様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　元 年　 ６ 月 １８ 日現在

機関番号：１３９０１
研究種目：基盤研究(C)（一般）
研究期間：2016～2018
課題番号：１６Ｋ００００５

研究課題名（和文）多機能型システムを持ったプログラミング言語のための型付き中間言語の設計

研究課題名（英文）Design of a typed intermediate language for a richly typed programming language

研究代表者
Ｊ　Ｇａｒｒｉｇｕｅ（Garrigue, Jacques）

名古屋大学・多元数理科学研究科・教授

研究者番号：８０２７３５３０

交付決定額（研究期間全体）：（直接経費）　　2,500,000円

研究成果の概要（和文）：関数型プログラミング言語OCamlの型検証器の堅牢性を高めるために様々な研究に取り組んだ。型検証器の理論的な基盤の整理と型検証器の見通しの改善を目指し、まずGADT（一般化代数的データ型）のパターンマッチングの網羅性という問題に理論的に答えを与え、OCamlに反映させた。他の開発者とともに、型検証器のモジュール性を高めた。型付中間言語が安全性に効果的になるための条件も検討した。また、プログラムの形式的検証の様々な研究を通じて、今後の中間言語の形式化を準備した。

研究成果の学術的意義や社会的意義
社会のあらゆる所でソフトウェアは我々の生活をサポートしている。娯楽や書類の作成から、飛行機の飛行制御システムや医療機器まで、応用は幅広いが、その全ての場面で安全性が求められる。型システムがプログラムの安全性の確保に効果的であり、型の表現力が高いほどの効果が現れる。しかし、型システムが複雑になると、型の整合性の確認を行う型検証器とその後の処理の正しさの担保が難しくなる。この研究はプログラミング言語処理系の型の正しさを保証することでソフトウェアの安全性の向上を目指す。

研究成果の概要（英文）：I did several researches to strengthen the robustness of the OCaml type checker. Aiming at improving the theoretical foundation and structure of the type checker, I first solved the question of the interaction between GADTs (generalized algebraic datatypes) and the exhaustiveness check of pattern matching, both theoretically and practically. I worked with other developers at making the type checker more modular. I studid the conditions for improving the safety of the language through a typed intermediate language. I also worked on several topics of program verification, which should help with the formalization of this typed intermediate language.

研究分野：プログラミング言語の基礎理論

キーワード：型システム　中間言語　型検証　プログラム検証

様　式　C−１９、Ｆ－１９－１、Ｚ－１９、CK−１９（共通）

１．研究開始当初の背景

(1)プログラミング言語の安全性の必要性
社会のあらゆる所でソフトウェアは我々の生活をサポートしている。娯楽や書類の作成から、飛行機の飛行制御システムや医療機器まで、応用は幅広いが、その全ての場面で安全性が求められる。飛行機や医療機器の場合、安全性の必要性ははっきりしており、ソフトウェア開発の規格が用意されているが、たとえ娯楽でも、ソフトウェアの安定性やプライバシーの保護が求められている。残念ながら、不安定などのバグのあるソフトウェアが世の中に溢れており、高度な安全性が求められる場面でも、ボーイング737MAXの例で分かるように、ソフトウェアの間違いが完全に排除なれているわけではない。多くの問題がテストで排除できるものの、テストで予測できなかった場合を扱うために、プログラムの論理自体を検証の対象とする手法が増えている。それらがプログラミング言語の性質と言語処理系、特にコンパイラの正しさに依存している。

(2)型システムの功績
プログラミング言語がソフトウェアの安定性向上を支援できる一つの側面が型システムである。変数や関数の型を定めることで、関数呼び出しにおける型の整合性があらかじめ確認でき、実行時の不整合を理論的に排除できる。この確認をコンパイル時に行うことで、テストもせずに早い段階で様々な問題を未然に防げる。特に、コンパイル処理後に実行時の型の確認を行わないという多く処理系において、型の不整合性が未定義な動作を起こさせることになり、非常に危険であるが、C言語などで書かれたソフトウェアではそういう問題が珍しくない。型安全性という性質をもったプログラミング言語において、そういう不整合性が起きないことが保証される。この性質のおかげで、表現力の高い型システムを持った関数型プログラミング言語であるOCamlでは、プログラムの様々な性質が細かい型の定義によって表現でき、実際にコンパイル時の型検証が成功したらプログラムがバグを含むことが珍しいと言われている。

