that the meaning of the original program is preserved. A number of other components of a programming environment may also need to be trusted, including, for example, parsers, printers, type checkers, verification condition generators, and linkers. Then, of course, there is also the infrastructure needed to support the activity of building formal proofs. While the world of proofs in mathematics are usually produced by humans for other humans to read and to learn from, the world of formal proofs involving computer systems are tedious, detailed, and long. Such proofs can only be checked by other computer systems called proof checkers. If we are not careful, we might have replaced the problem of trusting one's original computer system by a long chain of tools that all need to be trusted.

Researchers are, in fact, working on many different parts of this chain of tools in order to increase our trust in those tools. In the end, if one can really construct a formal proof that clearly proves some property holds of a computer system, then one does not need to trust the reputation of a particular compiler or some team of programmers. Instead, we can invoke the bedrock of trust that arises from the scientific method reproducibility by enabling any number of sceptics to implement proof checkers to check

the claim that a given formal proof does, in fact, prove the theorem claimed. If different programmers working with different programming languages at different times on different computer hardware are all capable of verifying a formal proof, then we can trust the theorems established by such proofs as strongly as we trust anything established by the scientific method.

> "As it is painfully clear today, connecting your computer to the internet is similar to submerging a submarine into the depth of the sea: if there is a crack in your system, it will be exploited quickly and with serious consequences."

While formal proofs and reproducible proof checking can provide a trustable framework for the development of trustable computer systems, state-of-the-art theorem provers, the major source of formal proofs, do not contribute to this framework for at least 2 reasons. First, there is a large number of different provers and they collectively build proofs in a wide array of different formats. Such formats range from ad hoc designs used in specialised situations, to proof scripts that describe how to lead a particular interactive theorem prover to a proof, to any of the multitude of textbook formats such as resolution refutations, natural deduction, bi-simulations, etc. Second, in the case that a prover is willing to export their proofs, the actual format of their output is usually idiosyncratic and ill-defined.

ProofCert Project

Within the ProofCert project, which I have been leading for the past 5 years, we have turned to proof theory, a topic of mathematical logic that began in the 1930s and has been slowly evolving since then. Using proof theory, we have developed the foundational proof certificates framework for providing formal definitions of a wide range of proof systems. Given its roots in logic, this framework is both technology-independent and involves implementation techniques that have been well studied and analysed for the past few decades. Anyone interested in implementing a checker of proofs defined using this framework can easily understand exactly what needs to be implemented, as well as find a rich literature describing all the necessary algorithms. The framework provided by the ProofCert project can now be exploited to make formal proofs universal and as trustworthy as needed. With this method of minting the basic coins of trust, the formal methods community can continue building the infrastructure that allows us to trust more aspects of more of our computer systems.

Dale Miller
Inria, Université Paris-Saclay, LIX,
École Polytechnique, and CNRS
dale@lix.polytechnique.fr
www.lix.polytechnique.fr/Labo/Dale.Miller/
http://team.inria.fr/parsifal/proofcert/