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 ２０ 日現在

機関番号：３２７１０
研究種目：基盤研究(C)（一般）
研究期間：2013～2015
課題番号：２５３３００９６

研究課題名（和文）定理証明器によるモジュラーなソフトウェア検証

研究課題名（英文）Modular software verification with a proof assistant

研究代表者
田辺　良則（TANABE, Yoshinori）

鶴見大学・文学部・教授

研究者番号：６０４４３１９９

交付決定額（研究期間全体）：（直接経費）　　3,600,000円

研究成果の概要（和文）：定理証明支援系Coqから，プログラミング言語Scalaへのコード抽出を実現した．Coqによって，コードが正しく動作することの証明を記述した上でコード抽出をすれば，抽出されたコードの正しさが保証される．Scalaコードは，Javaで記述された多数のシステムと親和性が高いと期待される．コード抽出を行うコード自体もCoqで記述されており，その動作が妥当であることが，Coq上の証明として与えられている．

研究成果の概要（英文）：We have implemented code extraction from proof assistant Coq to programming language Scala. Once a proof is given to a code written in Coq, then the extracted code is guaranteed that it behaves correctly. There are many working systems written in Java, and Scala code is easily integrated with them because it runs on Java VM. Our code itself is originally written in Coq and is extracted from a proof on Coq that it works correctly.

研究分野：情報理工学

キーワード：定理証明支援系　ソフトウェア検証

１．研究開始当初の背景

あらゆる分野で情報技術が応用されているため，正しく動作するソフトウェアを設計，実装することは，重要な課題であると考えられる．これを実現するための諸技術の一つに「形式手法」によるソフトウェアの検証があり，大きな成功を収めている．

代表的で重要な２ つの技術に，モデル検査と定理証明がある．前者は，ソフトウェアを有限状態遷移系としてモデル化した上で，全経路を効率よく調べることによって，あらゆる場合の正しい動作を保証するものである．一方後者では，ソフトウェアを一般的な数学的構造としてモデル化した上で，検証すべき内容をその構造に関する定理として述べ，その証明を定理証明支援系を利用して構築する．これによって，そのソフトウェアが問題なく動作することを数学的に確認する．そして，証明支援系が持つ，証明から実行コードを抽出する機能により，実行可能な「検証済み」プログラムが生成される．

定理証明手法の潜在的な適用可能範囲は非常に広い．モデル検査手法の場合には，状態遷移系という特定のモデリングが可能な検証パターンでのみ強みを発揮するが，定理証明では，凡そ数学的モデリングが可能であれば，原理的には必ず適用することができるという意味で，非常に強力な手法であると言える．理論的な観点のみならず，ソフトウェア開発現場における実システムの検証にも応用することは可能であり，実際，今井はすでに定理証明支援系 Coq の OCaml 生成機能を用いて，このようなシステム開発の試みを行っており，成功を収めている．

しかし，研究開始当初の状況では上のような適用例はあまり多くなく，同じ形式手法による検証手法であるモデル検査手法がかなり普及していることと比較すると大きな差がある．よくあげられる理由は，適用する技術者に論理学に関する基礎知識が要求されることや，検証を完了するまでに要する工数が比較的大きいことである．一方で上記成功例を検討すると，開発環境の要因も大きいことがわかる．この例では，OCaml で全体システムが開発されており，そのコアの部分に Coq による検証を適用した上で，システムを構成する他のコンポーネントとシームレスに融合することが可能だった．

ソフトウェアシステムですべての部分が検証の観点から同様の重要性を持っていることはほとんどない．形式検証はコストの高いので，重要度の高い部分のみに適用することが望まれる．しかし，このシナリオが機能する機会が非常に多いとは言えない．定理証明器によって生成できるコードが関数型言語に限られており，開発現場で主流となっている言語で開発されたシステムとの親和性が低いからである．

２．研究の目的

本研究の目的は，モジュラーな検証を実現することであり，定理証明手法の検証への適用を現実的に可能にする手法を研究する．定理証明支援系としては，近年もっとも盛んに研究されている支援系である Coq を用いる．Coq が現在抽出できる実行可能コードの言語は OCaml，Haskell，Scheme であるが，本研究においては，プログラミング言語 Scala コードの抽出を可能とする．Scala コンパイラは Java VM 上で動作するバイトコードを生成するため，広く使用されている Java 言語で記述されているソフトウェアシステムとオーバーヘッドなしで統合することが可能となる．

定理証明支援系による実行可能コード生成は，理論的にはすでに成熟している手法であり，命令型言語のコード生成さえすでに実装が試みられている．しかし，この手法を現実のソフトウェア開発での検証に適用するためにはまだギャップがある．本研究ではモジュラーな検証を実現することでこのギャップを埋めることをめざす．

３．研究の方法

Coq からの Scala コード抽出については，今井による試験的な実装がすでに存在する．既存のコード抽出機能は，Coq コードを入力としてとり，まず，MiniML という言語への変換が行われ，その後，各対象言語へ変換される．この後半のステップで，対象言語の持つ強力な型推論機能が使用されている．Scala への変換を行う際には，限定された型推論機能しか用いることができず，このために抽出に失敗する場合があることが知られている．そこで本研究では，MiniML 中間言語を拡張して，Scala の型推論機能の範囲内で正しいコード生成をする方法をとる．

