２．取り組む研究について

（１）研究課題名*

型推論をそなえた言語による Unix ライク OS の研究開発と実証実験

（２）研究の概要

型推論を持つ言語による安全な設計はアプリケーションのみに適用できるものではなく Unix ライク OS の kernel 本体にも適用することができ、さらには既存のどのソフトウェア領域にも適用できることを実証する。

（３）研究計画等

【研究の背景】

Internet of Things (IoT) [1] の世界がやってこようとしています。日本語では「モノのインターネット」とも呼ばれ、なんの変哲のない物体(バスケットボール, コップ, ボールペン, 楽器, 車のワイパーなど)が直接もしくは間接的にインターネットに接続され、様々なサービスが構築されようとしています。このような IoT の世界はインターネットが出現した時のように、全世界の人々にこれまで体験したことがないサービスを提供するでしょう。例えば、「全世界の車のワイパーが、現在この瞬間にどのぐらいの強さで動いているのか集計できれば、センサーを全世界に配置するコストなしに全世界の現在の雨の強さを測定できる。するとより正確な天気予報ができるはずだ。」と考える人もいます。これは単なる例の 1 つにすぎず、より多くの応用が IoT の世界には手つかずのまま眠っています。IT 文明の輝かしい未来と言えるでしょう。

一方で IoT の基盤となるインターネットの信頼性については暗雲がたちこめています。2014 年に OpenSSL というセキュリティの基盤ソフトウェアに見つかった Heartbleed バグ [2] は記憶に新しいでしょう。インターネットに接続されているコンピュータがこの不具合を放置していると、OpenSSL ソフトウェアを経由した通信によって第三者にコンピュータ内のメモリを閲覧される危険性があります。この「コンピュータ」とはサーバ機器だけにとどまらず、スマートフォンなどの機器であったとしても OpenSSL を搭載していれば脅威となります。OpenSSL はオープンソースプロジェクトであり、ボランティアによって開発されています。その開発言語には C 言語が使われています。先の IoT デバイスにおいても暗号化のようなセキュリティの機能を搭載する際には OpenSSL のようなオープンソース製品を流用していたかもしれません。当然 2014 年の現在ではまだ IoT デバイスは普及しておらず、その点では Heartbleed バグの影響は小さな範囲で済んだとも言えます。

しかも、現在の製品開発にはボランティアで開発されているオープンソース製品の流用が必要不可欠です 。Android のようなスマートフォンはオープンソース製品群の寄せ集めです。インターネット上にあるサーバの多くは Linux デストリビューションを流用しています。 ARM のような大企業もオープンソースで自社の OS(mbed-rtos [3])を作り、外部からの貢献を受けつけています。ソフトウェア開発における 21 世紀は「オープンソースの世紀」と呼べるのです。それではなぜ現在オープンソースの採用が増えているのでしょうか？私は Unix ライク OS の持つ機能(ネットワーク, ファイルシステム, POSIX API)がどのドメインの製品にも要求されるようになったからだと考えます。例えば Android は携帯電話の OS として開発されましたが画面を持たないデバイスにも採用される動きがあります。また NetBSD を利用したルータ製品 [5] も数多くあります。

これらオープンソース製品の不具合はもちろん意図して放置されているのではありません。通常ソフトウェア

に不具合はつきものです。メーカでの製品テストでは形式手法のような方法を使って製品設計者はその不具合を許容できる範囲にまで少なくしています。それではオープンソースの信頼性はどのようにして作りこまれるのでしょうか？「目玉の数さえ十分あれば、どんなバグも深刻ではない [4]」というのはオープンソース製品である Linux kernel を作っているリーナス・トーバルズの発言であり、その真意はオープンソース製品を使う人が多けれ多いほどバグはすぐ見つかるというものです。つまり全世界的な人海戦術によってオープンソース製品の信頼性は担保されています。このような人力によって作られた信頼性に、現在の社会は完全に依存しているのです。その信頼性に対する信仰が裏切られた顕著な例が先の Heartbleed バグでしょう。これからインターネットに接続される多品種の IoT デバイスを作らなければならないにもかかわらず、その信頼性の担保を人力に頼るというのは科学的なのでしょうか？

