修士 (工学) 学位論文
Master's Thesis of Engineering

修士論文のテンプレート

2020 年 3 月
March 2020

清水 隆博
Takahiro Shimizu

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 和田 知久
Supervisor: Prof. Tomohisa Wada

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)　　玉城 史朗

印

(副　査)　　遠藤 聡志

印

(副　査)　　名嘉村 盛和

印

(副　査)　　河野　真治

要旨

ここに要旨を書く

Abstract

hogefuga

発表履歴

- 宮城 光希, 桃原 優, 河野真治. GearsOS のモジュール化と並列 API. 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, 2018
- 桃原 優, 東恩納琢偉, 河野真治. GearsOS の Paging と Segmentation・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, 2019

目 次

図 目 次

表 目 次

第1章　継続を中心としたプログラミングスタイル

コンピューター上では様々なアプリケーションが常時動作している。動作しているアプリケーションは信頼性が保証されていてほしい。信頼性の保証には、 実行してほしい一連の挙動をまとめた仕様と、 それを満たしているかどうかの確認である検証が必要となる。アプリケーション開発では検証に関数や一連の動作をテストを行う方法や、デバッグを通して信頼性を保証する手法が広く使われている。

アプリケーションは通常特定のプログラミング言語で実装されている。このプログラミング言語自身の信頼性は高く保証される必要がある。また、実際にアプリケーションを動作させる OS も高い信頼性が保証される必要がある。OS は CPU やメモリなどの資源管理と、 ユーザーにシステムコールなどの API を提供することで抽象化を行っている。

OS の信頼性の保証もテストコードを用いて証明することも可能ではあるが、 アプリケーションと比較すると OS のコード量、 処理の量は膨大である。また OS は CPU 制御やメモリ制御、 並列・並行処理などを多用する。テストコードを用いて処理を検証する場合、テストコードとして特定の状況を作成する必要がある。実際に OS が動作する中でバグやエラーを発生する条件を、 並列処理の状況などを踏まえてテストコードで表現するのは困難である。非決定的な処理を持つ OS の信頼性を保証するには、 テストコード以外の手法を用いる必要がある。

テストコード以外の方法として、 形式手法的と呼ばれるアプローチがある。形式手法の具体的な検証方法の中で、 証明を用いる方法とモデル検査を用いる方法がある。証明を用いる方法では Agda や Coq などの定理証明支援系を利用し、 数式的にアルゴリズムを記述する。Curry-Howard 同型対応則により、型と命題が、 プログラムと証明が対応する。

第2章　GearsOSのトランコンパイラ

2.1　GearsCbCの雛形生成

Interfaceとそれを実装するImplの型が決定すると、最低限満たすべきCodeGearのAPIは一意に決定する。ここで満たすべきCodeGearは、Interfaceで定義したCodeGearと、Impl側で定義した privateなCodeGearとなる。例えばStack Interfaceの実装を考えると、各Implで`pop`, push, shift, isEmptyなどを実装する必要がある。

従来はプログラマが手作業でヘッダーファイルの定義を参照しながら`.cbc`ファイルを作成していた。手作業での実装のため、 コンパイル時に次のような問題点が多発した。

- CodeGearの入力のフォーマットの不一致
- Interfaceの実装のCodeGearの命名規則の不一致
- 実装を忘れているCodeGearの発生

特にGearsOSの場合はPerlスクリプトによって純粋なCbCに一度変換されてからコンパイルが行われる。実装の状況とトランスコンパイラの組み合わせによっては、 CbCコンパイラレベルでコンパイルエラーを発生させないケースがある。この場合は実際に動作させながら、gdb, lldbなどのCデバッガを用いてデバッグをする必要がある。またCbCコンパイラレベルで検知できても、すでに変換されたコード側でエラーが出てしまうので、トランスコンパイラの挙動をトレースしながらデバッグをする必要がある。Interfaceの実装が不十分であることのエラーは、 GearsOSレベル、最低でもCbCコンパイラのレベルで完全に検知したい。

Interfaceを機能として所持している言語の場合、これらはコンパイルレベルか実行時レベルで検知される。例えばJavaの場合はInterfaceを満たしていない場合はコンパイルエラーになる。

InterfaceのAPIを完全に実装するのを促す仕組みとして、Interfceの定義からエディタやツールが満たすべき関数と引数の組を自動生成するツールがある。golangの場合は主に`josharian/impl`[1]が使われている。これはインストールすると`impl`コマンドが使用可能になり、 実装したいInterfaceとImplの型を与えると雛形が生成される。主要なエ

ディタである vscode の golang の公式パッケージである `vscode-go`[2] でも導入されており、 vscode から呼び出すことが可能である。

Interface では入力の引数が Impl と揃っている必要があるが、 第一引数は実装自身のインスタンスがくる制約となっている。そのため Interface でデフォルト型 `Impl` を各実装の型に置換することで自動生成が可能となる。

第3章　まとめ

3.1　総括

3.2　今後の課題

3.2.1　hogehoge

謝辞

ホゲ様，フガ様ありがとうございます

参考文献

[1] josharian. josharian/impl.

[2] golang. golang/vscode-go.

[3] Kaito TOKKMORI and Shinji KONO. Implementing continuation based language in llvm and clang. *LOLA 2015*, July 2015.

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

[5] 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.

[6] 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.

[7] 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.

付 録A　研究会業績

A-1　研究会発表資料