THE FORMAL VERIFICATION OF LARGE SOFTWARE SYSTEMS

Victor Dumitrescu, supervised by Jacques Fleuriot
Pervasive Parallelism - 2015 Cohort
How do we make sure that software systems do what they’re supposed to?
How do we make sure that software systems do what they’re supposed to?

· fully automated methods
How do we make sure that software systems do what they’re supposed to?

- fully automated methods
- interactive theorem proving
Interactive theorem proving can offer the highest degree of flexibility and assurance.

But... requires significant effort and skill from developers. Has mostly been used for the verification of:

- processor architectures
- OS kernels
- compilers and interpreters
- safety-critical algorithms
Proofs developed for:

- functional correctness (code and binary level)
- security properties of specification (integrity, confidentiality, availability)

Ongoing development and maintenance.
Huge effort:

- 500 proof files
- 95,000 proven theorems
- 500,000 lines of proof code
- 17 collaborators working for 5 years
A Verified Microkernel

Huge effort:

- 500 proof files
- 95,000 proven theorems
- 500,000 lines of proof code
- 17 collaborators working for 5 years

Used in real-world applications

- aviation & space flight
- automotive
- consumer devices
Proof engineering is an emerging discipline seeking to make proof development more accessible:

- better proof languages and libraries
- better tool support and automation
- better process management
Industry interest in formal methods is growing.

Formal verification of real-world systems is possible.
Industry interest in formal methods is growing.

Formal verification of real-world systems is possible.

Trustworthiness of computer systems is vital as technology plays an increasingly important role in our daily lives.