様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２８ 年　 ６ 月 １５ 日現在

機関番号：１２１０２
研究種目：基盤研究(B)（一般）
研究期間：2013～2015
課題番号：２５２８００２０

研究課題名（和文）信頼性の高いコード生成のためのプログラミング言語の実現

研究課題名（英文）Study on Highly Reliable Programming Languages for Code Generation

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）　　13,300,000円

研究成果の概要（和文）：コード生成法は、特定の環境やパラメータに応じた特化プログラムを生成することにより、プログラムの実行性能を高める手法である。本研究は、コード生成法の安全性と信頼性を高める手法についての理論体系の構築と言語実装を行った。主な成果は、1. さまざまな副作用を許すコード生成言語に対して、生成されるコードが安全であることを保証する型システムの設計および実装に成功した、2.データベース問合せ言語SQLの効率的コードを生成する型安全な体系を設計・実装した、3. 段階的計算の注釈を自動的に最適な位置に挿入する手法を提案した、などである。これらにより、安全で信頼できるコード生成プログラムの構築手法に貢献した。

研究成果の概要（英文）：Code generation is a leading approach to generate, for a given generic program, specialized code for individual environments, parameters, and architectures. In this research we have developed theories and programming languages for safe and highly reliable code generation, and have also implemented them. Our major results include: (1) We succeeded in designing and implementing a new type system in which one can use various side effects such as mutation and control operators as well as can write code generators, yet the system ensures well typedness and well scopedness statically. (2) We have designed and implemented a new language for generating efficient database queries in SQL that is type safe. (3) We have proposed an automatic technique for inserting staging annotations to programs that are guaranteed to be optimal. We believe that these results and many others of our research contributed to enhance the safety and reliability of program generation techniques.

研究分野：プログラム言語、プログラムの論理

キーワード：ディペンダブルコンピューティング　関数型プログラム言語　プログラム生成　プログラム検証　プログラム変換　プログラム特化　高性能計算　型システム

１．研究開始当初の背景

プログラム生成（コード生成）は、部分計算（プログラム特化）や実行時コード生成を包含する概念である。本研究は、安全で信頼性の高いプログラム生成法により効率の良いプログラムを生成するための理論構築、体系設計、言語実装、応用例の作成を行うことを目標としたものである。

プログラム生成のためのプログラム言語に関する先行研究では、Taha らが、純粋な関数型プログラム言語の範囲内でのコード生成に関して、静的な型システムを設計し、安全性を保証することに成功していた。これに基づくプログラム言語として、主流の関数型言語 OCaml の拡張である MetaOCaml が実用に供され、広く用いられていた。しかしながら、効率の良いプログラムを生成するためには、計算エフェクトを利用することが不可欠であるが、計算エフェクトを持つプログラム生成器の安全性は、非常に限定的な体系を除いて、保証されていなかった。このようなプログラム生成器では、型や変数束縛のエラーが起きやすく、静的にそれらのエラーが起きないことを保証する仕組みが強く望まれていた。さらに、コード生成のためのプログラム言語において、越段階埋め込み（CSP）機能や多相型など様々な先進的な言語機能との共存においても、安全性や信頼性を保証する仕組みが必要とされていた。

２．研究の目的

上記の背景のもと、本研究では、なるべく広い範囲の計算エフェクトを許し、効率良いプログラムを生成できるプログラム生成のためのプログラム言語体系を構築し、その安全性と信頼性を静的に保証する研究を行うことを目的とした。また、計算エフェクトに加えて、越段階埋め込みや多相型を取り込んだ体系への拡張についても検討し、安全性・信頼性を、できるだけ広い範囲で静的に保証するための理論の構築と言語設計を行うことを目的とした。

さらに、設計・構築した体系の有効性を実際に確認することも目的とした。具体的には、言語を実装して、高性能計算などの具体的なアプリケーション分野に適用し、高性能コードの生成を高い信頼性をもって行うことができることを実証することも目的とした。

３．研究の方法

本研究では、コード生成言語の安全性・信頼性に関するこれまでの研究代表者、分担者らの研究および先行研究にもとづき、純粋な関数型言語に基づくコード生成体系に、計算エフェクトを適切な形で追加し、洗練された型システムにより、安全性や信頼性を確保するという手法で体系の構築を行った。この体系をベースとして、任意の計算エフェクト、多相型、CSP などの機能を追加し、型システムを適切に拡張できるかを検討した。

