修士 (工学) 学位論文

Master's Thesis of Engineering

GearsOS での HoareLogic を用いた実装と検証

2020 年 3 月

March 2020

外間 政尊

Masataka HOKAMA

琉球大学

大学院理工学研究科

情報工学専攻

Information Engineering Course

Graduate School of Engineering and Science

University of the Ryukyus

指導教員：教授 玉城 史朗

Supervisor: Prof. Shirou TAMAKI

本論文は、修士 (工学) の学位論文として適切であると認める。

論　文　審　査　会

印

(主　査)　　和田　知久

印

(副　査)　　名嘉村　盛和

印

(副　査)　　長田　智和

印

(副　査)　　河野　真治

要旨

OS やアプリケーションの信頼性は重要である。信頼性を上げるにはプログラムが仕様を満たしていることを検証する必要がある。プログラムの検証手法として、Floyd–Hoare logic (以下 Hoare Logic) が存在している。HoareLogic は事前条件が成り立っているときにある関数を実行して、それが停止する際に事後条件を満たすことを確認することで、検証を行う。HoareLogic はシンプルなアプローチであるが通常のプログラミング言語で使用することができず、広まっているとはいえない。

当研究室では信頼性の高い OS として GearsOS を開発している。現在 GearsOS では CodeGear、DataGear という単位を用いてプログラムを記述する手法を用いており、仕様の確認には定理証明系である Agda を用いている。

CodeGear は Agda 上では継続渡しの記述を用いた関数として記述する。また、継続にある関数を実行するための事前条件や事後条件などをもたせることが可能である。

そのため Hoare Logic と CodeGear、DataGear という単位を用いたプログラミング手法記述とは相性が良く、既存の言語とは異なり HoareLogic を使ったプログラミングが容易に行えると考えている。

本研究では Agda 上での HoareLogic の記述を使い、簡単な while Loop のプログラムの作成、証明を行った。また、GearsOS の仕様確認のために CodeGear、DataGear という単位を用いた記述で Hoare Logic をベースとした while Loop プログラムを記述、その証明を行なった。

Abstract

研究関連業績

1. 外間政尊, 河野真治. GearsOS の Agda による記述と検証. 研究報告システムソフトウェアとオペレーティング・システム（OS), May, 2018
2. 外間政尊, 河野真治. GearsOS の Hoare Logic をベースにした検証手法. 電子情報通信学会 ソフトウェアサイエンス研究会 (SIGSS) 1 月, Jan, 2019

目 次

図 目 次

表 目 次

ソースコード目次

第1章　プログラミング言語の検証

現在の OS やアプリケーションの検証では、実装と別に検証用の言語で記述された実装と証明を持つのが一般的である。kernel 検証 [?],[?] の例では C で記述された Kernel に対して、検証用の別の言語で書かれた等価な kernel を用いて OS の検証を行っている。また、別のアプローチとしては ATS2[?] や Rust[?] などの低レベル記述向けの関数型言語を実装に用いる手法が存在している。

証明支援向けのプログラミング言語としては Agda[1]、 Coq[?] などが存在しているが、これらの言語自体は実行速度が期待できるものではない。

そこで、当研究室では検証と実装が同一の言語で行う Continuation based C[?] という言語を開発している。Continuation based C(CbC) では、処理の単位を CodeGear、 データの単位を DataGear としている。CodeGear は値を入力として受け取り出力を行う処理の単位であり、 CodeGear の出力を 次の GodeGear に接続してプログラミングを行う。CodeGear の接続処理はメタ計算として定義されており、実装や環境によって切り替えを行うことができる。このメタ計算部分で assertion などの検証を行うことで、 CodeGear の処理に手を加えることなく検証を行う。現段階では CbC 自体に証明を行うためのシステムが存在しないため、証明支援系言語である Agda を用いて等価な実装の検証を行っている。

第2章　Continuation based C

CbC について

2.1　Code Gear と Data Gear

概念

2.2　メタ計算

メタ計算の話

2.3　Meta Gear

Meta Gears の概念

第3章　Agda

Agda についての説明

3.1　Agda の Data Type

data とか record とか

3.2　Agda の関数

関数定義だったり型とかの話

3.3　Agda での検証

検証の話

3.4　定理証明とプログラミング検証

たぶん Curry-Howard isomorphism の話

第4章　Floyd-Hoare Logic

HoareLogic の話

4.1　Hoare Triple

HoareLogic の概念

4.2　Agda での Hoare Logic

人のコード解説になりそう

4.3　Hoare Logic の Soundness

Soundness 周り

4.4　Hoare Logic での証明

whileTestPrim.agda の解説

第5章　Continuation based C と Agda

対応のお話

5.1　DataGear の対応

agda での datagear

5.2　CodeGear

agda での codegear

5.3　Meta 部分の話

あんまりでなさそうだけどとりあえず

5.4　CbC 上での HoareLogic の実現

研究の主な部分

第6章　BinaryTree

CbC-Agda 上での Binary Tree

6.1　BinaryTree

BinaryTree 概要

6.2　BinaryTree の実現

BinaryTree の記述等

6.3　BinaryTree の検証時の問題点と改善

つまってたとこ (条件付きとかそのあたり)

