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 ２９ 年　 ６ 月　 ８ 日現在

機関番号：１４３０１
研究種目：基盤研究(B)（一般）
研究期間：2013～2016
課題番号：２５２８００２４

研究課題名（和文）ソフトウェア契約に基づく高階型付プログラムの理論

研究課題名（英文）Theory of Higher-Order Typed Programs based Software Contracts

研究代表者
五十嵐　淳（Igarashi, Atsushi）

京都大学・情報学研究科・教授

研究者番号：４０３２３４５６

交付決定額（研究期間全体）：（直接経費）　　13,900,000円

研究成果の概要（和文）：計算効果を持つ言語機構とソフトウェア契約の統合について，FindlerとFelleisenの契約計算やBlame計算の限定継続機構による拡張を行い，動的契約検査のアルゴリズムの形式化やBlame計算としての諸性質の証明を与えた．また，顕在的契約計算の代入機構による拡張を形式化し，その諸性質を示した．代数的データ型による顕在的契約計算を提案し，その上で，代数的データ型に対する契約記述のふたつの方法について比較し，一方から他方へは変換が可能であること示した．また，この計算体系の実装として OCaml 処理系の拡張を行った．ソフトウェア契約の代数的意味論としてトレース意味論を研究した．

研究成果の概要（英文）：We studied the integration of computational effects such as delimited continuatios and mutable state and software contracts; we proposed a contract calculus and a blame calculus with delimited continuations and a manifest contract calculus with mutable state and proved their desirable properties. We proposed an extension of a manifest contract calculus with algebraic datatypes; we compared two styles of specification for datatypes and showed that a contract written in one style can be reduced to the other without changing its meaning in a certain sense. We extended OCaml to implement manifest contracts for datatypes. Finally, we studied trace semantics for a contract calculus as denotational semantics of software contracts.

研究分野：プログラミング言語

キーワード：プログラミング言語　ソフトウェア契約　計算効果　プログラム検証　トレース意味論　代入　限定継続

１．研究開始当初の背景

ソフトウェア契約（以下，単に契約と呼ぶ）の概念は，80 年代に Meyer が提唱した「契約による設計（Design by Contract）」と呼はれるソフトウェアの分割開発に関する考えに基ヅいている（参考文献[1]）．契約は，ソフトウェア部品の正しい挙動についての詳細かつ計算機によって検査可能な仕様記述である．例えば，スタックを実装したモジュールであれば，契約として push 操作の結果か非空スタックとなること，また pop 操作を行うためにはスタックか非空でなければならないことなどか表現できる．契約を使う利点は，契約が満たされているかを実行時に監視することにより，違反か発生した，すなわち正しい挙動を示さなかった場合に，部品の側の欠陥なのか，部品を正しく使用しなかった側の瑕疵なのかの不具合の原因の分離（これを blame と呼ふ）か容易であることにある．例えば，push 操作の結果空のスタックが得られてしまったとしたら，それが直ちにスタックモジュールの欠陥であることがわかりデバグの重要な手がかりとなる．このように，ソフトウェア契約はソフトウェア開発の負担を軽減するとともに信頼性を向上させる技術として注目された．

しかし，実際のプログラミング言語で契約機構を持つものは多くない．その理由のひとつは，契約で記述できる仕様の範囲が狭く，例えば計算過程そのものが第一級データとなる高階関数・オブジェクトについてうまくサポートする方法が知られていなかったためである．この問題に対し Findler と Felleisen は，動的型付けされる関数型言語に対する高階関数をサポートする契約機構を提案し（参考文献[2]），契約検査と blame のためのアルゴリズムをプログラミング言語の操作的意味論の形デ定式化した．この研究をきっかけに，それまで主にソフトウェア工学分野で行われてきた契約についての研究がプログラミング言語の基礎理論分野でも盛んになっている．（参考文献 [2] は，2012 年の ACM ICFP（関数型言語に関する国際会議）て「最も影響を与えた ICFP 2002 論文」として表彰された．）