(3)中間言語における型システムの必要性
プログラミング言語処理系がソフトウェアの安全性の向上を支援できるものの、その処理系自身もまたソフトウェアであり、バグがつきものである。OCamlの処理系はOCaml自身で書かれており、その恩恵を多大に受けているが、同時に多くの複雑なアルゴリズムを実装しているせいで、型システムで防ぎきれない論理的なバグも生じることがある。アルゴリズムを精査し、バグを潰していくことが当然可能で、実際に行われているが、それを理論的に保証することが難しい。それに代わる方法として、型検証を行った後の中間コードの型の整合性を確認するというやりかたも考えられる。もしも型検証やコンパイルのバグがあれば、そのせいで中間コードの型の整合性が崩れる可能性が非常に高く、未定義な動作が封じられる上、バグも確実に発見できる。この確認を効率よく行うために、中間言語における型システムが必要になる。

２．研究の目的

この研究の最終目的は型システムによってプログラミング言語処理系から得られたソフトウェアの安全性を確保することである。OCamlの様な表現力の高い型システムが前提となり、型付中間言語とその型検証器の構築がそれを得るための方法である。
具体的、以下の3つの目的を掲げた。
①OCamlのための型安全な中間言語を設計し、それを実装する。
この中間言語により、以下の2つの効果が期待できる。まず、型推論が終った後にプログラムがこの中間言語に変換されるが、変換後に整合性を確認することで型推論のバグが未然に防げる。また、この中間言語自身が変換の対象となり、この段階での変換の安全性も確認できる。
②基本的なプログラム変換を実装する。
変数代入など、基本的な変換を実装することで、この中間言語が実際に使えることを確認する。また、その変換が型安全性を壊さないことをテストできる。
③中間言語の安全性の形式的な証明を作る。
中間言語では表層言語の多くの構造がもっと単純なものに変換されるが、直接的に扱わなければならないものもある。

３．研究の方法

当初はHaskell(GHC)の先行研究に基づいてOCamlの型システムにあった型付中間言語を設計・実装した上で、プログラム変換への応用と形式的な証明を行う予定だった。
ただ、二つの事実によって一時的に方針を変えることになった。
まず、時期を同じくして、Pierrick　Coudercというフランスの博士の学生が同じテーマで研究を始めた。彼の指導教員のMichel MaunyがOCamlの関係者の1人である。Pierrickが精力的にこの問

題に取り組んだので、彼と連絡を取りながらその様子を見ていくことになった。
もうひとつはOCaml の開発形態の変化である。以前、OCaml は数人の研究者が蜜に連絡を取り、開発を行っていたが、2016年の暮れから、GitHubにコードを移し、多くの人が貢献できるようにした。結果的には、外部から提供されるコードが多くなり、特に型検証器は今までと違い、その変遷を知らない人が手を加えるようになった。
これを受けて、型付中間言語の導入以前に、理論的な基盤の整理と型検証器の見通しをよくすることが優先課題になり、特にその部分に関わるJane Street LLCのLeo WhiteとThomas Réfisと一緒に型検証器の修正と改良を行うことになった。

４．研究成果

この研究に関連するいくつかの分野で研究成果があった。

(1) 型付中間言語の設計
① 先行研究を広くサーベイし、OCamlの型システムと比較した結果、非常に表現力な高い言語が必要という結論に至った。それに対して形式的な証明を行うという目標と相まって、新しい中間言語を設計するより、定理証明支援系Coqの内部言語Gallinaを直接に利用した方が効果的という結論に至った。Gallinaは構文的にOCamlの一部に当たるが、型システムが極めて強い。引き続きその研究を続ける予定である。
② OCamlの型検証器の出力を確認するという役割に関して、Pierrick Courdercの検証器が完成している。TypedTreeというOCamlによって型注釈を加えられた構文木を入力とし、簡単なアルゴリズムでその整合性を確認する。代表者の成果ではないが、2018年10月のCouderc氏の学位審査会には報告審査委員として参加した。

