Generating pseudocode from source code

December 2022

candidate	Sergey Jakobsen
supervisors	Lars Tveito and Michael Kirkedal Thomsen
group	Programming Technology (PT)
type	60ECTS
study program	Informatics: Programming and System Architecture
planned date of completion	May 2024

Background and motivation

Pseudocode is commonly used for providing a description of an algorithm at a suitable level of abstraction, especially in educational and scientific literature. There is no standardisation or formal semantics for pseudocode. Having a non-executable language gives the author complete freedom to omit details or de-emphasize certain aspects of an algorithm.

Since it is (deliberately) non-executable, it leaves the author (and the audience) unable to test it. This can lead to accidental inclusion of critical inaccuracies.

Correct presentations are especially important in academia, where the goal is to teach students concepts they were previously unfamiliar with. Traditionally, concepts within the algorithms and data structures realm have proved challenging for undergraduates. If then their first impression of an algorithm is an incorrect presentation, their path is already hampered.

The aim of this master thesis is to explore the possibilities of generating pseudocode directly from source code. Algorithms from an introductory course on algorithms and datastructures will be used as a case study.

Methodology

The thesis will consist of both theoretical and practical work. The theoretical work will consist of reading relevant research, studying code, and exploring macros features in functional programming languages. I will spend some time examining the most sensible way of designing the DSL/tool. At the moment there seems to be two plausible albeit different approaches:

- Writing an interpreter/compiler for a DSL specifically designed for this purpose.
- Extending an existing language like Racket/Clojure/Common Lisp with macros.

The practical work will consist of implementing software features (either in an existing language or a brand-new one) to generate pseudocode, tests and visualisations from a programs source code. I might also carry out independent research on how pseudo code and visualisations can aid students in grasping key concepts of (primarily) algorithms and datastructures.

Progress plan

Spring 2023

- Explore literature relevant to the thesis and review what has already been done on the topic

- Explore built-in macros features in different programming languages.
- Decide on approach regarding how to generate pseudocode from source code
- Formulate a thesis and write essay based on this

Autumn 2023

- Interview students enrolled in "Algorithms and Datastructures" and "Algorithms: Design and Efficiency" at the Univeristy of Oslo
- Either write interpreter/compiler for a new language, or extend existing language with macros

Spring 2024

- Finish developing the tool
- Finish writing on the thesis

Curriculum

Relevant courses

- IN4000 - Operativsystemer
- IN4110 - Problemløsning med høynivå-språk
- INF5110 - Kompilatorteknikk
- IN5800 - Declarative Data Engineering

Relevant readings

- QuickCheck: a lightweight tool for random testing of Haskell programs [1]
- Hypothesis: A new approach to property-based testing [2]
- Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation [3]
- Structured flowcharts outperform pseudocode: an experimental comparison [4]

References

[1] K. Claessen and J. Hughes. "QuickCheck: a lightweight tool for random testing of Haskell programs". In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*. Ed. by M. Odersky and P. Wadler. ACM, 2000, pp. 268–279.

[2] D. Maciver and Z. Hatfield-Dodds. "Hypothesis: A new approach to property-based testing". In: *J. Open Source Softw.* 4.43 (2019), p. 1891.

[3] Y. Oda et al. "Learning to Generate Pseudo-Code from Source Code Using Statistical Machine Translation (T)". In: *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*. Ed. by M. B. Cohen, L. Grunske, and M. Whalen. IEEE Computer Society, 2015, pp. 574–584.

[4] D. A. Scanlan. "Structured Flowcharts Outperform Pseudocode: An Experimental Comparison". In: *IEEE Softw.* 6.5 (1989), pp. 28–36.