私はこの問題を根絶するために Metasepi プロジェクト [12] を立ち上げ、以下を目的に活動をしています。

- C 言語より安全な型推論と強い型を持つ言語による OS の設計手法の探索
- 上記の OS をオープンソースとして開発し、全世界に公開する
- OS の内部部品を強い型を使って設計することで組み合わせ可能にする

これら Metasepi が生み出すコア技術は IoT のような新しい産業の足場として、日本の製造業はもちろん全世界に貢献すると信じています。

【これまでの研究成果】

Metasepi の開発はまだ本格的に開始されて 1 年半程度ですが、いくつかの成果が出ています。またこれらの成果は去年の研究計画とほぼ一致しています。去年描いた 5 年の研究計画を、わずか 1 年で達成したことになります。

A. Ajhc Haskell コンパイラ [6]

Jhc Haskell コンパイラ [7] に Metasepi プロジェクト独自の改造である Context Local Heap (CLHs)を施すことで、再入可能なバイナリを生成することができるようになり、世界ではじめてプリエンプション・スケジューラ [8] を持つ Unix ライク kernel を型推論を持つ言語である Haskell 言語 [9] で設計できるようになりました。この成果について現在論文を作成しています。

B. Ajhc でマイコン上ネットワークプログラミング

デモ動画: https://www.youtube.com/watch?v=C9JsJXWyajQ

RAM サイズが 32kB と非常に小さいメモリのデバイス上で Haskell プログラミングが可能なことを世界ではじめて実証しました。これは先の Ajhc コンパイラの CLHs の応用例の 1 つです。この成果は「第 55 回プログラミング・シンポジウム」[10] にて公開/論文化済みです。

C. Android アプリケーションを Ajhc で設計

デモ動画: https://www.youtube.com/watch?v=n6cepTfnFoo

世界ではじめて Android マーケットに Haskell で設計されたアプリケーションを登録しました。これは先の Ajhc コンパイラの持つ CLHs の応用例の 1 つです。

D. NetBSD kernel サウンドドライバの一部を Ajhc で再設計

デモ動画: https://www.youtube.com/watch?v=XEYcR5RG5cA

世界ではじめて実際に使われている Unix ライク kernel である NetBSD の割り込みハンドラを、CLHs を使うことで Haskell 言語で設計することに成功しました。また、この Haskell 言語による設計は時間的にも空間的に

も元の C 言語の設計と同等の効率であることをベンチマークで示しました。この成果は現在論文化中です。

【研究目的】

C 言語で記述されたソフトウェアを型推論をそなえた言語で少しずつ設計置換することが OS のような低レベルプログラミングでも可能であることを実証します。さらにこの設計置換手法がコンピューターアーキティクチャの全てのドメインに適用可能であることを実証します。

【研究計画と方法】

これまでは C 言語で設計された Unix ライク kernel である NetBSD OS を Ajhc コンパイラと Haskell 言語を使って設計置換してきました。C 言語を型の強い言語を使って置換設計することが可能であることを実証できました。しかし Ajhc にはいくつか OS 開発に向かない解決困難な課題が見つかっており、特に Haskell 言語の場合には C 言語表現とあまりにも乖離しているために Ajhc の吐き出すプログラムのデバッグは比較的困難でした。kernel 開発ではソフトウェアのデバッグだけではなく、ハードウェアをソフトウェアからデバッグすることも稀ではないため、デバッグ効率はアプリケーションよりも重要です。

そこで今後の研究開発では Haskell 言語の代わりに ATS 言語 [11] を使って NetBSD を設計置換します。この ATS は依存型と線形型が使える処理系で、Ajhc と同じく C 言語を経由して実行バイナリにコンパイルされます Ajhc と比較すると、線形型によって GC が不要になることと、コンパイル前の ATS 言語表現と C 言語表現があまり変わらないことが利点です。当然 C 言語表現と近いために ATS で記述された設計は Ajhc のそれよりも C 言語でのデバッグ手法を適用しやすいと考えられます。また、依存型を使うことができるため ATS 言語は Haskell 言語よりさらに安全な設計をできると言えます。しかし ATS 言語は世界的に見ても有名な言語ではありません。ATS はその基本設計の論文 [13] が公開されてからまだ 10 年しか経っていない若い言語です 。Metasepi OS を ATS で設計するにしても設計者の人数を確保しないことには、開発者コミュニティを形成できません。つまり ATS には Haskell のようなエコシステムがまだ無いのです。