6.4　BinaryTree 検証の HoareLogic を用いた解決

できたらいいなの話 (1/2 時点)

第7章　結論

まだ終わってないので最後に

7.1　今後の課題

いつかくる後輩の修論や卒論に備えて

謝辞

本研究の遂行、本論文の作成にあたり、御多忙にも関わらず終始懇切なる御指導と御教授を賜わりました河野真治准教授に心より感謝致します。そして、 共に研究を行い暖かな気遣いと励ましをもって支えてくれた並列信頼研究室の全てのメンバーに感謝致します。最後に、 有意義な時間を共に過ごした理工学研究科情報工学専攻の学友、並びに物心両面で支えてくれた家族に深く感謝致します。

2018 年 3 月
伊波立樹

参考文献

[1] Ulf Norell. Dependently typed programming in agda. In *Proceedings of the 4th International Workshop on Types in Language Design and Implementation*, TLDI '09, pp. 1–2, New York, NY, USA, 2009. ACM.

[2] Maged M. Michael and Michael L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, pp. 267–275, New York, NY, USA, 1996. ACM.

[3] Haogang Chen, Daniel Ziegler, Tej Chajed, Adam Chlipala, M. Frans Kaashoek, and Nickolai Zeldovich. Using crash hoare logic for certifying the fscq file system. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pp. 18–37, New York, NY, USA, 2015. ACM.

[4] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pp. 207–220, New York, NY, USA, 2009. ACM.

[5] Helgi Sigurbjarnarson, James Bornholt, Emina Torlak, and Xi Wang. Push-button verification of file systems via crash refinement. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pp. 1–16, Berkeley, CA, USA, 2016. USENIX Association.

[6] Jean Yang and Chris Hawblitzel. Safe to the last instruction: Automated verification of a type-safe operating system. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pp. 99–110, New York, NY, USA, 2010. ACM.

[7] 比嘉健太, 河野真治. Verification method of programs using continuation based c. 情報処理学会論文誌プログラミング（PRO）, Vol. 10, No. 2, pp. 5–5, feb 2017.

[8] J. Meyerson. The go programming language. *IEEE Software*, Vol. 31, No. 5, pp. 104–104, Sept 2014.

[9] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, Vol. 93, No. 1, pp. 55–92, July 1991.

[10] 大城信康, 河野真治. Continuation based c の gcc4.6 上の実装について. 第 53 回プログラミング・シンポジウム予稿集, 第 2012 巻, pp. 69–78, jan 2012.

[11] 宮城光希, 河野真治. Code gear と data gear を持つ gears os の設計. 第 59 回プログラミング・シンポジウム予稿集, 第 2018 巻, pp. 197–206, jan 2018.

[12] Chris Lattner and Vikram Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

[13] Tokumori Kaito and Kono Shinji. Implementing continuation based language in llvm and clang. *LOLA 2015, Kyoto*, July 2015.

[14] 小久保翔平, 伊波立樹, 河野真治. Monad に基づくメタ計算を基本とする gears os の設計. 第 133 回情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS), May 2015.

[15] 東恩納琢偉, 伊波立樹, 河野真治. Gears os における並列処理. 第 140 回情報処理学会システムソフトウェアとオペレーティング・システム研究会 (OS), May 2017.

[16] Openmp: Simple, portable, scalable smp programming. `http://www.openmp.org,`. Accessed: 2018/02/05(Mon).

[17] Cuda zone — nvidia developer. `https://developer.nvidia.com/cuda-zone`. Accessed: 2018/02/05(Mon).

[18] 徳森海斗. Llvm clang 上の continuation based c コンパイラ の改良. Master's thesis, 琉球大学 大学院理工学研究科 情報工学専攻, 2016.

[19] 小久保翔平. Code segment と data segment を持つ gears os の 設計. Master's thesis, 琉球大学 大学院理工学研究科 情報工学専攻, 2016.