契約は，当初，プログラム実行時に検査を行う目的で提案されたが，Flanagan や Pierce などは，契約をプログラム実行前に検証を行うための技術である型理論に融合させ，ハイブリッド検査という，契約検査をプログラム実行前に行う枠組みを提案している（参考文献[3,4]）．研究代表者も Pierce のグループと，ML や Haskell などの型多相性を持つ言語でのハイブリッド検査（参考文献[5]）や，再帰関数の存在する言語でのハイブリッド検査についての研究を行ってきている．

ハイブリッド検査には，契約違反がないことを保証することによるソフトウェアの信頼性の向上と，違反しない契約を除去することによる実行時オーバーヘッドの除去という利点があるが，これを現実の言語に導入するためには，（1）代入・例外機構・継続など，プログラムの参照透明性を阻害する，いわゆる計算効果（エフェクト）と契約の有害な相互作用を除去し，（2）ユーザ定義されるデータ型に対する検証技術を向上させる必要があることが明らかになってきている（参考文献[5]）．また，ハイブリッド検査に限らずソフトウェア契約の問題として，（3）blame アルゴリズムの妥当性を議論するための基盤が確立していない，という問題が知られている（参考文献[4]）．

２．研究の目的

本研究では，上記の問題を解決するために

1. 計算効果を持つ言語機構とソフトウェア契約の安全な統合
2. ユーザ定義されるデータ型に対するハイブリッド契約検査手法の確立
3. ソフトウェア契約の代数的意味論の確立

という課題を設定し，オブジェクトや高階関数をサポートし静的に型づけされる高水準プログラミング言語に対しハイブリッド契約検査を適用するための理論的基盤の整備を目的とした．以下，各課題について説明する．

(1) 契約におけるプログラムの仕様は通常，事前条件・事後条件などの形でプログラムを記述する言語と同一の言語で記述する．このため，プログラムに代入やジャンプなと，プログラムの実行状態に大域的な影響（計算効果）を与える機構があると，契約検査の結果が契約上陽に現れないプログラムの実行状態に依存したり，契約の検査そのものが実行状態に影響を与えたりするため，契約が意味するところが不確かなものになりかねない．この問題に対して，本研究では契約の記述にどのような制限を加えれば実用的な仕様記述が可能な範囲で妥当な意味を持つ契約検査が可能なのかを明らかにする．

(2) 一般に，契約が成立することをプログラム実行前に検証するためには，契約が言及するデータ型についての知識が検証器に備わっている必要がある．既存のハイブリッド契約検査で対象としていたのは組み込みのデータ型のみであり，ユーザが定義するデータ型については対象となっていなかった．この問題に対し，ハイブリッド契約検査を助ける，ユーザが定義するデータ型についての知識（条件）の記述法を考案し，それに基づく検証器の作成に取り組む．

(3) 上述のように， Findler と Felleisen は，契約検査の操作的意味論を与えることにより，プログラムの欠陥箇所を特定する blame のアルゴリズムを定式化した．しかし，このアルゴリズムがどういった意味で妥当なのかは自明ではない．実際， Findler と Felleisen のアルゴリズムには欠点があることが指摘されてきている（参考文献[4]）．本研究では契約の代数的意味論を与えることにより，妥当な契約検査・blame アルゴリズムの基準を明らか

にし，その基準による既存のアルゴリズムの検証を行う．

３．研究の方法

(1) 計算効果を持つ言語機構とソフトウェア契約の統合

研究目的の項で述べたように，この課題では，代入や例外，継続など計算効果を起こす機構のある言語のもとで，契約が妥当な意味を持つための条件（制限）を型理論的な立場から探求する．計算効果のための一般的な型理論としてエフェクトシステムという枠組みがあるので，これと契約の型理論を組み合わせることを目標とするが，ひとまず，限定継続と呼ばれる特定の計算効果の元での契約を対象とする．限定継続を考えるのは，問題が具体的になって取り扱いが容易になること，一方で，限定継続は表現力が高く，古典的な継続の使い方である大域脱出だけでなく，代入機能（の一部）も表現できることから，その後の一般的な計算効果と契約の理論につながり易いと考えているためである．

