東北大学 電気通信研究所

研究室外部評価資料

(2013 年度-2018 年度)

Activity Report of Research Laboratory for External Review

April 2013 - March 2019

(FY.2013-2018)

Research Institute of Electrical Communication

Tohoku University

ソフトウェア構成研究室

Software Construction

A. 研究室名 / Research Laboratory

ソフトウェア構成研究室
Software Construction

B. 構成員 / Faculty and Research Staff (as of May 1, 2019)

教授 / Professor

名前 Name	大堀 淳 Atsushi Ohori
分野名 Research Field	ソフトウェア構成研究分野 Software Construction

准教授 / Associate Professor

名前 Name	上野 雄大 Katsuhiro Ueno
分野名 Research Field	高信頼ソフトウェア開発研究分野 Reliable Software Development

助教 / Assistant Professor

名前 / Name	菊池 健太郎 / Kentaro Kikuchi

C. 研究目的 / Research Purpose

今実現しつつある高度情報化社会が，従来通りの信頼性と安全性を確保しながら発展していくためには，高信頼ソフトウェアを効率よく構築する技術の確立が必須である．高信頼プログラミング言語の開発は，その中核をなす重要な課題である．そこで当研究分野では，高信頼プログラミング言語の構築原理，型理論を中心とする基礎理論，および実装技術の確立，さらに，それら基礎研究成果を生かした実用的な次世代高信頼プログラミング言語の実現を目指している．

The general goal of our research is to establish firm theoretical basis and implementation method for flexible, efficient and reliable programming languages and environments for advanced applications. Our main focus is on typed higher-order functional languages, ranging from their type theoretical foundation, compilation method, and efficient implementation on multicore processors. In addition to those basic researches, we are also developing a new practical ML-style programming language, SML#, that embodies some of our research results such as record polymorphism, and high-degree of interoperability with existing languages and databases.

D. 主な研究テーマ / Research Topics

1. 次世代高信頼言語 SML#の開発
2. 関数型言語の並行ゴミ集め方式
3. マルチコアプロセッサのための超並列軽量スレッドサポート
4. コンパイラ最適化のための静的解析
5. 外部データアクセスのための静的型システム

1. Development of SML#, a new language in the ML family
2. Concurrent garbage collection method for functional languages
3. Massively parallel lightweight thread support for multicore processors
4. Static program analysis for compiler optimization
5. Static typing for external data access

E. 学術論文の編数 / The Number of Research Papers						
	2013	2014	2015	2016	2017	2018
(1) 査読付き学術論文 Refereed journal papers	0	1	2	2	0	2
(2) 原著論文と同等に扱う査読付国際会議論文 Full papers in refereed conference proceedings equivalent to journal papers	1	1	2	2	0	1
(3) 査読付国際会議論文 papers in refereed conference	1	6	2	4	3	0
(4) 査読なし国際会議・シンポジウム papers in refereed conference	0	0	0	0	0	0
(5) 総説・解説 Review articles	0	1	0	0	0	0
(6) 査読付国内会議 Refereed proceedings in domestic conference	3	0	0	0	0	3
(7) 査読なし国内会議・講演会 Proceedings in domestic conference	3	2	3	7	2	2
(8) 著書 Books	0	0	0	0	0	0
(9) 特許 Patents	0	0	0	0	0	0
(10) 招待講演 Invited talks	1	0	1	0	0	0

F. 特記すべき研究成果/ Signification Research Achievements (FY.2013–2018)

See Ref. 1. “＃” mark indicates research carried out at a former organization.

2013-2018 年度の研究成果（論文・特許など）のうち，前半（2013-2015 年度）と後半（2016-2018 年度）それぞれで代表的な数件（2-3 件程度ずつ）について，参考資料を引用して，その特徴と学術的意義などを簡単に紹介する．英文のみ，もしくは和文と英文で記載．要約は 300 字程度．論文誌の要約/Abstract のコピー可．学術面での国際的インパクトならびに社会的影響を 100 字程度で記載．必ずしも当該期間内に発表・出版したものに限るのではなく，例えば過去に発表したものでもこの期間内に成果が得られたり，評価されるようになったりしたものも含むものとする．インパクトファクターや被引用件数など，できる限り第三者が定量的に評価できる指標を用いてアピールすること．それらの指標にはそぐわない場合には，その事情とそれに変わる適当な評価指標・尺度を示すこと．