ソフトウェアシステムの開発では，全コードを自ら書くことは少なく，フレームワークを利用することが多い．ウェブシステム構築のための J2EE，Spring，Lift，分散計算のための MapReduce，Hive 等がある．利用の際には，システム開発者は Java クラスの形で部品を作成し，提供されるサーバ等の環境と組み合わせてシステムを構成する．本研究では，　モジュラー検証機構を利用して，フレームワーク上のアプリケーションを検証できるようにすることもめざす．

４．研究成果

(1) コード抽出基本機能

Coq から Scala へのコード抽出に関して，今井の方法を改良した．

この研究では，コード抽出のためのコード自体を，Coq で記述することを行っている．

そして，コード抽出が妥当であることの証明を Coq で記述してる．このため，実装したコード抽出プログラムは，それ自身が検証済プログラムであるという性質を持っている．なお，コード抽出の妥当性としては，抽出したコードが型検査を通ること(型の妥当性) と，抽出したコードの計算結果が，抽出前の計算結果と一致すること(計算の妥当性) の二つの性質が考えられるが，実際に Coq による証明を行ったのは，型の妥当性までであり，計算の妥当性については，(いくつかの仮定の下で) 自然言語による記述にとどまっている．

このプログラムは Coq のバージョンアップによる影響が少なく，また抽出される Scala コードが型検査を通ることは保証されているという特長を持つ．Coq から MiniML (後述) への変換には既存の実装をそのまま利用している．

従来のコード抽出の方法では，まず Coq の項は MiniML と呼ぶ言語の項に変換される．このとき変換後の MiniML の項は型が付かないことがあるため，必要な場所に型キャストが挿入される．型キャストを挿入する場所を調べるには，型推論アルゴリズム M を元にしたアルゴリズム M'を用いる．MiniML の構文は OCaml などの関数型言語のサブセットとみなすことができるため，こうして得られた MiniML の項は自然に抽出先言語のコードとして書き出すことができる．この従来のコード抽出は正当性を満たすことが証明されている．しかし MiniML に変換する過程で項に含まれる型の情報が捨てられてしまうため，型を明示する必要がある Scala へ同様にしてコード抽出することはできない．Scala コードを書き出すには，中間言語の項が適切な型の情報を持っている必要がある．そこで，我々は，MiniML に型の情報を加えた言語 MiniML'を定義した．MiniML'の項を作るために考えられる方針の一つとして，既存のコード抽出の実装に手を加えることで，Coq から MiniML への変換と同様に Coq から MiniML'に変換するというものがある．しかしこの方針では，既存の実装に変更があるたびに対応しなければならないという問題点がある．そこで，別の方針によって MiniML'の項を作り，Scala へコード抽出する方式を考えた．この抽出方式では，Coq から MiniML への変換，MiniML から MiniML'への変換という二段階の変換を行う．一段階目の変換は従来のコード抽出と共通の変換である．二段階目の変換アルゴリズムは M'の持つ型推論能力を利用して開発し，M''と名付けた．この新しい抽出方式が型の妥当性を満たすことを証明した．その一方で型の情報を完全に復活させることはできないため，計算の妥当性は満たさず，型キャストが実行時エラーを起こす可能性がある．そこで，計算の妥当性が満たされる十分条件を示し，この条件のもとでは正しく実行されることを証明した．

(2) 副作用への対応

Coq を用いて記述してコード抽出をしたプログラムは，当然，副作用を持たない．しかし，現実のシステムでは，副作用を持つモジュールも多数使用されており，この一部を検証済みコードで置き換えたいという要望も起こり得る．

この問題に対処するため，既存のライブラリである Ynot を用いて副作用を持つコードを検証する試みを行った．加算操作を持つカウンタとスタックを用いて，式の値を計算するメソッドを対象として選んだ．

Scala コードから Ynot への変換は，(機械的とまでは言えないが) 明瞭であり，結果として，22 個の証明責務が生成され，これらに証明を与えることで，検証を行うことができた．Ynot における表明をどの程度記述するかがポイントになるが，今回の検証では，まず多少冗長に表明文を附して，生成された証明責務を完全に処理した後，試行錯誤により不要なものを段階的に削除して洗練するという手順をとった．妥当な手順であったと評価している．

一般に適用することを考えると，Scala コードから Ynot コードへの変換をする指針を与える必要があるが，これは今後の課題として残っている．

(3) 事例研究

① MapReduce モデリング

MapReduce は，クラウドコンピューティングなどの分散システムで用いられるフレームワークである．先行研究において我々は MapReduce の計算モデルを Coq 上に構築していた．この研究プロジェクトでは，この計算モデルをベースに，Coq コード生成も利用して，検証付き map メソッドと reduce メソッドを作成し，その性能を評価した．