そこで、本研究を大きく 2 つの柱に分けて計画します。1 つ目は ATS 言語の環境整備とその利用促進です。ATS は OS 開発に向いている言語と私は考えていますが、不足している機能はまだ多く存在します。

- Linux 上でしか ATS コンパイラを利用できません。特に Windows や Mac OS X 上で利用可能にすることで多数の開発者を獲得できると考えます。Mac OS X 対応には既に着手しています。
- Haskell のようなパッケージシステムが ATS にはありません。Haskell コミュニティは Cabal [14] というパッケージシステムができてから劇的な成長をとげました。そこで OCaml 言語の OPAM [15] というパッケージシステムを真似て ATS のパッケージシステムを設計します。
- ATS では線形型を使うことで GC が不要になりますが、線形型を使った設計には工数がかかることで知られています。そこで Ajhc の持つ CLHs の技術を ATS 言語に転用します。
- 一部の開発者はエディタではなく IDE を使って開発を行ないます。しかし ATS にはまだ IDE がありません。そこでオープンソースで公開されている C++言語の IDE である Qt Creator [16] をベースにして ATS 言語の IDE を設計します。これは Metasepi OS の開発にも役立ちます。
- 新しい文法を ATS コミュニティに提案します。現在の ATS 言語は ML 文法をベースにしています。しかし ML 文法は既存の C 言語設計者に馴染みがなく、彼らに ATS を普及させる障害となっています。Rust 言語 [22] のような C 言語と見た目が似た文法を ATS コミュニティに提案し、設計します。

- 日本において ATS 言語の普及を推進します。この目的のために私は Japan ATS User Group [19] を立ち上げ、ATS 関連のドキュメントを翻訳/公開しています。先の Japan ATS User Group ホームページはすでに ATS 言語のホームページからリンク [20] されています。

2 つ目は ATS 言語を使った Metasepi OS 本体の開発です。いくつかの段階にそって開発を行ないます。

- Arduino [17] や ARM マイコン [18] のような Unix ライク OS が動作しないが、IoT デバイスに組み込まれる可能性の高い小規模マイコンの上のプログラムを ATS 言語で設計します。マイコン上のプログラムは Unix ライク OS のような大規模なプログラムよりもデバッグしやすいため、この段階で ATS を使った OS 開発に必要な設計手法を洗い出します。また、近年はマイコンによるホビー開発が注目を集めており [21]、ATS 言語に対する宣伝効果も狙います。
- Android OS 上で ATS 言語で設計したアプリケーションを動作させます。これも Android 市場に対して ATS 言語をアピールするのが狙いです。
- NetBSD の仮想ファイルシステムを ATS 言語で再設計します。近年ではファイルシステムは複雑な構造になることが多く、kernel の中でも特に複雑なロジックを内包しています。この領域を ATS の依存型で効果的に安全に再設計できることを実証します。
- NetBSD の仮想メモリを ATS 言語で再設計します。仮想メモリは Unix ライク kernel の心臓部です 。NetBSD でもこの部分は活発に開発されており、またこの部分での不具合は kernel 全体の信頼性をいとも簡単に低下させます。仮想メモリの設計における不変条件を ATS の依存型でコンパイル時検査にすることができることを実証します。
- Metasepi kernel を動作させる具体的なハードウェアを選定し、そのハードウェアで動作する NetBSD kernel の 80%を ATS 言語で再設計します。現時点では世界中から開発者を獲得できるように安価なハードウェアである Raspberry Pi [23] を検討しています。C 言語と ATS 言語の設計比率はコンパイルされたバイナリサイズで測定します。
- Linux kernel のファイルシステムの内 1 つを ATS 言語で再設計した後、ATS 言語で再設計された NetBSD kernel に移植します。これは Unix ライク OS の部品化のための第一歩で、NetBSD と Linux の設計を ATS 言語の世界で安全に混ぜることができることを実証します。
- メモリ管理ユニット(MMU)のないハードウェア上で ATS 言語化された NetBSD kernel を動作させます。IoT デバイスは非常に安価なハードウェアで実現しなければならないため、MMU のない安いハードウェア上で Unix ライク OS の機能実現できればより良いと考えられます。uClinux [24] のような実装が既にありますが、Metasepi ではより進んで、仮想メモリを部品として捉えようとしています。TCP/IP やファイルシステムのように Unix ライク kernel では必須の部品で取り外し不可能であった部品を取捨選択できるようにします。この部品化によって安価な IoT デバイスを Unix ライク kernel を使って設計可能になります。