[2013–2015]

1. **The development of SML# language**
   SML# is a new generation of Standard ML we have been developing since 2003. The original goal we had set is to provide moderate but practically important features including: record polymorphism and rank-1 polymorphism, seamless interoperability with C, seamless integration of SQL, and separate compilation and linking, which are not well supported in conventional implementation of ML. Properly supporting some of them is indeed inherently difficult in conventional compilation methods for ML. At the end of previous evaluation period of 2007 – 2012, we had successfully realized all the functionalities we had originally planned, and had released the version 1.0 compiler. We have continued the research and development, and have significantly enhanced its functionalities, stability and efficiency. The important features achieved in the current evaluation period of 2013 – 2018 include the following.
   - the LLVM based native code generator ([1-18]),
   - a fully concurrent GC ([1-13]),
   - runtime support for massively parallel lightweight native (system C library) threads on multicore professors ([1-30]),
   - optimized type-directed compilation based on *finitary polymorphism* ([1-14]),
   - type-safe external interface for JSON and other persistent data ([1-12]).

   This long-standing effort has been well recognized in Japan. Ohori and Ueno have been given several invited talks, including the keynote speech of Ohori at the 30th annual meeting of JSSST (Japan Society for Software Science and Technology) in 2013 ([1-57]). Ohori's proposals for developing software production infrastructure based on SML# have been accepted to Science Council of Japan's Recommendation of the 22nd and 23rd “Japanese Master Plan of Large Research Projects” (Master Plan 2014 and Master Plan 2017) ([8-1, 8-2]). Our development of SML# has also been well recognized in the international research community and the “SML#” have become listed as one of key words of ACM ICFP conference, the leading conference on functional programming.
2. **SML# in Industry**
   The original motivation of SML# includes making the ML language as a robust software development tool in industry. To contribute to software development in industry, we have done various efforts to inseminate SML#, including open lectures for software engineers in industry and consultation and industry-academia joint projects of software developments ([3-1, 3-2]). A success of one of these efforts have been reported in ACM ICFP 2014 conference, as show below.

   Article [1-9]: Atsushi Ohori, Katsuhiro Ueno, Kazunori Hoshi, Shinji Nozaki, Takashi Sato, Tasuku Makabe, Yuki Ito. SML# in Industry: A Practical ERP System Development.
   Abstract: This paper reports on our industry-academia project of using a functional language in business software production. The general motivation behind the project is our ultimate

goal of adopting an ML-style higher-order typed functional language in a wide range of ordinary software development in industry. To probe the feasibility and identifies various practical problems and needs, we have conducted an 15 month pilot project for developing an enterprise resource planning (ERP) system in SML#. The project has successfully completed as we have planned, demonstrating the feasibility of SML#. In particular, seamless integration of SQL and direct C language interface are shown to be useful in reliable and efficient development of a data intensive business application. During the program development, we have found a number of possible extensions of an ML-style language with records. This paper reports on the project details and the lessons learned from the project.

[2016–2018]

We list the papers with their abstracts that represent our major results of our basic research on functional languages and compilers achieved in this period.

1. **High-level and type-safe interface to external data**
   Article [1-12]: Atsushi Ohori, Katsuhiro Ueno, Tomohiro Sasaki, Daisuke Kikuchi. A Calculus with Partially Dynamic Records for Typeful Manipulation of JSON Objects. (ECOOP 2016))
   Abstract: This paper investigates language constructs for high-level and type-safe manipulation of JSON objects in a typed functional language. A major obstacle in representing JSON in a static type system is their heterogeneous nature: in most practical JSON APIs, a JSON array is a heterogeneous list consisting of, for example, objects having common fields and possibly some optional fields. This paper presents a typed calculus that reconciles static typing constraints and heterogeneous JSON arrays based on the idea of partially dynamic records originally proposed and sketched by Buneman and Ohori for complex database object manipulation. Partially dynamic records are dynamically typed records, but some parts of their structures are statically known. This feature enables us to represent JSON objects as typed data structures. The proposed calculus smoothly extends with ML-style pattern matching and record polymorphism. These results yield a typed functional language where the programmer can directly import JSON data as terms having static types, and can manipulate them with the full benefits of static polymorphic type-checking. The proposed calculus has been embodied in SML#, an extension of Standard ML with record polymorphism and other practically useful features. This paper also reports on the details of the implementation and demonstrates its feasibility through examples using actual Web APIs. The SML# version 3.1.0 compiler includes JSON support presented in this paper, and is available from Tohoku University as open-source software under a BSD-style license.