先行研究では，中間に作成されるリストの大きさによりその性能に問題が生じることがわかったので，この研究では，その問題を回避できる枠組みの作成を行い，性能に改善を与えた．

② LMNtal 中間コード

LMNtal は，一種のグラフ書き換えシステムであり，シミュレーションとモデル検査をサポートしている．この研究では，LMNtal コンパイラの内部命令生成を検証付きコードで行うことをめざして，一部の生成系について実験的に検証を行った．

複数のレベルのモデリング方法が考えられたが，コード生成に寄与するよう，低いレベルでの記述を選択した．従来，複数回円バグをしていたモジュールについて，記述を行い，現在導入されている方式では正しくコードが生成されることを確認できた．

(4) その他

以上の内容から派生した得られた研究成果で，定理証明支援系 Coq を用いたものとして，以下があげられる．

① Petri Net の被覆性判定の形式化

Petri Net は，分散システムのモデリング化のためによく用いられている数学的構造である．この被覆性判定には．Karp-Miller 木を用いた証明が知られているが，trivial なものではない．Karp-Miller 木構成が停止することの証明に困難さがあるが，almost-full 関係を利用して，これを実現した．

③ BASE 特性保証のための検証

システム群の整合性を保証するために，BASE 特性の保証が有効である．このためには，システムの更新操作が冪等かつ可換であることが条件になる．従来はこれをレビューによって確認していたが，Coq を用いた保証ができれば，より確実である．

実際のシステム構築現場で利用できるようにするためには，できるだけの自動化とノウハウの明示化が望ましい．この研究では，自動化を進めるための tactics を多数定義し，これを用いることにより，証明手続きの自動化が進められることを示した．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計　5 件）

① 姜帆，田辺良則，本位田真一：Coq を使用した MapReduce アプリケーションの検証と Scala コード．電子情報通信学会論文誌．査読有．J97-D (2014) pp.625-634

〔学会発表〕（計　13 件）

① Mitsuharu Yamamoto: A formal proof of Higman's lemma in Coq/SSReflect. 11th Theorem Proving and Provers (TPP2015). 2015 年 9 月 16 日．神奈川大学（神奈川県平塚市）

② 松本早貴，山本光晴：ペトリネットに対する Karp-Miller 木構成の Coq/SSReflect による実装．11th Theorem Proving and Provers (TPP2015). 2015 年 9 月 16 日．神奈川大学（神奈川県平塚市）

③ 湯浅能史，田辺良則：Java プラットホーム上での Coq 検証済コードの実行について．日本ソフトウェア科学会第 32 回大会．2015 年 9 月 9 日．早稲田大学（東京都新宿区）

④ 田辺良則，逸見港，今井宜洋，萩谷昌己：Coq から Scala へのコード抽出とその妥当性．Theorem Proving and Provers for Reliable Theory and Implementations (TPP2014). 2014 年 12 月 3 日．九州大学・西新プラザ大会議室（福岡県福岡市）

⑤ 逸見港，田辺良則，今井宜洋，萩谷昌己：検証済のコードによる Coq から Scala へのコード抽出．第 31 回ソフトウェア科学会大会．2014 年 9 月 10 日．名古屋大学（愛知県名古屋市）

⑥ 逸見港，田辺良則，萩谷昌己：Coq から Scala へのロバストなコード抽出．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（PPL2014）．2014 年 3 月 6 日．阿蘇の司ビラパークホテル(熊本県阿蘇市)

⑦ 逸見港，田辺良則，萩谷昌己：Coq から Scala へのロバストなコード抽出に向けて．第 11 回ディペンダブルソフトウェアワークショップ(DSW2013)．2013 年 12 月 26 日．ホテルリゾーピア熱海（静岡県熱海市）

⑧ Yuuki Shinobu, Yoshinori Tanabe, Kazunori Ueda: Formalization of the Graph Rewriting Operations of LMNtal by Coq. APLAS2013（ポスター）2013 年 12 月 10 日．Rydges on Swanstion (Melbourne, Australia)

⑨ 高橋哲也，今井宜洋，田辺良則：システム群の BASE 特性を保証するための Coq を用いた検証．第 20 回ソフトウェア工学の基礎ワークショップ（FOSE2013）．2013 年 11 月 29 日．ゆのくに天翔（石川県加賀市）

⑩ 信夫裕貴，田辺良則，上田和紀：LMNtal におけるグラフ書き換え操作の Coq による形式化．第 30 回日本ソフトウェア科学会大会．2013 年 9 月 13 日．東京大学(東京都文京区)

６．研究組織

(1)研究代表者

田辺　良則（TANABE, Yoshinori）
鶴見大学・文学部・教授
研究者番号： 60443199

(2)研究分担者

萩谷　昌己（HAGIYA, Masami）
東京大学・
大学院情報理工学系研究科・教授
研究者番号：30156252

山本　光晴（YAMAMOTO, Mitsuharu）
千葉大学・大学院理学研究科・准教授
研究者番号： 00291295

(3)連携研究者

なし

(4) 研究協力者

湯浅　能史（YUASA, Yoshifumi）
逸見　港（HENMI, Ko）