【研究場所】

日本国

【参考文献】

1. Ashton, Kevin (22 June 2009). "That 'Internet of Things' Thing, in the real world things

matter more than ideas". RFID Journal. http://www.rfidjournal.com/articles/view?4986
2. "Cyberoam Security Advisory - Heartbleed Vulnerability in OpenSSL". April 11, 2014. http://kb.cyberoam.com/default.asp?id=2909&Lang=1
3. mbed-rtos http://mbed.org/
4. 『伽藍とバザール』著：エリック・スティーブン・レイモンド／山形浩生 （訳・解説）USP 出版 2010 年 ISBN 978-4904807026
5. Products based on NetBSD http://www.netbsd.org/gallery/products.html
6. Ajhc - Haskell everywhere http://ajhc.metasepi.org/
7. Jhc Haskell Compiler http://repetae.net/computer/jhc/
8. Operating Systems: Design and Implementation, (共著) ISBN ISBN 0-13-142938-8
9. The Haskell Programming Language http://www.haskell.org/
10. 「第 55 回プログラミング・シンポジウム」 http://www.ipsj.or.jp/prosym/55/55CFP.html
11. The ATS Programming Language http://www.ats-lang.org/
12. Metasepi Project http://metasepi.org/
13. Applied Type System (extended abstract) http://www.ats-lang.org/PAPER/ATS-types03.pdf
14. The Haskell Cabal http://www.haskell.org/cabal/
15. OPAM http://opam.ocamlpro.com/
16. Qt Creator http://qt-project.org/wiki/category:tools::qtcreator
17. Arduino http://arduino.cc/
18. Cortex-M Series http://www.arm.com/products/processors/cortex-m/index.php
19. JATS-UG - Japan ATS User Group http://jats-ug.metasepi.org/
20. The ATS Programming Language - Community http://www.ats-lang.org/COMMUNITY/
21. Maker Faire | The Greatest Show and Tell on Earth http://makerfaire.com/
22. The Rust Programming Language http://www.rust-lang.org/
23. Raspberry Pi http://www.raspberrypi.org/
24. uClinux™ -- Embedded Linux Microcontroller Project http://www.uclinux.org/

３．研究業績について

【論文】

1. 「強い型による OS の開発手法の提案」岡部究 水野洋樹 瀬川秀一, 情報処理学会第 55 回プログラミング・シンポジウム, 査読有り, 2014 年 1 月

【著書】

2. 「僕のカノジョはスナッチャー」岡部究, λカ娘 6 巻, 査読無し, 17-47 頁, 2013 年 12 月
3. 「めたせぴ☆ふぁうんでーしょん」岡部究, λカ娘 5 巻, 査読無し, 3-44 頁, 2013 年 8 月