具体的には，多相契約計算と呼ばれる契約のための既存の型理論（参考文献[5]）と Asai, Kameyama の限定継続の型理論(参考文献[6])を組み合わせた型理論を構築しその性質を調べる．この際，契約に使われるプログラムとして，純粋な(全く計算効果を許さないような)式のみを許すような，強く制限をした体系にすれば，契約検査中に大域脱出などをしないという意味で，契約の型理論として妥当な体系が得られるはずである．これを起点として，強い制限をどのような形で緩めることができるかについて実用的な観点から検討を進める．

(2) ハイブリッド契約検査技術の発展

研究目的の項で述べたように，既存のハイブリッド契約検査の研究では困難であった，プログラマによって定義されるデータについての性質を表現し，それを契約検証器にヒントとして与える枠組みを研究する．既存研究では，契約の最小単位は言語に組込みとして用意されているデータ型に対する仕様記述であったが，研究代表者らの多相契約計算の研究(参考文献[5])により，抽象データ型など，プログラマが定義するデータについての契約を最小単位とする契約が記述可能になった．しかしながら，プログラマが定義するデータについての効果的な検証を行うためには，そのデータと関連する操作についての性質，例えばスタックであれば push の直後に pop したら元のスタックに戻るといった性質，などを検証器に与える必要がある．

具体的には，代数仕様言語の研究や現在知られているハイブリッド検査手法の実装と実例を使った実験を行い，プログラマが定義するデータについての性質の表現方法を検討し，表現された性質を検証器へのヒントとして与えるための手法を考案する．この際，偽の性質をプログラマ（すなわちデータ型を実装する側）が宣言すると，検証が健全ではなくなってしまうという問題が発生するため，性質の宣言の正しさを何らかの意味で担保する仕組みを導入することが重要であると考えている．

(3) ソフトウェア契約の代数的意味論の確立

研究目的の項で述べたように，契約の代数的意味論を与えることにより，既存研究では不十分であった契約検査と blame のアルゴリズムの妥当性について考察するのがこの課題の目的である．blame のアルゴリズムの肝は，プログラム部品の内部実装と，それをクライアントとして使う外部に分割して，契約違反が生じた際に内部・外部のどちらに瑕疵があったかを指摘する部分にある．一方，プログラミング言語の代数的意味論として使われるゲーム意味論は，プログラムの実行を外部環境との相互作用として捉える点で，blame アルゴリズムの考え方との類似点が認められる．このことをヒントとして研究を進める．

具体的には，まず，単純型付ラムダ計算に単純な契約を付加した体系を考え，単純型付ラムダ計算のゲーム意味論を拡張することで，契約や blame の概念をゲーム意味論上で捉える．その上で，既存の blame のアルゴリズムが，ゲーム意味論上での blame に対して（ある意味で）健全であるかどうかを調べていく．

参考文献

[1] B. Meyer. Touch of Class. Springer-Verlag (2009).

[2] R. B. Findler and M. Felleisen. Contracts for Higher-Order Functions. In Proc. of ICFP (2002). [3] C. Flanagan. Hybrid Type Checking. In Proc. of ACM POPL (2006).

[4] M. Greenberg, B. C. Pierce, S. Weirich. Contracts made manifest. In Proc. of POPL (2010).

[5] J. F. Belo, M. Greenberg, A. Igarashi, B. C. Pierce. Polymorphic contracts. In Proc. of ESOP (2011).

[6] K. Asai, Y. Kameyama. Polymorphic Delimited Continuations. In Proc. of APLAS 2007.