このような理論研究におけるボトムアップのアプローチとは別に、コード生成法の主要な応用先のひとつである高性能計算のうち、線形代数計算、高速フーリエ変換、画像フィルタなど典型的なアプリケーションを取り上げ、本研究の体系で、ドメインやアーキテクチャに応じた効率良いコードが生成できるか実証するトップダウンのアプローチの研究も行った。

これら２つのアプローチは、相互に影響を与えあい、研究を進展させることができた。また、研究当初は予想していなかった様々な研究テーマを発掘することができたため、これらの研究もおこなった。

４．研究成果

3 年間の研究成果のうち、主要なものについて述べる。

（１）任意のモナディックな計算エフェクトを許しつつ、静的な型安全性を保証するコード生成ライブラリの作成（発表論文[9, 16]）。Haskell における計算エフェクトはモナドの形で表現されるが、本研究では、モナドとしてあらわされる全ての計算エフェクトに対応した安全なコード生成ライブラリを Haskell 上で実装し、種々の高性能コードの生成器を記述することに成功した。

従来研究では、安全性を失わずに許容することのできた計算エフェクトは非常に制限が大きく、特に生成されたコードの変数束縛のスコープを超える計算エフェクトがある場合の安全性の確保は大きな未解決問題であった。本研究は、従来研究の範囲を大きく超えた画期的な成果であると考えている。本研究の成果は Staged Haskell Library としてインターネットで公開しており、計算エフェクトとコード生成が共存し、安全性が確保された体系として、この分野の研究の標準の１つとなると考えている。

（２）型安全で拡張が容易な統合言語クエリの実現（発表論文[6, 11]）。関数型プログラム言語を使ってデータベース問合せ言語 SQL のコードを生成する統合言語クエリは、プログラミングを容易にするが、生成された SQL コードが非効率であることが問題であった。本研究は、効率的 SQL コード生成を可能とし

た Cheney らの先行研究を改善し、対象言語や効率化ルールの拡張に対して頑健な言語体系を構築した。この体系では、型安全性や変数スコープの安全性がメタ言語の型システムにより自動的に保証されるため、拡張が極めて容易であり、本格的・大規模な統合言語クエリの実現に向けた大きな一歩となった。

（3）低レベル最適化を高レベル言語で行うためのモジュラーなコード生成法（発表論文[7]）。CPU のベクトル命令や融合命令を活用した高性能コードの生成は、従来はコンパイラの仕事であり、コード生成器がそのような命令を利用した最適化を行うことはできなかった。例外は、Intrinsics であるが、それを用いたプログラムは柔軟性に乏しく、プログラムの保守や検証が容易でなかった。本研究はこれらを解決するため、CPU のベクトル命令等を直接高水準言語のデータタイプとその上の演算として表現するモジュールを容易し、低レベル最適化を直接、高レベル言語で記述しすつ、型安全性を失わないコード生成手法を提案した。この手法は Kiselyov の終タグレス法（tagless-final approach）に基づいており、モジュラーで型安全なコード生成器の記述を可能とするという大きな利点がある。

（4）次世代コード生成言語の提案（発表論文[5,10]）。従来のコード生成のためのプログラム言語は、もっぱら「項（式）」をコードとして生成しており、型、宣言、モジュールなどを生成することは、型安全性が十分保証されない一部の言語でしか実現できていなかった。本研究では、特に、OCaml スタイルのモジュールを生成する言語機能を MetaOCaml などの言語に追加することについて、その必要性、応用例、そして言語設計等を論じ、次世代コード生成言語の設計指針を与えた。

（5）コード生成器が必ず安全なコードを生成することを保証するための型システムの研究を行った（発表論文[14]）。cross-stage persistence (CSP) と呼ばれる，コード内にコード生成時に計算した結果を埋めこむための機構についての型システムを Tsukada-Igarashi の既存の体系を拡張して構築し，その型安全性を定理として示した Taha らによる，既存の CSP の形式化に比べて，より単純な意味論・型安全性の証明を与えることができた．