(2) OCamlの型検証の理論と実装
①代表者がJacques Le Normandと一緒に実装して来たGADTという一般化代数的データ型が複雑な不変量をもったプログラムの安全性に寄与しているが、GADTの型付けとパターンマッチングの網羅性の検証の関係についてまだ完全な結果が得られていなかった。論文(3)では、この検証が理論的には決定不能であることを示し、制限された決定的なアルゴリズムを提案した。また、反駁節を構文に加えることで、パターンマッチングの不可能な場合を検証器に認識させる仕組みを提案した。新しいアルゴリズムと構文が現在配布されているOCamlに含まれている。
②岐阜大学の今井敬吾と一緒に、OCamlでの線形型やセッション型のエンコーディングを論文(1)で提唱した。型システムを拡張するのではなく、現在の型システムとモナドなどを使い、値が線形的に使われることが保証できる。これはOCaml自体を一つの中間言語とみなすことに当たる。このエンコーディングは等価再帰型というOCaml独自の機能を利用している。汎用性が高く、量子プログラミングへの応用も考えられる。
③他の開発者とともに今後のOCamlの発展に合ったデザインを検討し、型推論を現在の一体型のアルゴリズムから、制約の生成と解消を分けたものに変えることが望ましいという結論に至った。ただ、フロントエンド(パーザや前処理および型推論)とバックエンド(コードの生成・変換と最適化)が常に発展を続けており、現状で大きな変更を加えることが難しいので、対応策として、OCamlのモジュールシステムを活用し、型抽象によって各部の独立性を高めた上で部分的な変更を段階的に導入する方針を固めた。既に型のスコープの仕組みなどがその方針に沿って変更されている。
④理論と実装の狭間に生まれたいくつかのバグを修正した。特に再帰モジュールとファンクター(モジュール間の関数)や実在モジュールの抽象化から得られたシグネチャー(モジュールの型)の組み合わせにより健全でないモジュールが前から作れていたことが発覚し、その問題の理解を深めながら理論と実装を修正した。同様にクラスの全称化のときに起きていた健全性の問題を修正した。

(3) プログラムの検証
① 産業技術総合研究所の田中哲とReynald Affeldtとともに、CoqのGallinaで書かれたプログラムを効率の良い（C言語なんどの）コードに変換する方法を論文(2)で対案した。Gallinaのコードはの中で動作が証明できるのがメリットだが、証明の前提となっているコードの性質が変換後に必ずしも成り立たないので、モナド化という提案技法によって、必要な前提をCoqの中で証明できるようにした。
② Reynald Affeldtと学生のXuanrui Qiと田中一成とともに、簡潔データ構造における木構造を利用するアルゴリスムの実装と検証をCoqで行い、発表(2)で報告した。特に幅優先捜査に対する証明技法と複雑な木構造の不変量を小規模自己反映(ssreflect)を用いて証明する方法についての貢献になる。

５．主な発表論文等

〔雑誌論文〕（計 6 件）

(1) Keigo Imai, Jacques Garrigue, Lightweight linearly-typed programming with lenses and monads, Journal of Information Processing, 2019, 印刷中, 査読有

(2) Akira Tanaka, Reynald Affeldt, Jacques Garrigue, Safe Low-level Code Generation in Coq Using Monomorphization and Monadification, Journal of Information Processing, 26巻, 2018, 54–72, 査読有
DOI: 10.2197/ipsjjip.26.54

(3) Jacques Garrigue, Jacques Le Normand, GADTs and Exhaustiveness: Looking for the Impossible, Electronic Proceedings in Theoretical Computer Science, 241巻, 2017, 23–35, 査読有
DOI: 10.4204/EPTCS.241.2

(4) Akira Tanaka, Reynald Affeldt, Jacques Garrigue, Formal Verification of the rank Algorithm for Succinct Data Structures, Formal Methods and Software Engineering, Springer-Verlag LNCS, 10009巻, 2016, 243–260, 査読有
DOI: 10.1007/978-3-319-47846-3_16

〔学会発表〕（計 7 件）

(1) 今井 敬吾, Jacques Garrigue, レンズとモナドを用いた軽量な線形型付きプログラミング, 第121回情報処理学会プログラミング研究発表会, 2018/11/1

(2) Reynald Affeldt, Jacques Garrigue, Xuanrui Qi, and Kazunari Tanaka, Proving tree algorithms for succinct data structures, 第35回日本ソフトウェア科学会全国大会, 2018/08/30

(3) Akira Tanaka, Reynald Affeldt and Jacques Garrigue, Future Work Towards a Coq Library of Succinct Data Structures, 第20回プログラミングおよびプログラミング言語ワークショップ(PPL2018), 鳥取県米子市, 2018/03/06, ポスター

(4) Akira Tanaka, Reynald Affeldt and Jacques Garrigue, Safe Low-level Code Generation in Coq using Monomorphization and Monadification, 第114 回プログラミング研究発表会情報処理学会プログラミング研究会, 静岡市, 2017/06/09

(5) Akira Tanaka, Reynald Affeldt and Jacques Garrigue, Certified Mon{omorphizladific} ation of Gallina for Low-level Code Extraction, 第19 回プログラミングおよびプログラミング言語ワークショップ(PPL2017), 山梨県笛吹市, 2017/03/09, ポスター

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計 0 件）

〔その他〕

(1) OCamlポータル
https://ocaml.org/

(2) OCaml開発
https://github.com/ocaml/ocaml

６．研究組織

(2)研究協力者
研究協力者氏名： 田中一成
ローマ字氏名： Kazunari Tanaka