４．研究成果

(1) 計算効果を持つ言語機構とソフトウェア契約の統合

まず，計算効果を持つ言語機構として限定継続を取り上げ、動的契約検査の方式の設計および意味論の検討を行った。検査方式は、Danvy と Fillinski のアンサー型の変更を許す型システムにヒントを得て、関数に対する契約を、契約の四つ組として与えることとした。この方式をラムダ計算上でモデル化するために操作的意味論を検討し、その定義を与えた。たダし、契約違反の際にどのモジュールに 責任があるかを定める blame アルゴリ

ズムについて検討課題が残った。
その後，契約計算体系の意味論を見直し，まず限定継続機構を除去する CPS 変換を定義し，その意味論を引き戻すことで新しい意味論を得た．また，この意味論から導出できるプログラムの等価性が CPS 変換を通じて健全であることを証明した．古い意味論にあった，契約違反が生じた時の blame の非自明な部分も，新しい意味論では直観的には理解がしやすいものになっているが，この直観を形式化することがまだ課題として残っている．(研究発表[20])
その後, 同じ限定継続機構 shift/reset に対して，静的型システムと動的型システムを統合したような検査機構を形式化し，その諸性質を証明した．(論文[5]・研究発表[7])これは Wadler と Findler による blame calculus の拡張になっている．また諸性質として型健全性だけでなく，動的検査失敗時のエラー (blame) の正しさを示す blame theorem や，blame calculus の CPS 変換とその健全性などを示すことができた．さらに，この結果を元に，プログラマが実際にプログラムを書く表面言語(これを blame calculus へ変換することによって実装される)を設計し, 型推論問題について研究し主要型性を持つ型推論アルゴリズムの構築に成功した．(論文[1]・研究発表[1,3,5])
また，従来の顕在的契約計算に計算効果の代表のひとつである代入機構を導入し，形式的計算体系を与え，その型健全性を示すことに成功した．本研究で考えているソフトウェア契約の枠組みにおいては，プログラムの仕様をプログラム記述のための言語で書くので，その言語に代入など状態を変化させる機能と素朴に組み合わせると，プログラムの仕様が状態に依存してしまう，仕様記述を実行するとプログラム状態を変更してしまう，という問題があった．本研究では，プログラムで代入を使う箇所と使わない箇所を切りわけることでその問題を解決した．また，代入を使用する部分の契約記述の新しい手法を既存のホーア型と呼ばれる機構を修正し組み込んだ．この結果には，プログラミング言語分野で最も権威のある国際会議のひとつである ACM POPL に採択された．(論文[3]・研究発表[2])

(2) ハイブリッド契約検査技術の発展
新しいハイブリッド契約検査技術として、プログラマが定義可能な代数的データ型に対するハイブリッド契約検査の基礎となる計算体系について研究を行った。代数的データ型に対する契約は、データ型の値を入力として真偽値を返す関数で記述する方法と、Coq などに代表される、コンストラクタに契約を与えることで記述する方法があるが、それらのふたつの方法の利害得失について考察するとともに、前者の方法で記述された契約を後者の方法に変換する手法を考案し、その正当性について形式体系の上で証明した。また、OCaml 用のプリプロセッサ CamlP4 を用い、形式体系のプロトタイプ実装を行った。(研究発表[21])
これらの結果についても ACM POPL などで成果発表を行った．(論文[8,12]・研究発表[15,22])また，この枠組みではデータ型間のキャストの際に生じるコンストラクタ変換のコストが実用上問題になると思われたため，その削減手法についても研究したが目立った成果はでなかった．しかし，計算体系の実装を OCaml コンパイラを改造することで進め，その過程で, OCaml の型推論とハイブリッド契約機構の融合に取組んだ．結果として，型を省いた契約情報つきのプログラムから，型情報を OCaml と類似の型推論技法により回復するとともに適当なキャストを挿入するための型推論アルゴリズムを構築し，その健全性を示した．(研究発表[6])