2. **Fully Concurrent GC for Functional Languages**
   Article [1-13]: Katsuhiro Ueno, Atsushi Ohori. A fully concurrent garbage collector for functional programs on multicore processors. (ICFP 2016)
   Abstract: This paper presents a concurrent garbage collection method for functional programs running on a multicore processor. It is a concurrent extension of our bitmap-marking non-moving collector with the Yuasa's *snapshot-at-the-beginning* strategy. Our collector is *unobtrusive* in the sense of the Doligez-Leroy-Gonthier collector; the collector does not stop any mutator thread nor does it force them to synchronize globally. The only critical sections between a mutator and the collector are the code to enqueue/dequeue a 32 kB allocation segment to/from a global segment list and the write barrier code to push an object pointer onto the col-

lector's stack. Most of these data structures can be implemented in the standard lock-free data structures. This achieves both efficient allocation and unobtrusive collection in a multicore system. The proposed method has been implemented in SML#, a full-scale Standard ML compiler supporting multiple native threads on multicore CPUs. Our benchmark tests show drastically short pause time with reasonably low overhead compared to the sequential bitmap-marking collector.

3. **Finitarty Polymorphism: theory and its application**
   Article [1-14]: Atsushi Ohori, Katsuhiro Ueno, Hisayuki Mima: Finitary Polymorphism for Optimizing Type-Directed Compilation. (ICFP'18)
   Abstract: We develop a type-theoretical method for optimizing type directed compilation of polymorphic languages, implement the method in a full-scale compiler of Standard ML extended with several advanced features that require type-passing operational semantics, and report its effectiveness through performance evaluation. For this purpose, we first define a predicative second-order lambda calculus with *finitary polymorphism*, where each type abstraction is explicitly constrained to a *finite type universe*, and establishes the type soundness with respect to a type-passing operational semantics. Different from a calculus with stratified type universes, type universes of the calculus are terms that represent the exact finite set of instance types. We then develop a universe reconstruction algorithm that takes a term of the standard second-order lambda calculus, checks if the term is typable with finitary polymorphism, and, if typable, constructs a term in the calculus of finitary polymorphism. Based on these results, we present a type-based optimization method for polymorphic functions. Since our formalism is based on the second-order lambda calculus, it can be used to optimize various polymorphic languages. We implement the optimization method for natural (tag-free) data representation and record polymorphism, and evaluate its effectiveness through benchmarks. The evaluation shows that 83.79% of type passing abstractions are eliminated, and achieves the average of 15.28% speed-up of compiled code.

G. 特記すべき活動/ Signification Achievements (FY.2013–2018)

See Ref. 1. “＃” mark indicates research carried out at a former organization.

研究室外部評価参考資料の 2 以降を参照しながら，2013-2018 年度のなどの活動の中から特筆すべきものを取り出し，前半（2013-2015 年度）と後半（2016-2018 年度）に分けて簡単に紹介する．英文のみ，もしくは和文と英文で記載．

[2013 – 2015]

1. The proposal for Japanese Master Plan 2014([88])

[2013 – 2015]

1. The proposal for Japanese Master Plan 2017 ([88])

Description of the two:

We have taken the initiative in planning an academia-industry collaborative project on developing an infrastructure for highly efficient and reliable software production. This proposal was accepted by the Science Council of Japan and was included in the 22nd “Recommendation: Japanese Master Plan of Large Research Projects” (Master Plan 2014). We further refined and extended the plan, which was again accepted by the Science Council of Japan and was included in the 23rd “Recommendation: Japanese Master Plan of Large Research Projects” (Master Plan 2017). The revised and the extended plan involves 5 universities (Tohoku U., JAIST, U Tokyo, Tokyo Institute of Technology, Kyusyu U.), 2 national institutes (NII, AIST) and 7 companies (NEC solution innovators, Fujitsu, and others).

Although the Master Plans of Japanese Science Council are only the recommendations to Japanese government, and the master plans themselves do not have research budgets to realize the planned projects, they represents the state of the art of scientific activities of Japan in all the research fields, from philosophy to applied engineering. Acceptance of these proposals have positive impact on inseminating SML# and the related research activities.