Applicability of Effect Handler Oriented Programming (EHOP) for Text-Based Game Development

1. Background

- Effect handler oriented programming (EHOP): a recently proposed programming paradigm.
- Main goal of EHOP: provide a higher degree of separation of concerns than other programming paradigms by using effects and effect handlers [1], in order to isolate the handling of side effects from the main application logic.
- State-of-the-art programming languages which support EHOP: Koka [2], Frank [3], Haskell [4] (through the use of libraries).
- Main idea of EHOP: a program's source code is implicitly divided into the following three parts [5]:
  - Effect signatures;
  - Effect handlers;
  - Effectful functions.

```
// Emitting messages; how to emit is TBD.  Just one abstract operation: emit.
effect fun emit(msg : string) : ()

// Emits a standard greeting.
fun hello()
  emit("hello world!")

// Emits a standard greeting to the console.
pub fun hello-console1()
  with handler
    fun emit(msg) println(msg)
  hello()
```

Figure 1: An example of the declaration, usage and handling of a simple emit effect in the Koka programming language. [6]

- In theory, EHOP seems to be a very promising programming paradigm. However, its practical applicability as a software development tool is yet to be explored.

2. Research Question

When EHOP is used for text-based game development, to what extent does the support for separation of concerns of EHOP affect the modularity, readability and maintainability of the source code, compared to "traditional" implementations?

3. Methodology

Three main sub-tasks:
1. Choose a text-based game and implement it in Koka;
2. Assess the following qualitative aspects of the game's source code: modularity, readability, maintainability;
3. Analyse the following quantitative aspects of the game's source code: compilation time, start time, peak memory usage, average time needed to execute a player's turn.

4.Results

Initially, a fully-featured shared-screen multiplayer text-based chess game was developed entirely in Koka.[1]

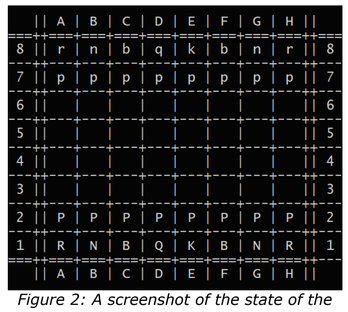

Figure 2: A screenshot of the state of the board at the beginning of a game.

Results of the qualitative assessment of the source code of the text-based chess game:
- Significantly increased modularity;
- Mildly improved readability;
- Substantially enhanced maintainability.

Results of the quantitative analysis of the source code of the text-based chess game:

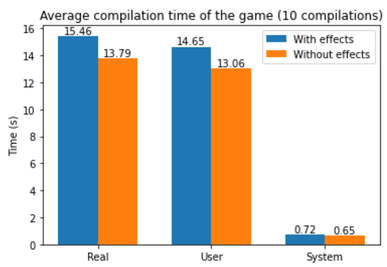

Figure 3: The average compilation time of the finished text-based chess game with and without the usage of custom effects.

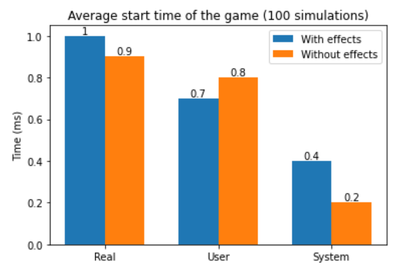

Figure 4: The average start time of the finished text-based chess game with and without the usage of custom effects.

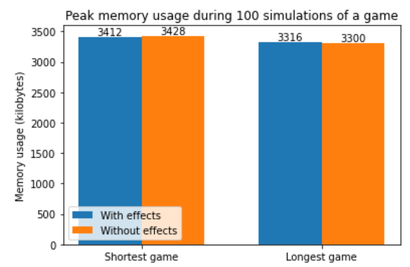

Figure 5: The peak memory usage measured during one hundred simulations of a game with and without the usage of custom effects.

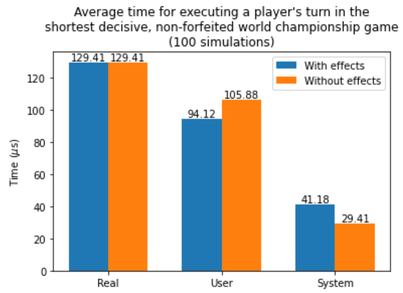

Figure 6: The average time needed to execute a player's turn in a simulation of the shortest decisive, non-forfeited world championship game with and without the usage of custom effects.

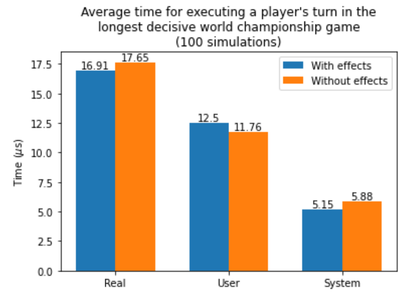

Figure 7: The average time needed to execute a player's turn in a simulation of the longest decisive world championship game with and without the usage of custom effects.

Overall, the quantitative analysis revealed that the usage of EHOP for text-based game development has almost no downsides with regard to the quantitative aspects of the source code.

[1] Available at: https://github.com/ivanstodorov/chess-game-koka

5. Conclusion

There are substantial benefits to using EHOP for text-based game development. It significantly improves the modularity, readability and maintainability of the source code at the cost of very little to no performance.

6. Future Work

- Develop a software quality tool, which supports Koka.
- Implement the exact same text-based game in either an object-oriented or functional programming language.
- Implement the exact same text-based game in another EHOP language.
- Explore a different potential area of application of EHOP.

7. References

[1] G. Plotkin and M. Pretnar, "Handlers of algebraic effects," in European Symposium on Programming, pp. 80–94, Springer, 2009.
[2] D. Leijen, "Type directed compilation of row-typed algebraic effects," in Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, pp. 486–499, 2017.
[3] L. Convent, S. Lindley, C. McBride, and C. McLaughlin, "Doo bee doo bee doo," Journal of Functional Programming, vol. 30, 2020.
[4] "Haskell: An advanced, purely functional programming language." https://www.haskell.org/. Accessed: 2022-06-14.
[5] J. I. Brachthauser, P. Schuster, and K. Ostermann, "Effect handlers for the masses," Proceedings of the ACM on Programming Languages, vol. 2, no. OOPSLA, pp. 1–27, 2018.
[6] "The Book of Koka: The Koka Programming Language." https://koka-lang.github.io/koka/doc/book.html. Accessed: 2022-05-30.

Author: Ivan Todorov
Responsible Professor: Casper Bach Poulsen
Supervisors: Cas van der Rest
Jaro Reinders

CSE3000 Research Project
June 2022

TU Delft Delft University of Technology