(3) ソフトウェア契約の代数的意味論の確立
契約計算に対する新しい意味論として、トレース意味論を検討した。トレース意味論は、並行計算などにも見られるが、プログラム部品と環境との相互作用の系列を意味として与えるもので、ゲーム意味論とも関連が深い。具体的には、Findler と Felleisen による契約計算に対するトレース意味論の定義を与えた。その後，Findler と Felleisen による操作的意味論とある意味で一致することの予想や，また，付加的結果として，この意味論が文脈等価性と呼ばれるプログラム等価性に対し完全抽象的(fully abstract)であるという予想をたてた．(論文[10]，研究発表[13,19])
その後，その内容を精査する過程で計算体系の定義や証明に細かなミスが多数見つかったために，体系定義の見直し・簡略化とミスの除去に取り組んだが，残念ながら期間内に定義の修正が完了せず研究発表を行うまでに至らなかった．操作的意味論とトレース意味論というふたつの意味論の間である種の等価性を示すのが目標のひとつとなるが，両者が整合するようにそれぞれの定義を与えるのが困難であるのに起因している．

(4) その他，多相的顕在的契約計算の先行研究[参考文献 5]にあった誤りを発見し, その修正を行う論文をプログラミング言語分野で最も権威のある論文誌のひとつである ACM TOPLAS に発表した．(論文[2])

５．主な発表論文等

〔雑誌論文〕（計 13 件）
[1] Yusuke Miyazaki, Atsushi Igarashi.“A Type Reconstruction Algorithm for Gradually Typed Delimited Continuations” 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)論文集， 査読有, 2017 (会場でのオンライン配布のためページ番号なし)
[2] Taro Sekiyama, Atsushi Igarashi, and Michael

Greenbeg. “Polymorphic manifest contracts, Revised and Resolved”ACM Transactions on Programming Languages and Systems,**査読有**,39**巻**,2017,3:1-3:36
DOI:10.1145/2994594
[3] Taro Sekiyama, Atsushi Igarashi. “Stateful manifest contracts” Proc. of ACM SIGPLAN-SIGACT Symp. on Principles of Prog. Languages, **査読有**, 2017, 530-544
DOI:10.1145/3009837.3009875
[4] Qi Tan, Kohei Suenaga, Atsushi Igarashi. ”An Extended Behavioral Type System for Memory-Leak Freedom”**日本ソフトウェア科学会第** 33 **回大会論文集**, **査読無**, 2016 (**会場でのオンライン配布のためページ番号なし**)
[5] Taro Sekiyama, Soichiro Ueda, Atsushi Igarashi. “Shifting the Blame: A Blame Calculus with Static Delimited Control”Lecture Notes in Computer Science (Proceedings of Asian Symposium on Programming Languages and Systems), **査読有**, 9458 **巻**, 2015, 189-207
DOI:10.1007/978-3-319-26529-2_11
[6] Yuki Nishida, Atsushi Igarashi.”Manifest Contracts for OCaml” Online Proceedings of ACM Workshop on ML, **査読有**, 2015 (**会場でのオンライン配布のためページ番号なし**)
[7] Seiji Umatani, Tomoharu Ugawa, Masahiro Yasugi. “Design and Implementation of a Java Bytecode Manipulation Library for Clojure” Journal of Information Processing, **査読有**,23(5) **巻**,2015,716-729
DOI:10.2197/ipsjjip.23.716
[8]Taro Sekiyama, Yuki Nishida, Atsushi Igarashi.”Manifest Contracts for Datatypes” Proceedings of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL),**査読有**, 1 **巻**, 2015, 195-207
DOI:10.1145/2676726.2676996
[9] Tan Qi, Kohei Suenaga, Atsushi Igarashi. ”A Behavioral Type System for Memory-Leak Freedom” **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**(PPL2015) **論文集**, **査読有**, 2015 (**会場でのオンライン配布のためページ番号なし**)
[10]**村井 涼**, **中澤 巧爾**, **五十嵐 淳**. ”**高階契約を持つプログラミング言語に対するトレース意味論**” **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**(PPL2015)**論文集**,**査読無**, 1 **巻**, 2015 (**会場でのオンライン配布のためページ番号なし**)
[11] Tatsuya Sonobe, Kohei Suenaga, Atsushi Igarashi. ”Automatic Memory Management Based on Program Transformation using Ownerships” Springer LNCS (Proc. of Asian Symposium on Programming Languages and Systems), **査読有**, 8858 **巻**, 2014, 58-77
DOI:10.1007/978-3-319-12736-1_4
[12]**関山太朗**, **西田雄気**, **五十嵐淳**. ”**顕在的契約計算における代数的データ型**” **第** 14 **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**(PPL 2014)**オンライン論文集**,**査読有**, 1 **巻**, 2014 (**会場でのオンライン配布のためページ番号なし**)
[13] Seiji Umatani.”Practical Implementation Techniques of Ambient Calculus in Conventional Dynamic Languages” Proc. of ACM Symp. on Applied Computing, **査読有**, 1 **巻**, 2014, 1345-1351