上記の結果は，単相型システムでしかも副作用のない純粋な言語に対するものであるが，CSP 機構は，ML の値多相・参照機構と素朴に組み合わせると安全性が損なわれることが知られている．この問題を解決するための新しい型システムを提案した．

（6） 信頼性の高いコード生成を行うためには、コード生成器そのものの信頼性も重要となる。そこで、自動でコード生成を行う offline の部分評価器について、その定式化を定理証明系 Agda を用いて行った（発表論文[13]）。Type preservation は、term の定義から明らかな形で定義するとともに、semantic preservation（部分評価の前後でプログラムの意味が変わらないこと）は、論理関係を用いて証明した。

また、自動でコード生成を行う部分評価と手動でコード生成部分を指示する段階的計算の間の関係を明らかにした（発表論文[4]）。これにより、段階的計算の柔軟なコード生成を損なうことなく、一部、部分評価で使われる束縛時解析の手法を取り入れることにより、自動でコード生成部分を指示できるようにした。さらに、ふたつの手法の関係を精査することで、段階的計算では明らかではなかった「最もよい注釈」を定義し、部分評価の束縛時解析を通して得られる注釈が最も良いものであることを示した（発表論文[1]）。

（7）高レベルプログラムの検証のための形式体系であるリファインメント型システム上の型推論問題を、多目的最適化問題として一般化したリファインメント型最適化問題を提案し、実際にそのような最適化問題を解くための、制約最適化に基づく新手法を開発した（発表論文[8]）。本研究ではさらに、天使的・悪魔的非決定性をともなうプログラムの検証を可能とするためのリファインメント型システムおよび型推論法の拡張も行った。これによって、高レベルプログラムの、実行が停止しない入力条件の推論や、最悪の計算ステップ数を引き起こす入力条件の推論といった新しい応用が可能となった。

５．主な発表論文等

〔雑誌論文〕（計 16 件）

[1] Kenichi Asai, Yukiyoshi Kameyama: Automatic Staging via Partial Evaluation Techniques, Proc. of the 7th International Symposium on Symbolic Computation in Software Science, EPiC Series in Computing, Vol. 39, pp. 1-13, 2016. 査読有

[2] 門脇香子、浅井健一： Agda による型推論器の拡張と改良、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

ップ論文集（オンライン）、15 ページ、2016. 査読有

[3] Akihiro Murase, Tachio Terauchi, Naoki Kobayashi, Ryosuke Sato, Hiroshi Unno: Temporal Verification of Higher-order Functional Programs, Proc. of the 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 57-68, 2016. 査読有
DOI 10.1145/2837614.2837667

[4] Kenichi Asai: Toward Introducing Binding-Time Analysis to MetaOCaml, Proc. of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp.97-102, 2016. 査読有
DOI 10.1145/2847538.2847547

[5] Jun Inoue, Oleg Kiselyov, Yukiyoshi Kameyama: Staging beyond Terms: Prospects and Challenges, Proc. of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp.103-108, 2016. 査読有
DOI 10.1145/2847538.2847548

[6] Kenichi Suzuki, Oleg Kiselyov, Yukiyoshi Kameyama: Finally, Safely-Extensible and Efficient Language-Integrated Query, Proc. of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp.37-48, 2016. 査読有
DOI 10.1145/2847538.2847542

[7] Naoki Takashima, Yukiyoshi Kameyama, Oleg Kiselyov: Generate and Offshore: Type-safe and Modular Code Generation for Low-Level Optimization, Proc. of the 4th ACM SIGPLAN Workshop on Functional and High-Performance Computing, pp.45-53, 2015. 査読有
DOI 10.1145/2808091.2808096

[8] Kodai Hashimoto, Hiroshi Unno: Refinement Type Inference via Horn Constraint Solving, Proc. of the 22nd International Static Analysis Symposium, LNCS 9291, pp. 199-216, 2015. 査読有
DOI 10.1007/978-3-662-48288-9_12

[9] Yukiyoshi Kameyama, Oleg Kiselyov, Chung-chieh Shan: Combinators for Impure yet Hygienic Code Generation, Science of Computer Programming, Vol. 112, pp. 120-144, 2015. 査読有
DOI 10.1016/j.scico.2015.08.007