【講演】

4. 「Metasepi team meeting #13: NetBSD driver using Haskell」岡部究, オープンソースカンファレンス 2014 Tokyo/Spring, 2014 年 3 月
5. 「Functional MCU programming」岡部究, mbed 祭り 2014@冬の横浜, 2014 年 1 月
6. 「How to rewrite the OS using C by strong type」岡部究, 第 11 回クリティカルソフトウェアワークショップ http://stage.tksc.jaxa.jp/jedi/event/20140115.html, 査読有り, 2014 年 1 月
7. 「強い型による OS の開発手法の提案(ポスター発表)」岡部究, 情報処理学会 第 55 回プログラミング・シンポジウム http://www.ipsj.or.jp/prosym/55/55CFA.html, 2014 年 1 月
8. 「Functional MCU programming #0: Development environment」岡部究, mbed ではじめる関数型マイコンプログラミング講習会β, 2014 年 1 月
9. 「Metasepi team meeting #8: Haskell apps on Android NDK」岡部究, オープンソースカンファレンス 2013 Fukuoka, 2013 年 11 月
10. 「Metasepi team meeting #7: Snatch application on tiny OS」岡部究, KOF2013：関西オープンフォーラム 2013, 2013 年 11 月
11. 「Metasepi team meeting #6: "Snatch-driven development"」岡部究, オープンソースカンファレンス 2013 Tokyo/Fall, 2013 年 10 月
12. 「Metasepi team meeting: Ajhc Project Overview」岡部究, オープンソースカンファレンス 2013 Hiroshima, 2013 年 10 月
13. 「mbed ではじめる組み込み Haskell プログラミング」岡部究, mbed 祭り in Sapporo, 2013 年 9 月
14. 「組込向け Haskell コンパイラ Ajhc / mbed マイコンどうでしょう編」岡部究, オープンソースカンファレンス 2013 Hokkaido, 2013 年 9 月
15. 「組込向け Haskell コンパイラ Ajhc / POSIX 依存から脱出しよう編」岡部究, オープンソースカンファレンス 2013 Kansai@Kyoto, 2013 年 8 月

【オープンソース活動】

16. Japan ATS User Group http://jats-ug.metasepi.org/, 2013 年 12 月～
17. Ajhc Haskell コンパイラの開発 http://ajhc.metasepi.org/, 2013 年 3 月～
18. Metasepi OS の研究開発 http://metasepi.org/, 2012 年 8 月～
19. プレゼンテーションツール Carettah の開発 http://carettah.masterq.net/, 2011 年 7 月～
20. NetBSD man ページ翻訳環境の整備 http://netbsdman.masterq.net/, 2011 年 4 月～
21. Debian Maintainer による複数の Debian パッケージの管理 http://udd.debian.org/dmd.cgi?email=kiwamu%40debian.or.jp, 2009 年 9 月～

４．応募の動機と研究者としての抱負について

（１）プロジェクトへ応募した理由を記してください。

私は10年間大規模組み込み開発を経験してきました。その開発を通じて機能開発はもちろんですが、それよりも多くの不具合/市場障害を経験しました。私はこの日本の製造業におけるソフトウェアの不具合とそれによる開発者の不幸の根本原因がいったい何なのか、この数年間追い求めてきました。現在OSのような基盤ソフトウェアの品質維持は人海戦術に頼っていると言わざるをえません。そのノウハウは工学的な解決をなされているのでしょうか？またこの問題を解決することに研究者は積極的ではありません。たしかに彼らが作るOSは新規性にあふれています。しかし実用からはかけはなれています。彼らの作ったOSの上でFirefoxは動きません日常用途として使わないソフトウェアはいずれ朽ちる運命にあります。この研究状況を観察した結果、私は手をこまねいてただ待つだけではこの領域における研究開発は進まないと考えるようになりました。この領域における研究開発には以下の技能が必須です。

- プロジェクトモチベーションを理解するための大規模組み込み開発の実地経験
- UNIXのような近代的なOSに関する深い理解
- 論文ではなくソースコードを読解する体力
- 型推論をそなえた言語の利用経験

幸いにもOS開発の用途に使えそうなATS言語を開発しているボストン大学のHongwei Xi http://www.cs.bu.edu/~hwxi/ はMetasepiプロジェクトに強い関心を示してくれました。しかしその他にはMetasepiプロジェクトについて紹介をして共同研究できる人材はなかなか見つかりません。このMetasepiの開発に注力するために私は自営業者となりましたが、Metasepiの開発成果が事業化できるのはまだまだ10年以上先のことに思えます。Metasepiの目指すゴールはそれほどに巨大なのです。それまでは無償での研究開発を続ける他ありません。「数年の間MetasepiとATSに関する研究開発はほぼ独自に行ない、研究成果をわかりやすい形でアピールし研究の裾野を広げる必要がある」というのが私の現在の結論です。その数年間の純粋な研究開発のためにプロジェクトに応募しました。

