ICAI 2023
12th International Conference on Applied Informatics
Eger, Hungary, 2023

Visualization for Read-Copy-Update Synchronization Contexts in C Code

Endre Fülöp[a], **Attila Gyén**[a], **Norbert Pataki**[a]

[a]Department of Programming Languages and Compilers
Eötvös Loránd University
gamesh411@gmail.com, gyenattila@gmail.com, patakino@elte.hu

Abstract

The Read-Copy-Update (RCU) mechanism is a way of synchronizing concurrent access to variables with the goal of prioritizing read performance over strict consistency guarantees [4]. The main idea behind this mechanism is that RCU avoids the use of lock primitives while multiple threads try to read and update elements concurrently. In this case, elements are linked together through pointers in a shared data structure. RCU is used in the Linux kernel, but there are user-space libraries which implement the technique as well [3]. One of the user-space solutions is librcu that is a C language library [1].

Our earlied work developed a code comprehension framework for the RCU synchronization contexts [2]. We have developed a static analysis solution based on the Clang compiler. Our static analysis tool takes advantage of the LLVM IR (Intermediate Representation) which is generated from the source code.

For context detection, the iterative algorithm of forward dataflow analyses uses reverse postorder traversal of the control-flow graph (CFG) elements in case of forward analysis in order for performance reasons. This results in a scalable method for gaining an overview about the synchronization aspects of the software. The modular nature of the approach lends itself to distributed use.

The transfer function saves the interesting locations (the instructions that can be used to get the locations), by appending them to the basic block level global fact, but only if this global fact is does not already contain them. In addition, if a context ending API call is detected, the exit state of the instruction set to the current global state of the basic block. The reverse postorder visitation guarantees, if a context starting instruction then happens to precede a context ending one, there

is path in the CFG from the starter to the ending one. The set-like nature of the list in turn allows for the halting of the fixed-point algorithm in finite steps, as there are a finite amount of interesting locations inside a program.

The meet function is responsible for merging the exit states of multiple incoming dataflow facts. This is defined as the concatenation of the dataflow fact lists in a manner, that guarantees uniqueness of elements inside the resulting list, and the preservation of relative ordering among the interesting locations.

Monaco Editor is maintained by Microsoft and available worldwide for free [5]. Figure 1 shows its default appearance. Monaco has a playground with full of interactive examples and provides wide access to the editor and it supports feature like colorize the editor line-by-line, add different error and warning messages or add a hover message when the cursor is hovered over the text. Doing all this with JavaScript programming language for the dynamic parts, CSS for styling and HTML to build the raw frame. It gives full access to the Document Object Model (DOM) supplemented by its own special elements. However, it sets up some limitations.

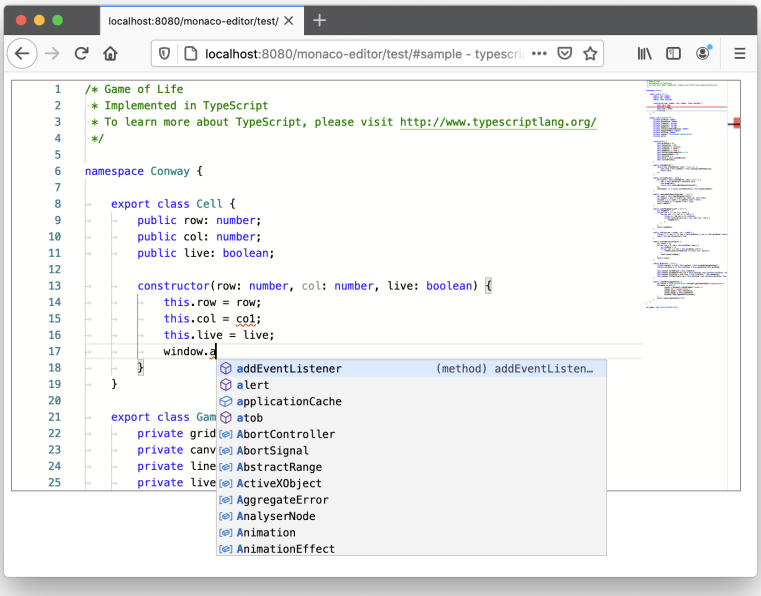

Figure 1. Monaco Editor

In this paper, we present our visualization techniques and methods for RCU contexts in the Microsoft's Monaco Editor. We cover many aspects of this style of concurrency for an improved code comprehension. We present how to visualize the read/write locks and contexts. This approach can help the developers to find

bugs in the code without its execution. We show how the shared variables can be highlighted in this paradigm. The proposed visualization and its implementation details are demonstrated.

References

[1] M. Desnoyers, P. E. McKenney: *Userspace RCU*, https://liburcu.org/.

[2] E. Fülöp, A. Gyén, N. Pataki: *Code Comprehension for Read-Copy-Update Synchronization Contexts in C Code*, in: Geoinformatics and Data Analysis, ed. by S. Bourennane, P. Kubicek, Cham: Springer International Publishing, 2022, pp. 187–200, isbn: 978-3-031-08017-3.

[3] G. Márton, I. Szekeres, Z. Porkoláb: *Towards a High-level C++ Abstraction To Utilize The Read-Copy-Update Pattern*, Acta Electrotechnica et Informatica 18.3 (2018), pp. 18–26, doi: 0.15546/aeei-2018-0021.

[4] P. E. McKenney, J. Walpole: *What is RCU, fundamentally?*, 2007, url: https://lwn.net/Articles/262464/.

[5] Microsoft: *Monaco Editor*, https://microsoft.github.io/monaco-editor/.