〔**学会発表**〕 **（計** 22 **件）**
[1] Yusuke Miyazaki, Atsushi Igarashi. “A Type Reconstruction Algorithm for Gradually Typed Delimited Continuations” **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**, 2017/03/08～10, **山梨県笛吹市**
[2] Taro Sekiyama, Atsushi Igarashi. “Stateful manifest contracts” ACM SIGPLAN-SIGACT Symp. on Principles of Prog. Languages (POPL) **（国際学会）**, 2017/01/18～20, Paris(France)
[3] Yusuke Miyazaki, Taro Sekiyama, Atsushi Igarashi. “Gradual typing for delimited continuations” International Workshop on Scripts to Programs **（国際学会）**, 2016/06/17, Roma(Italy)
[4] Qi Tan and Kohei Suenaga, Atsushi Igarashi. “An Extended Behavioral Type System for Memory-Leak Freedom”**日本ソフトウェア科学会第** 33 **回大会**, 2016/09/06～09,**仙台市青葉区**
[5]**宮﨑 勇輔**, **関山 太朗**, **五十嵐 淳**.”**限定継続演算子** shift/reset **のための漸進的型付け**” **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**（PPL2016）, 2016/03/07～09, **岡山県玉野市**
[6] Yohei Miyamoto, Kohei Suenaga, and Koji Nakazawa.”A Denotational Semantics of a Probabilistic Stream-Processing Language” Workshop on Probabilistic Programming Semantics (PPS 2016) **（国際学会）**, 2016/01/23, St. Petersburg, FL,(USA)
[7] Taro Sekiyama, Soichiro Ueda, Atsushi Igarashi. “Shifting the Blame: A Blame Calculus with Static Delimited Control”13th Asian Symposium on Programming Languages and Systems (APLAS2015) **（国際学会）**, 2015/11/29～12/01, Pohang(Korea)
[8] Koji Nakazawa, Ken-etsu Fujita, “Compositional Z: Confluence Proofs for Permutative Conversion”**第** 32 **回日本ソフトウェア科学会大会**,2015/09/08～11,**東京都新宿区**
[9] Yuki Nishida, Atsushi Igarashi.”Manifest Contracts for ML” ACM Workshop on ML **（国際学会）**, 2015/09/03, Vancouver(Canada)
[10] Kohei Suenaga. “Formal Verification of Software, Continuous, and Hybrid Systems; Or: How Do We Verify Our Program is Correct?” Machine Learning Summer School 2015 **（招待講演）（国際学会）**, 2015/08/27, **京都府京都市**
[11] Koji Nakazawa.”Lambda Calculi and

