3C: Interactive Conversion of C to Checked C

Submitted by mwhicks1 on Fri, 08/14/2020 - 12:14pm. Contributors:
Michael W. Hicks Aravind Machiryy Ray Chen Hasan Touma Aaron Eline

Checked C\(^1\) is a freely available, backward compatible extension to C that aims to support spatial memory safety. Checked C defines annotations for pointer types that its compiler uses to verify, either statically or dynamically, that pointer accesses are safe. Such safety assurance is useful in itself (e.g., it precludes buffer overflows) and also sets a useful foundation for subsequent static analyzers.

To streamline the process of upgrading a legacy C program to Checked C, we have been developing a tool, 3C, to automatically infer Checked C annotations. 3C is designed to be interactive, in the sense that a developer will use it repeatedly on the same codebase. As shown in Figure 1, 3C performs a wholeprogram, constraint-based static analysis to infer safe-pointer annotations. The developer can make changes to the resulting code, e.g., to correct mistakes or remove anti-patterns (e.g., unsafe casts) or to add missing annotations (e.g., due to hard-to-discover loop invariants), and then re-run 3C to perform further work.

To support this workflow, 3C's static analysis has two novel elements. First, it localizes its determination of when pointers can be deemed safe, and expresses that at the function boundary. If a function void foo(int *x) { ... } uses its parameter x safely, then it can be annotated as safe in the definition, even if a caller passes in an unsafe argument, e.g., foo((int *)5)--we deem this a problem with the caller, not with foo. Conversely, if foo uses x unsafely then x is not annotated (as such, a caller foo(y) would have its argument y deemed unsafe). 3C can point out the "root cause" for a safety problem, which the programmer can fix prior to re-running 3C.

Second, 3C is able to analyze code that is a mix of legacy and Checked C code, even when that mix is self-inconsistent, e.g., because it assigns a legacy (unsafe) pointer to a safe (annotated) one. With the experience of converting 100s of KLoC, we find that 3C's interactive approach speeds up the overall process of converting code because leverages the particular strengths of the human and machine components of the conversion process. Our talk will describe our approach, our experience using it, and the challenges we are addressing in ongoing work. 3C is part of the public release of Checked C.

\(^1\)https://github.com/microsoft/checkedc
Michael W. Hicks is a Professor in the Computer Science Department at the University of Maryland, where he has been since 2002, and the CTO of Correct Computation, Inc. (CCI), a role he has held since late 2018. He is a leading member of the software security community, having carried out diverse and award-winning research published in top venues covering computer security, programming languages, systems, and software engineering. He was the first Director of the University of Maryland's Cybersecurity Center (MC2), and was the elected Chair of ACM SIGPLAN, the Special Interest Group on Programming Languages; he now edits its blog, PL Perspectives, at https://blog.sigplan.org. Over his career, he has published more than 125 peer-reviewed scientific articles, and his research has twice won the NSA's Best Scientific Cybersecurity Paper competition; he is the only two-time winner. A key recent focus at UMD and CCI has been to explore how to make legacy software written in low-level languages, like C, more secure. He has been collaborating on the Checked C project, carrying on a nearly 20-year arc of work that started with the Cyclone safe programming language, which itself was a strong influence on Rust, a next-generation language. He has also currently looking at synergies between cryptography and programming languages; techniques for better random (fuzz) testing and probabilistic reasoning; and high-assurance tools and languages for quantum computing.

![Diagram of workflow](image)

**Figure 1:** Workflow of using 3C to refactor a program from C to Checked C

**Michael W. Hicks**  |  **Aravind Machiryy**  |  **Ray Chen**  |  **Hasan Touma**  |  **Aaron Eline**
---|---|---|---|---
**License:** Creative Commons 2.5

Other available formats:

3C: Interactive Conversion of C to Checked C
Switch to normal viewer
Switch to experimental viewer