OK computer: an automatic for the people

Associate Professor Laura Kovács leads a team combining symbolic computation methods with computer theorem proving – research especially relevant to a world increasingly dependent on software. The development of automated approaches to verify and analyse software will cost-effectively improve reliability involving complicated control-flow, use of mixed data structures and non-trivial arithmetical operations, the expressiveness of the language in which program properties can be formulated, and the limitations of modern theorem provers to reason and prove program properties.

My work addresses the above challenges in the following ways: my group develops new methods for generating properties of programs with complicated logical flow. We also handle programs with unbounded data structures, nested conditionals and polynomial operations over numerical data types. For such programs, we automatically derive loop properties, such as loop invariants, Craig interpolants and bounds on the number of loop iterations. On the other hand, we use full first-order logic to express program properties and extend automated theorem provers to support and work with such extensions.

You are designing efficient techniques for automatically proving properties of complex software systems. What impact do you anticipate this will have?

I expect my group’s work on proving properties with theories and quantifiers to have a deep and long-lasting impact in program analysis, bringing us closer to the goal of automatically analysing and verifying programs with millions of lines of code. I’m confident our research will yield new results in reasoning-based program analysis – new methods in theorem proving can often solve many new problems.

Can you explain your most significant achievements in your research to date?

Probably my biggest scientific achievement came with the introduction of the symbol elimination method in 2009. Various techniques used in program analysis, such as Gröbner basis computation, quantifier elimination and Craig interpolation, can be considered as applications, or special cases, of symbol elimination. Thanks to my previous joint work with Dassault Aviation and Intel Haifa, I made symbol elimination a useful and powerful tool in program analysis.

I’m a founding member of the Austrian Society for Rigorous Systems Engineering (ARiSE) and a principal investigator of the National Research Network ‘RiSE’ project of the Austrian Science Fund. After joining Chalmers, I was awarded a research grant for junior researchers by the Swedish Research Council. Recently, I’ve become a Wallenberg Academy Research Fellow 2014 – Sweden’s largest private investment programme in young researchers. My research has also been awarded a European Research Council Starting Grant 2014, Europe’s most competitive research programme for young scientists.

Finally, how do you see your research progressing over the next decade?

I will continue to carry out research in all aspects of reasoning-based program analysis: theory, implementation, use, applications and the development of very powerful computer tools supporting my research. I am the co-developer of the award-winning Vampire theorem prover, which is developed jointly by the team of Professor Andrei Voronkov at the University of Manchester and my group at Chalmers.

I will also maintain the international scientific visibility and competitiveness of my group, engage my group in collaborations with other software verification teams and publish our results at top scientific venues in automated reasoning.
Researchers at Chalmers University of Technology in Sweden have adopted a unique approach to automating software verification. Their development of symbol elimination is a world first, requiring no user guidance and boasting far-reaching industrial applications.

**Approaching programs from a mathematical standpoint**

Software can be broadly understood as a term denoting the various types of programs that direct the operations of computers and related devices. Ada Lovelace is credited with outlining what would have been the first piece of software (had it been created) in the 19th Century, but it was Alan Turing who proposed the first theory of software in 1935. Since then, computers and – by proxy – software, have become an essential part of our everyday lives, enabling the successful functioning of things such as cars, aeroplanes and smartphones.

Indeed, as the technological developments of the 21st Century continue to become more and more complex, so too does the software upon which they rely. Unfortunately, despite the fact that many of these improvements in technology are designed to make our lives better and easier, their operation becomes ever more complicated. While the languages and tools used by programmers increase productivity, they also have the undesired effect of perpetuating problems with the reliability, safety and security of the software produced.

This presents a major, global problem, as error-prone and insecure systems cost the world economy billions upon billions of pounds per year. Additionally, there is the potential cost to human lives; bugs in flight control software or medical equipment, for example, could have serious and even fatal consequences.

One means of addressing these issues is to manually inspect software programs to determine the potential sources of errors. However, from a commercial and practical point of view, this is often not a viable option. It is therefore essential to find a better means of ensuring reliable software to avoid the significant costs associated with their failure.

