some thoughts on machine-assisted proofs

luís cruz-filipe

department of mathematics and computer science
university of southern denmark

international congress of mathematicians
rio de janeiro, brazil
august 7th, 2018

a not-so-new trend in mathematics

the 4-color theorem

Appel, Haken and Koch (1977)
traditionally considered "the" birth of the field

more than 10 years before...

Theorem 5. $S(7) = 16$.

Proof. This theorem was proved by exhaustive enumeration on a CDC G-21 computer at Carnegie Institute of Technology in 1966. The program was written by Mr. Richard Grove, and its running time was approximately

(Floyd & Knuth, 1973)

the present day

software and hardware verification

- critical systems (testing is not enough)
- lots of mechanical, "boring" proofs with lots of (simple) cases
- often largely/completely automatic

mathematical results

- because we can
- proofs in "mathematical style", typically interactive
- "less elegant" proofs by encoding, often automatic

two styles of proving

ad hoc programs

highly specialized programs check that some property holds
(cf. early examples)

- the program must be correct
- not always easy to trust

theorem provers

general-purpose programs that can construct/check proofs in a particular logic

- we still need to trust the program (but...)
- the encoding in the logic must be correct

an example

optimal sorting networks

same domain as Floyd and Knuth, proving $S(9) = 25$

- *ad-hoc* Prolog program, following "good" practices
- independently verified by encoding in propositional logic
- algorithm rerun by a provenly correct program

why so much work?

- can we trust Prolog?
- can we trust sat-solvers (more on that later)?
- is the sat encoding correct? (it wasn't – several times)
- certified programs are typically MUCH slower

another example

sat solving

general problem: is a given propositional formula satisfiable?

- very efficient solvers exist, able to deal with gigantic formulas
- usable in practice to solve other problems by encoding
- nearly impossible to understand the code

recent trend

independently check a trace of the sat solver's "reasoning"

- checking a proof is much easier than finding it
- possible to do efficiently
- state-of-the-art traces of around 400 TB