The authors dream of combining a fully-abstract compiler with protected-memory hardware to enable modular verification of C programs via separation logic. We readers are dazzled with a proof that static Hoare logic assertions are never invalidated *at runtime* and impressive benchmark results ($< 10\%$ overhead) from a closed-source OCaml binary. In all, a compelling work of fan fiction [1].

**Strengths**

- Clever combination of features. I hope that hardware designers soon give us a fast way to check memory integrity.
- Very well-written introduction.

**Weaknesses**

- The references to (modular) verification tools were good motivation; I would have also liked citations to companies using these tools. Especially, I would have liked to know of companies writing high-security code that still use "untrusted" code—either as a library or because they cannot afford complete verification.
- The macro-benchmarks were not very convincing; `gcc -O3` is far from fully-abstract! More detailed Sancus results would have been more useful than the benchmarks in §7.2.
- On that note, I don't think "fast" and "fully-abstract" will ever coexist. The latter rules out too many of the optimizations. Shame on these guys for trying to keep us programming in C `;)`.
- In §7.4, I don't understand how copying the memory region could be cheaper than hashing. Is copying less expensive than multiplication? Maybe they need a faster (rolling) hash algorithm.
- Blame is *not* orthogonal to this work. Part of their informal guarantee is that bugs triggered in unverified code are detected. This idea of correctly blaming the unverified code should be formalized.

Who is P. W. O'Hearn?

# References

[1] Pieter Agten, Bart Jacobs, and Frank Piessens. Sound modular verification of C code executing in an unverified context. In *POPL*, 2015.