**Automatically verifying and analysing**

In response to this, researchers have focused on developing automated software verification and analysis approaches to find and correct errors in a cost-effective and logistically viable way. Associate Professor Laura Kovács leads a team at Chalmers University of Technology who is working on computer theorem proving and using computer programs to solve problems in other computer programs.

The key idea behind automated software verification and analysis lies in treating programs as mathematical objects. In so doing, their properties can be proven in similar ways to theorems. Using this approach, Kovács’ team has developed unique methods of reasoning and reasoning-based program analysis. “We combine symbolic computation techniques, such as algorithmic combinatorics, Gröbner basis computation and quantifier elimination, with first-order theorem proving,” explains Kovács. “Such a combination is not yet used in state-of-the-art program analysis tools, so our work allows us to derive properties of programs involving arithmetic and logical operations over the computer memory and numeric variables.”

**Perfectly timed findings**

While other researchers make use of theorem provers to only solve program properties, Kovács’ team uses them to generate program properties. In addition, their work makes use of logical quantifiers and symbols from mathematical theories to generate these properties. The findings from Kovács’ work have tremendous applications for a range of industrially relevant problems, such as expressing that a computer’s memory has been accessed at a valid or existing position – a property essential in verifying specific device drivers.
AUTOMATED SOFTWARE VERIFICATION

OBJECTIVE
To solve logically complex parts of software programs through computer theorem proving and computer algebra.

COLLABORATORS
Associate Professor Wolfgang Ahrendt, Chalmers University of Technology, Sweden

Professor Andrei Vorkonkov, University of Manchester, UK, and Chalmers University of Technology, Sweden

Professor Jens Knoop, Vienna University of Technology, Austria

PARTNERS
Faculty of Informatics, Vienna University of Technology, Austria

FUNDING
Austrian Science Fund
European Research Council
Swedish Research Council
Wallenberg Foundation

CONTACT
Laura Kovács
Associate Professor
Department of Computer Science and Engineering
Chalmers University of Technology
41296 Gothenburg
Sweden

T +46 31 772 1696
E laura.kovacs@chalmers.se
http://www.cse.chalmers.se/~laurako/

ELIMINATING SYMBOLS TO PROPAGATE RELIABILITY

The team that Associate Professor Laura Kovács leads has recently developed a unique computer program that has significant potential to solve problems with other computer programs. Called symbol elimination, it helps to develop reliable software and, unlike other approaches, it requires no user guidance.

Importantly, symbol elimination is the first ever method to automatically discover complex program properties with an arbitrary use of logical quantifiers. Now, Kovács uses symbol elimination in her work to combine first-order theorem proving and symbolic computation.

IT WORKS AS FOLLOWS:

- The team is given a program, then extends the language of that program with additional symbols (variables, function/predicate symbols), such as loop counters.

- Next, they automatically discover the properties of the program that can be expressed by first-order logic formulae in these extended languages. For a program loop manipulating an array, for example, the team can derive a property expressing at which positions the array was updated and by what value.

- To generate program properties that can be used by software developers, those properties need to be in the program’s original language. The auxiliary symbols from the properties generated in the extended language are therefore eliminated using a first-order theorem prover. Thus, the resultant program properties only use symbols from the input program and can be used by software developers.

Solving logical puzzles using newly developed features in computer theorem proving:

However, there are further applications too: “Our work enables the timing analysis of computer systems, ensuring that a computer program reacts or terminates within a given time limit,” explains Kovács. “A key example of this technology is in the braking system of a car, where reacting on time is crucial to its effectiveness. To ensure this function works, we derive and prove the worst-case execution time estimates within computer systems.”

AVOIDING ERRORS AND SAVING MONEY

By combining symbolic computation methods with computer theorem proving for program analysis, Kovács’ team distinguishes itself from others working in the field. Close ties with manufacturers who can make use of their tools in program analysis has ensured their research has practical applications, not least through preventing programmers from introducing errors while making changes to software. The fewer errors contained within programs, the less time it takes for those programs to be developed – a crucial step in achieving automated verification.

Not only does Kovács’ research facilitate the implementation of world-leading tools with huge potential to save billions of pounds around the world each year, it could also save human lives.