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

平成　３０　年　　６　月　１８　日現在

機関番号：　１２１０２
研究種目：　挑戦的萌芽研究
研究期間：　2015～2017
課題番号：　１５Ｋ１２００７

研究課題名（和文）マルチステージ証明記述言語の設計と開発

研究課題名（英文）Design and development of multi-stage languages for verification

研究代表者
亀山　幸義（KAMEYAMA, YUKIYOSHI）

筑波大学・システム情報系・教授

研究者番号：１０１９５０００

交付決定額（研究期間全体）：（直接経費）　　2,700,000円

研究成果の概要（和文）：本研究はメタプログラミングの手法・研究成果を、定理証明系における証明記述言語に適用することを目的とした研究である。メタプログラミングの中でも静的な型付けによる安全性の保証を与える手法に基づいて、破壊的変数や限定継続などのコントロールオペレータを用いてコード生成器の記述言語の表現力を高めるとともに、生成されるコードの記述言語にモジュールを追加した場合の拡張についても考察し、いずれも健全性が成立し一定の記述力をもつ型システムの設計に成功した。これを用いて非常に簡単な証明記述言語を設計し、副作用を含む証明生成器を記述する手法にについての知見を得ることができた。

研究成果の概要（英文）：This study aims at applying the method and results in metaprogrmaming to proof-description languages. Based on static assurance of generated code in terms of static typing for metaprograms, we enriched the expressiveness of code description languages by introducing mutable cells and delimited control. We have also extended the object language to ML-style modules. In both cases, we have successfully designed a type system which enjoy the soundness property, that implies that each generated code is scope safe and type safe. Using these results, we have designed a very simple proof-description language which allows side effects, and obtained a certain preliminary thoughts on proof generation using side effects.

研究分野：　プログラミング言語論、関数プログラミングと方システム、プログラム生成、定理証明

キーワード：　証明記述言語　プログラム生成　メタプログラミング　コントロールエフェクト　副作用　型安全性　モジュール

１．研究開始当初の背景

プログラムを生成するプログラム、すなわちメタプログラムについての研究が活発に行われている。メタプログラムの信頼性を向上させる鍵となる技術の１つが型システムに基づくスコープ安全性や型安全性の検証法である。一方で、Coq などの定理証明支援系においては、証明そのものを記述する言語のほかに証明を生成する方法（タクティック）を記述言語が必要であり、タクティックが証明項を生成するという見方に立てば、メタプログラムと非常に類似した設定である。タクティックは型システムなどを持たない言語で記述され信頼性に問題があることが多いが、タクティック記述言語対する信頼性・安全性の研究はまだ非常に少ない。

Coq システムにおけるタクティック記述言語は Ltac と呼ばれ、非常に柔軟に様々な証明生成アルゴリズムを記述できる一方で、非常に信頼性が低い。近年、関数型言語のモナドに着想を得た Mtac と呼ばれる新しいタクティック記述言語が提案され型付けなどの点で Ltac を改善することが示されたが、記述力がまだ不十分であり、また、メタプログラムの研究で議論されてきたスコープの問題と同様な問題が生じていた。

２．研究の目的

上記の背景をもとに、本研究では、メタプログラム（プログラム生成法）においてこれまで研究代表者らが培ってきた型システムに基づく信頼性や安全性を向上するための研究を、定理証明支援系の証明記述言語に適用することにより、定理証明支援系の証明生成手法の安全性を高めることを目的とした。また、これにより、従来のメタプログラムにはなかった新しい応用領域を開拓し、新たな展開を得ることを目的とした。

本研究の焦点は、純粋な関数型プログラミングの範囲におさまらない「非純粋」なプログラミングを必要とする証明生成器の扱いに置いた。非純粋なプログラムは、先行研究の Mtac では「モナド」とよばれる関数型プログラミング言語のデータ構造を利用して（それを若干変形して）表現していたものである。本研究では、非純粋なコード生成器を表すため「コントロールオペレータ」や「参照型の値（破壊的変数）」などを利用し、これらをもつメタプログラムの安全性をベースにして、非純粋な証明記述言語の安全性を考察することとした。また、研究の過程で、定理証明記述においてもモジュール機能が重要であることが判明し、モジュール機能を持つプログラムや証明の安全な生成についても研究をおこなうことを研究目的に追加した。

３．研究の方法

本研究では、主に以下の２つの方法で研究を遂行した。

３－１．コントロールエフェクトを持つメタプログラミング言語および証明記述言語の研究：

参照型のデータや限定継続などのコントロールオペレータを利用することにより、プログラムや証明を生成する言語の記述力は格段にあがり、効率的なコードを生成することができるようになる。たとえば、生成されるプログラムに同一の計算式が 2 回以上含まれている場合、これらを 1 回だけ計算して計算結果を各所で利用するようなプログラムにすると、生成されたプログラムの実行性能が向上することが期待できる。このようなプログラム変換をプログラム生成時に行うためには参照型か、あるいは、コントロールオペレータが必要である。これらの機能を持つコード生成言語を MetaML の拡張の形で設計し、さらに型安全性を厳密に保証する型システムを導入して、その安全性の証明を行った。