（２）あなたにとっての理想の研究者像とはどのようなものですか？

目的のためには手段を選ばない研究者を理想としています。また学術的な通説(もしくは権威ある意見)についても検証が終わるまでは中立の立場を貫くようにしようと考えています。

例えば、GCを持つ言語でOSを設計する際には通常のUnixライクkernelのように「割り込みハンドラ」を使うのではなく、「割り込み要因をポーリング」した方が良いとされていました。しかしこれではUnixライクkernelをそのような言語では設計できないことになってしまいます。私はもう一度ゼロからこの問題を再検討し、Context Local Heap (CLHs)というGCの設計手法を編み出しました。またこのCLHsを使うことでUnixライクkernelのサウンドドライバをHaskell言語で記述できることをデモによって実証しました。これからのMetasepiとATSの開発においても学術的な通説を疑い、ありとあらゆる手段をつくすつもりです。

私は自分の専門領域の中にのみ注力することが目標達成のための最善手ではないと考えます。専門外にある分野を自分で理解し、自分の中にある解を発見しようと努めます。もちろん専門家の意見は尊重します。しかしプロジェクトにおける解は他者である専門家の中にあるのではなく、ステークホルダーである自分の中にしかないと硬く信じています。

（３）一人の研究者として、現在、世界が抱える諸課題の解決に向けてどのような貢献が出来るとお考えですか？

Linux kernel を代表とするフリー Unix ライク OS は世界を変えました。20 世紀に発明されたこの OS は世界のありとあらゆる機器を動かす基盤ソフトウェアになりました。20 世紀において携帯電話がフリー Unix で動く(Android, iPhone)と誰が想像しえたでしょうか？電話交換機のようなリアルタイムが要求されなおかつミッションクリティカルな機器が Linux によって制御され、旧来の μITRON による設計がお払い箱になると誰が想像しえたでしょうか？さらには現代においてサーバ OS の代名詞は Linux です。しかも 2014 年の現在もその適用領域は拡大しています。

しかしこのフリー Unix ライク OS の信頼性の担保は 21 世紀の現代でも難題であり、「オープンソース開発」と名前のつけられた人海戦術に頼っています。その問題の多くはソフトウェアの実行時エラーを人力でデバッグしていることに起因すると考えます。確かにこの「オープンソース開発」手法の上で開発された機能については人海戦術によって信頼性を確保することはできるかもしれません。しかしその主流開発の外での機能追加つまり製品開発への応用において信頼性の確保を行なうにはより多くの人海戦術のための人材が必要です。また当然、この人海戦術のために世界中の有能なソフトウェア開発者の頭脳が日々浪費されています。このような開発スタイルは文明的なのでしょうか？私達のソフトウェア工学は「品質を作るには群れるしかない」というこの地点で潰えてしまうのでしょうか？また、フリー Unix ライク OS は RTOS のような小規模な OS よりも機能と設計の規模が大きいため、人工衛星の制御のような領域にはまだ踏み込めていません。ハードリアルタイム制御についても同様で、これらの領域は VxWorks や μITRON のような機能の不足した OS に占有されています。

Metasepi はこれらの状況を打開します。強い型による実行時エラーの防止が Linux kernel のような基盤ソフトウェアの領域でも有効であると実証します。また、左記設計に必要な強い型を持つ言語のコンパイラを世界に提供します。このコンパイラを使って Metasepi ではない別のソフトウェアを第三者が作ることも可能になるでしょう。さらにこのコンパイラを用いることでほとんど全てのソフトウェア領域に強い型による安全な設計手法を適用できることを実証します。また、型による設計によって設計の見通しが良くなり高信頼性が要求される領域にもフリー Unix ライク OS のような機能規模が大きい基盤ソフトウェアが進出できるようになると考えます。Metasepi と ATS は、C 言語よりも安全な設計を提供する次世代の Linux と GCC を作る挑戦と言いかえることもできます。