[10] Jun Inoue, Oleg Kiselyov, Yukiyoshi Kameyama: The Next Stage of Staging, 第 17 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論文集（オンライン）、11 ページ、2015. 査読有

[11] 鈴木健一、亀山幸義、オレッグキセリョーフ: ついに SQL を組み立てる：拡張可能で安全な統合言語クエリ、第 17 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論文集（オンライン）、17 ページ、2015. 査読有

[12] 門脇香子、浅井健一: Agda による型推論器の定式化、第 17 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論文集（オンライン）、13 ページ、2015. 査読有

[13] Kenichi Asai, Luminous Fennell, Peter Thiemann, Yang Zhang: A Type Theoretic Specification of Partial Evaluation, Proc. of 2014 Symposium on Principles and Practice of Declarative Programming, pp.57-68, 2014. 査読有
DOI 10.1145/2643135.2643146

[14] Yuichiro Hanada, Atsushi Igarashi: On Cross-Stage Persistence in Multi-Stage Programming, Proc. of International Symposium on Functional and Logic Programming, Lecture Notes in Computer Science 8475, pp. 103-118, 2014. 査読有
DOI 10.1007/978-3-319-07151-0_7

[15] Noriko Hirota, Kenichi Asai: Formalizing a Correctness Property of a Type-Directed Partial Evaluator, Proc. of ACM SIGPLAN Workshop on Programming Language meets Program Verification, pp. 41-46, 2014. 査読有
DOI 10.1145/2541568.2541572

[16] Yukiyoshi Kameyama, Oleg Kiselyov, Chung-chieh Shan: Combinators for Impure yet Hygienic Code Generation, Proc. of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, pp.3-14, 2014. 査読有
DOI 10.1145/2543728.2543740

〔学会発表〕（計 8 件）

[1] 渡部恭久、亀山幸義：1ML のサブセット言語に対する型システムの構築、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、2016.3.7-9, ダイヤモンド瀬戸内マリンホテル、岡山県玉野市。

[2] 小林恵、五十嵐淳： 参照を備えた多段階計算のための多相的型システム、日本ソフトウェア科学会第 32 回大会、2015.9.8-11, 早稲田大学、東京都新宿区。

[3] オレッグキセリョーフ、亀山幸義：Prolog 再考： 急がないで推測、日本ソフトウェア科学会第 31 回大会、2014.9.8-10, 名古屋大学、愛知県名古屋市。

[4] 須藤悠斗、オレッグキセリョーフ、亀山幸義、コード生成のための自然演繹、日本ソフトウェア科学会第 31 回大会、2014.9.8-10, 名古屋大学、愛知県名古屋市。

[5] 鈴木健一、亀山幸義、オレッグキセリョーフ： 拡張可能で安全な統合言語クエリ（ポスター発表)、日本ソフトウェア科学会第 31 回大会、2014.9.8-10, 名古屋大学、愛知県名古屋市。

[6] 石井柚季、浅井健一： 型デバッガのログの解析とエラーメッセージの改良、第 16 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、2014.3.5-7, 阿蘇の司ビラパークホテル、熊本県阿蘇郡阿蘇町。

[7] 清水春樹、亀山幸義： 段階的計算における最適なステージ化プログラム生成の自動化、日本ソフトウェア科学会第 30 回大会、2013.9.11-13, 東京大学、東京都文京区。

[8] 花田裕一朗、五十嵐淳：多段階計算ラムダ |>のための越段階埋込、日本ソフトウェア科学会第 30 回大会、2013.9.11-13, 東京大学、東京都文京区。

〔その他〕 ホームページ等
http://logic.cs.tsukuba.ac.jp/~kam/programming/

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
浅井 健一 (ASAI, Kenichi)
お茶の水女子大学・人間文化創成科学研究科・准教授
研究者番号：10262156

五十嵐 淳 (IGARASHI, Atsushi)
京都大学・情報学研究科・教授
研究者番号：40323456

海野 広志 (UNNO, Hiroshi)
筑波大学・システム情報系・助教
研究者番号：80569575

(3)連携研究者

キセリョーフ,オレッグ(KISELYOV, Oleg)
東北大学・情報科学研究科・助教
研究者番号：50754602