Confluence from A to Z” 4th International Workshop on Confluence (IWC2015)**（招待講演）（国際学会）**, 2015/08/02, Berlin(Germany)
[12] Kohei Suenaga. “Nonstandard Analysis Meets Programming Language Theory” The 12th Intl. Conference on Computability and Complexity in Analysis (CCA 2015) **（招待講演）（国際学会）**, 2015/07/13, **東京都千代田区**
[13]**村井 涼**, **中澤 巧爾**, **五十嵐 淳**. ”**高階契約を持つプログラミング言語に対するトレース意味論**” **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** (PPL2015), 2015/03/04〜06, **愛媛県松山市**
[14] Tan Qi, Kohei Suenaga, Atsushi Igarashi. ”A Behavioral Type System for Memory-Leak Freedom”**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**(PPL2015), 2015/03/04〜06, **愛媛県松山市**
[15] Taro Sekiyama, Yuki Nishida, Atsushi Igarashi. ”Manifest Contracts for Datatypes” ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2015/01/15〜17, Mumbai(India)
[16] Tatsuya Sonobe, Kohei Suenaga, Atsushi Igarashi. ”Automatic Memory Management Based on Program Transformation using Ownerships” Asian Symposium on Programming Languages and Systems, 2014/11/17 〜 19, Singapore
[17] Minoru Kinoshita, Kohei Suenaga, Atsushi Igarashi. ”Automatic Synthesis of Combiners in the MapReduce Framework: An Approach with Right Inverse” International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR2014), 2014/09/09〜11, Canterbury(UK)
[18] Seiji Umatani. ”Practical Implementation Techniques of Ambient Calculus in Conventional Dynamic Languages” ACM Symp. on Applied Computing, 2014/03/24〜28, Gyeongju(Korea)
[19]**村井 涼**, **五十嵐 淳**, **中澤 巧爾**. ”**契約つきモジュール計算のトレース意味論に向けて**” **第** 14 **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** (PPL 2014), 2014/03/05〜07,**熊本県阿蘇市**
[20]**上田 宗一郎**, **関山 太朗**, **五十嵐 淳**. ”**限定継続を備えた計算体系へのソフトウェア契約の導入**”**第** 14 **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**(PPL 2014), 2014/03/05〜07,**熊本県阿蘇市**
[21]**西田雄気**, **関山太朗**, **五十嵐淳**. ”**型に基づく実行時契約検査機構の実装**” **第** 14 **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**(PPL 2014), 2014/03/05〜07,**熊本県阿蘇市**
[22]**関山太朗**, **西田雄気**, **五十嵐淳**. ”**顕在的契約計算における代数的データ型**”**第** 14 **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**(PPL 2014), 2014/03/05〜07,**熊本県阿蘇市**

〔図書〕（計 0 **件）**
〔産業財産権〕
○出願状況（計 3 **件）**

名称：不変条件生成装置，コンピュータプログラム,不変条件生成方法プロ グラムコード製造方法
発明者：末永幸平，樹下稔，小島健介
権利者：京都大学
種類：特許
番号：特願 2016-017441
出願年月：2016 **年** 02 **月** 01 **日**
国内外の別：国内

名称：不変条件生成装置，コンピュータプログラム，不変条件生成方法，プログラムコード製造方法
発明者：末永幸平，樹下 稔，小島健介
権利者：京都大学
種類：特許
番号：特願 2016-017419
出願年月日：2016 **年** 02 **月** 01 **日**
国内外の別：国内

名称：不変条件生成装置，コンピュータプログラム，不変条件生成方法，プログラムコード製造方法
発明者：末永幸平，樹下 稔，小島健介
権利者：京都大学
種類：特許
番号：PCT/JP2017/003295
出願年月日：2017 **年** 01 **月** 31 **日**
国内外の別：国内

〔その他〕
五十嵐淳のホームページ
http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/

６．研究組織
(1)**研究代表者**
五十嵐　淳(IGARASHI, Atsushi)
京都大学・大学院情報学研究科・教授
研究者番号：４０３２３４５６
(2)**研究分担者**
馬谷　誠二(UMATANI, Seiji)
京都大学・大学院情報学研究科・助教
研究者番号：４０３７８８３１
末永　幸平(SUENAGA, Kohei)
京都大学・大学院情報学研究科・准教授
研究者番号：７０６３３６９２
中澤　巧爾(NAKAZAWA, Koji)
名古屋大学・大学院情報学研究科・准教授
研究者番号：８０３６２５８１
(3)**連携研究者**
(**なし**)
(4)**研究協力者**
(**なし**)