論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Supporting multi-scope and multi-level compilation in a meta-tracing just-in-time compiler
著者(和文)	伊澤侑祐
Author(English)	Yusuke Izawa
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第12328号, 授与年月日:2023年3月26日, 学位の種別:課程博士, 審査員:増原 英彦,遠藤 敏夫,南出 靖彦,脇田 建,渡部 卓雄,千葉 滋
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第12328号, Conferred date:2023/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Category(English)	Doctoral Thesis
種別(和文)	審査の要旨
Type(English)	Exam Summary

（博士課程）

論文審査の要旨及び審査員

報告番号	甲第　　号		学位申請者氏名		伊澤 侑祐	
論文審査 審査員		氏　名	職　名		氏　名	職　名
	主査	増原英彦	教授	審査員	渡部卓雄	教授
	審査員	遠藤敏夫	教授		千葉滋	教授
		南出靖彦	教授			
		脇田建	准教授			

論文審査の要旨（2000 字程度）

本論文は「Supporting multi-scope and multi-level compilation in a meta-tracing just-in-time compiler (メタトレーシングコンパイラにおけるコンパイル範囲・最適化レベルのハイブリッド化に関する研究)」と題し、英文で全 6 章から構成されている。

今日のプログラミング言語の多くは仮想機械を用いて実現されているが、中でも言語仕様定義であるインタプリタプログラムからその言語の仮想機械を生成するメタコンパイラフレームワークを用いた実現方式が注目されている。しかし現在のメタコンパイラフレームワークが提供する実行時コンパイラは固定的なコンパイル方針を備えるのみであり、特定言語向け仮想機械の実行時コンパイラで培われた様々なコンパイル方針に対抗し得るものになっていない。本論文は、メタコンパイラフレームワークの言語仕様定義には、言語の意味定義のみならずコンパイル方針をも記述できることを見出し、実際にメタ実行履歴コンパイラフレームワーク上で複数のコンパイル方針が実現できることを示した。また性能評価を通して、提案手法によって既存の固定的なフレームワークの性能を上回ることが可能だと示した。

論文第 1 章「Introduction」では、研究の背景にある今日の仮想機械を用いた言語実現とメタコンパイラフレームワーク技術について述べている。さらにメタコンパイラフレームワーク技術のコンパイル方針が固定的であるという問題点を指摘し、論文の主張を展開し、論文各章の成果を説明している。

第 2 章「Background」では、論文の背景技術である実行時コンパイラの概要と、特にそのコンパイル範囲について詳細な分類を行っている。またメタコンパイラフレームワーク技術として PyPy および Truffle についてプログラム例を交えながら解説している。

第 3 章「Motivation and Proposal」では論文の動機と提案技術の着想について述べている。ここでは既存の特定言語向け実行時コンパイラには異なるコンパイル方針として、コンパイル範囲の選択方式や生成コードの最適化水準が複数あるにも関らず、現在のメタコンパイラフレームワークではそれらは固定されており、その変更には多大な労力が予想されることを動機として示している。そしてメタコンパイラフレームワークに、コンパイラの挙動を制御する疑似命令をいくつか追加し、言語仕様であるインタプリタの中に疑似命令を埋め込むことで、様々なコンパイル方針が実現可能であることを主張している。従来インタプリタは、解釈されるプログラムの振舞いのみを記述すると考えられていたところに、コンパイラの制御も可能であるという着想はこれまでにない独創的なものだと言える。

第 4 章「BacCaml - a Proof-of-Concept Multi-Scope Meta-Tracing JIT Compiler」は、前章の提案に基づいて異なるコンパイル範囲、具体的には実行履歴単位とメソッド単位の 2 つを選択できる実行時コンパイラを実証的に作成した。直線的な実行をする実行履歴型のコンパイラに対して、条件分岐命令やジャンプ命令の解釈部に疑似命令を用いたインタプリタを与えることによって、メソッド全体に対するコンパイルを実現している。簡素なメタコンパイラフレームワークと、その上での簡素な関数型言語を実現した処理系を構築し評価実験を行ったところ、再帰関数呼出と繰り返し処理を混合したようなプログラムにおいてメソッド単位と実行履歴単位のコンパイルを使い分けることによって全体性能が向上する例を示している。従来から実行履歴型の実行時コンパイラは、再帰関数呼出しを多用するプログラムに対して性能上の問題があることが知られていたところ、それを解決する方向性をメタコンパイラフレームワーク上で示した意義は大きいと考える。

第 5 章「Threaded Code Generation with a Real-World Meta-Tracing JIT Compiler」は、第 3 章の提案に基づいてコード最適化水準が制御できることを実用的なメタコンパイラフレームワーク上で示した。インタプリタは命令取得・デコード・引数取得・実行・結果書込の処理から構成されているが、その一部のみをメタコンパイルの対象とすれば最適化水準の低いコード生成が可能であることを見出し、実際に RPython メタコンパイラフレームワーク上で Smalltalk バイトコードインタプリタに対する実現を行った。このコード生成は通常のコード生成より高速なため、それを早期前階の軽量コンパイラとして用いることで、大規模プログラムを模したプログラムの速度を約 14%向上させる実験結果を得ている。このコード生成は古くより threaded コード生成として知られているものと同等であるが、そのようなコード生成がフレームワークおよびインタプリタに対する少量の改変で達成でき、実際の性能向上が可能であることを示した意義は大きい。

第 6 章「Conclusion」では、本論文を総括するとともに、今後の課題について述べている。

以上のように、本論文ではメタコンパイラフレームワークに対し、コンパイル範囲や最適化といったコンパイル方針を、インタプリタプログラムを通して制御するという新しい着想の下に、実用的なメタコンパイラフレームワークを用いた実現を行い、実際の性能向上可能性を示した。これら本論文で与えられた知見は、仮想機械による言語処理系実現方式に新たな可能性を示すものであり、プログラミング言語の研究分野への新たな価値を創造したという観点からも理学上貢献すること大である。よって、本論文は博士(理学)の学位論文として十分価値があるものと認める。