C functional correctness verification comparison (C-verif-mark?)

Peter Sewell
University of Cambridge

2022-08-03 Isaac Newton Institute, Verified Software (VSO2)

C functional correctness verification

Crucial – especially for security-critical systems code

Still a major research problem – despite many impressive projects, it's still much harder and more limited than we would like

C functional correctness verification

Crucial – especially for security-critical systems code

Still a major research problem – despite many impressive projects, it's still much harder and more limited than we would like

Hard to understand the state of the art:

- there are almost as many approaches as there are verification projecs
- the papers don't always give a clear view of their current strengths, limitations, and future plans,
- ...or an overview of the design challenges and best solutions.

C functional correctness verification

Crucial – especially for security-critical systems code

Still a major research problem – despite many impressive projects, it's still much harder and more limited than we would like

Hard to understand the state of the art:

- there are almost as many approaches as there are verification projecs
- the papers don't always give a clear view of their current strengths, limitations, and future plans,
- ...or an overview of the design challenges and best solutions.

So, can we establish a better comparison, and shared understanding of the verification-tool design space and alternatives?

Previous comparisons

Automated tools: many very successful (SV-COMP,...)

Interactive tools:

- A benchmark for C program verification
- lets-prove-leftpad.
- VerifyThis, 2011-2021.
- The 2nd Verified Software Competition: Experience Report, 2011.
- The 1st Verified Software Competition: Extended Experience Report

Previous comparisons

Automated tools: many very successful (SV-COMP,...)

Interactive tools:

- A benchmark for C program verification
- lets-prove-leftpad.
- VerifyThis, 2011-2021.
- The 2nd Verified Software Competition: Experience Report, 2011.
- The 1st Verified Software Competition: Extended Experience Report

Not quite hitting the spot?

Goals

- lightweight (shared github repo, no separate assessment)
- aiming to expose a clear comparison in this multidimensional space (not a "competition" – no judging, scores, or winners)
- aiming for better understanding of how the different approaches vary
- aiming to stimulate future research

Mechanisms

- these talks
- smallish collection of smallish examples
  - exercising various C language features and programming idioms
  - small enough for solutions to not be too much work
  - ask (for at least some) for the most *instructive* solution, not the shortest, and for detailed explanations of what's going on under the hood
- larger list of C language features and programming idioms
- one or two large examples to focus on scaling
- (ideally, ultimately) consensus list of main design challenges and options