３－２．モジュール生成機能を持つメタプログラミング言語および証明記述言語の研究：

ここでのモジュールは ML スタイルのモジュールであり型と式の定義の並びをまとめたものである。ML スタイルのモジュールの特徴は、インタフェースと実装の分離であり、インタフェース側は抽象型をふくむことができるため、実装されたモジュールの型によらない一般的なインタフェースの記述が可能である。Coq システムのモジュールは ML モジュールに類似しており、モジュール機能を持つメタプログラミング言語を設計して型安全性の保証を行えば、モジュールを持つ証明記述言語の安全性の保証にもつながると考えた。

４．研究成果

４－１．コントロールエフェクトを持つメタプログラミング言語および証明記述言語の研究：

この研究では、MetaML を念頭においた言語に参照型を適切に導入し、型システムの健全性を保証することに成功した。当初は局所的なスコープを持つ参照型の値という、既存のプログラミング言語には存在しない機能に対する証明しか得られていなかったが、研究の進捗にともない、当初与えた方システムをそのままもちいることにより、無限の有効範囲を持つ（局所的でない）参照型の値に対

しても型システムの健全性が成立することを証明することができた。

さらに、研究対象を限定継続(delimited continuation)とよばれるコントロールオペレータにうつし、現芸継続を持つメタプログラミング言語に対して、適当な制限のもとで方システムの健全性が成立することを厳密に保証した。これにより、従来のメタプログラミング言語が対応できなかった複雑な let 挿入などが表現できるようになり、その技法を用いた効率的プログラム生成が可能となった。

これらの成果を証明記述言語の世界に輸入して、参照型や限定継続を持つ証明記述言語の基礎付けを行った。モナドの形で記述できるものについては Mtac などの先行研究の範囲内であるが、限定継続は通常のモナドでは記述できないため、本研究の成果は少なくとも理論的には先行研究でカバーできない部分に到達したと考えられる。現在のところ、この言語の実装は進行中であり、限定的な証明の記述のみが可能であるが、将来的には本格的な証明記述に利用できると期待できる。

４－２． モジュール生成機能を持つメタプログラミング言語と証明記述言語の研究：

ML スタイルのモジュールのコードを生成することができるメタプログラミングのための言語と型システムの設計を行った。このためのカギとなる観測は、モジュールがレコードとして表現できることである。これを利用して、モジュールのコードをコードからなるモジュールに対応付けることができ、これにより、人間にとってわかりやすいモジュール生成記述言語を得ることができた。

この言語を、モジュール生成機能を持たない通常の OCaml 言語へ変換する変換器を設計し、その型保存性を証明した。これにより、モジュール生成機能を持つメタプログラムが容易に記述できるとともに、それを容易に実行してコードを得ることができるようになった。

さらに、上記の成果を Coq のモジュールに対応付け、モジュールを含む証明を生成するための基礎的な考察をおこなった。Coq は依存型を持っているため、モジュールを、依存型を持つレコードとして表現することができ、より自然なモジュールを含む証明の生成が可能という点は確認できた。一方で、依存型をもつことの問題点として、モジュールの中の一部のコードだけを利用するコードの場合、自由変数が生じるのを防ぐために let 式を導入しなければならず、これが場合によってはコード・証明の爆発をひきおこすことがあるという問題点もわかった。この最後の問題点はメタプログラミングの世界でも同様に問題となることがわかっており、今後の研究で取り組むべき、新たな重要な研究課題であると考える。

５．主な発表論文等

〔雑誌論文〕（計 10 件）

1. Naoki Takashima, Hiroki Sakamoto and Yukiyoshi Kameyama, Generate and Offshore: Type-safe and Modular Code Generation for Low-Level Optimization, Workshop on Functional High-Performance Computing, 9 pages, 2015. DOI 10.1145/2808091.2808096 (査読有)
2. Ikuo Kobori, Yukiyoshi Kameyama and Oleg Kiselyov, Answer-Type Modification without Tears: Prompt-Passing Style Translation for Typed Delimited-Control Operators, Proceedings of the Workshop on Continuations, Electronic Proceedings in Theoretical Computer Science, Volume 212, pp. 36-52, 2015. DOI 10.4204/EPTCS.212.3 (査読有)
3. Yukiyoshi Kameyama, Oleg Kiselyov and Chung-chieh Shan, Combinators for Impure yet Hygienic Code Generation, Science of Computer Programming, Vol. 112, Part 2, pp. 120-144, Nov. 2015. DOI 10.1016/j.scico.2015.08.007 (査読有)
4. Kenichi Asai and Yukiyoshi Kameyama, Automatic Staging via Partial Evaluation Techniques, International Symposium on Symbolic Computation in Software Science, pp.1-13, 2016, http://www.easychair.org/publications/paper/262500 (査読有)
5. Jun Inoue, Oleg Kiselyov and Yukiyoshi Kameyama, Staging beyond terms: prospects and challenges, Proc. of the

ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, 2016, pp. 103-108, 2016, DOI 10.1145/2847538.2847548 (査読有)

6. Kenichi Suzuki, Oleg Kiselyov and Yukiyoshi Kameyama, Finally, safely-extensible and efficient language-integrated query, Proc. of the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp. 37-48, 2016, DOI 10.1145/2847538.2847542 (査読有)
7. Oleg Kiselyov, Yukiyoshi Kameyama and Yuto Sudo, Refined Environment Classifiers: Type- and Scope-Safe Code Generation with Mutable Cells, Proc. of Asian Symposium on Programming Languages and Systems, pp.271-291, 2016. DOI 10.1007/978-3-319-47958-3_15 (査読有)
8. 坂口和彦、亀山幸義, 定理証明器 Coq の効率的な有限ドメイン関数ライブラリ, 情報処理学会論文誌: プログラミング, Vol. 10, No. 1, pp. 14-28, Jan 2017 (査読有)
9. Junpei Oishi and Yukiyoshi Kameyama, Staging with control: type-safe multi-stage programming with control operators, Proc. of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, pp.29-40, 2017. DOI 10.1145/3136040.3136049 (査読有)
10. Takahisa Watanabe and Yukiyoshi Kameyama, Program generation for ML modules (short paper), Proc. of the ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp. 60-66, 2018. DOI 10.1145/3162072 (査読有)

〔学会発表〕(計１１件)

亀山幸義: 段階的計算と Proof Obligation, 11th Theorem Proving and Provers Meeting, 2015.9.16-9.17.

薄井千春、亀山幸義：ラムダ計算における shift/reset による call/cc の模倣、11th Theorem Proving and Provers Meeting, 2015.9.16-9.17.

坂口和彦、亀山幸義：Coq/SSReflect の extraction の改善、第 18 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、ポスター発表、2016.3.7-3.9.

大石純平、亀山幸義：多段階 let 挿入を行うコード生成言語の型システムの設計、日本ソフトウェ科学会第 33 回大会, 2016.9.6-9.9.

薄井千春、亀山幸義：ワンショット限定継続からコルーチンへの変換、情報処理学会研究会 112 回プログラミング研究会、2017.1.10-1.11.

坂口和彦、亀山幸義： 定理証明器 Coq の効率的な有限ドメイン関数、情報処理学会第 109 回プログラミング研究会、2016.6.9-6.10.

Yukiyoshi Kameyama: Type-safe Generation of High-Performance Code, NII Shonan Meeting on Putting Heterogeneous High-Performance Computing, 2016.11.17-11.20.

Yukiyoshi Kameyama: Type-safe Multi-Stage Programming with Control, IFIP WG2.11 on Program Generation, Koblenz Meeting (招待講演), 2017.7.17-7.20.

渡部恭久、亀山幸義： モジュールの生成が可能なマルチステージ言語の提案、第 34 回日本ソフトウェア科学会大会, 2017.9.19-9.21.

Yukiyoshi Kameyama: Modified Environment Classifiers: Type- and Scope-Safe Code Generation with Mutable Cells, 第 30 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、2018.4.5-4.7.

〔図書〕(計 1 件)

David Duke and Yukiyoshi Kameyama (editors), Proceedings of the 5th International Workshop on Functional High-Performance Computing, 2016, ACM. DOI 10.1145/2975991

〔産業財産権〕

○出願状況（計０件）

○取得状況（計０件）

〔その他〕
ホームページ等
http://www.cs.tsukuba.ac.jp/~kam/research-j.html

６．研究組織

(1)研究代表者

亀山　幸義（KAMEYAMA Yukiyoshi）
筑波大学・システム情報系・教授
研究者番号：10195000

(2)研究分担者

なし

(3)連携研究者

なし

(4)研究協力者

坂口　和彦（SAKAGUCHI Kazuhiko）
筑波大学・システム情報工学研究科・博士後期課程

渡部　恭久（WATANBE Takahisa）
筑波大学・システム情報工学研究科・博士前期課程

大石　純平（OISHI Jumpei）
筑波大学・システム情報工学研究科・博士前期課程

薄井　千春（USUI Chiharu）
筑波大学・システム情報工学研究科・博士前期課程

須藤　悠斗（SUDO Yuto）
筑波大学・システム情報工学研究科・博士前期課程