科学研究費助成事業　　研究成果報告書

平成　２６　年　　５　月　２９　日現在

機関番号：　１３９０１
研究種目：　若手研究(B)
研究期間：　2010～2013
課題番号：　２２７０００２８

研究課題名（和文）進んだ型システムを持ったプログラミング言語における型検証器の機械的な証明

研究課題名（英文）Mechanically certified proof of a type checker for an advanced type system

研究代表者
Ｊ　Ｇａｒｒｉｇｕｅ（Garrigue, Jacques）

名古屋大学・多元数理科学研究科・准教授

研究者番号：８０２７３５３０

交付決定額（研究期間全体）：（直接経費）　　2,500,000円、（間接経費）　　750,000円

研究成果の概要（和文）：プログラムの完全な正しさを保証するには、プログラムの論理自体を機械上に証明するしかない。この研究では，OCamlと同等の機能をもった型検証器を完全に証明し，正しさを証明されたプログラミング言語の実装の中心に据えることを目指した。具体的には、Aydemir等のEngineering Metatheoryという手法に基づいてOCaml特有の構造的多相性（オブジェクトおよび多相ヴァリアント）を形式化し、型システムの健全性、型推論の健全性と完全性、スタックベースのインタープリターの正しさを形式的に証明した。さらに副作用のための「緩和された値限定多相性」や再帰的型定義の検証の一部も証明した。

研究成果の概要（英文）：The only way to be completely certain of the correctness of a program is to prove it formally using a mechanical theorem prover. In this research, I aimed at formally proving the correctness of a type checker for a part of the OCaml functional programming language, to include it in a fully certified interpreter.
Concretely, based on the Engineering Metatheory approach, I formalized a small language including structural polymorphism, which allows to type OCaml objects and polymorphic variants, and proved its type soundness, the soundness and completeness of its type inference, and the correctness of its evaluation through a stack-based interpreter. Moreover, we obtained some concrete proofs concerning the so-called "relaxed value restriction" required for the typing of side-effects, and the coherence check for structural recursive type definitions.

研究分野：　13

科研費の分科・細目：　1002

キーワード：　型推論　アルゴリズムの証明　モジュールの証明　国際情報交換

１．研究開始当初の背景
研究開始当初は、長年携わって来たOCamlという関数型プログラミング言語の型システムの実装に段々不信を募らせていた。OCamlの型システムは複雑で、独自の機能も多い。各機能についてそれぞれ論文で仕様と証明が与えられているが、コンピューターが検証した形式的なものではなく、漏れの可能性が十分あった。また、機能同士の干渉も調べ尽くされていなかった。実際、理論的な細かい穴や実装のバグが次々と発見され、その度に修正を行っていた。
そういうバグを防ぐために、プロフラミング言語の理論と処理系の検証が既にあった。理論に関して、最も進んでいたのはStandard MLのTwelfによる形式化であった。そこで扱われていたのは、Core MLとモジュールシステムの型システムのみであったが、それでも巨大な証明になる。OCamlの型システムはそれよりさらに機能が多く、新たな研究が求められていた。処理系に関しては、CompCertでCのコンパイラの完全証明が行われていた。ただし、CとML/OCamlで言語の形が大きく違う。

２．研究の目的
プログラムからバグを排除するのに様々な手法があるが，完全な正しさを保証するには，プログラムの論理自体を機械上に証明するしかない．
この研究の目的は，進んだ型システムをもったプログラミング言語OCamlの型検証器を完全に証明し，正しさを証明されたプログラミング言語の実装の中心に据えることであった．
本質的な複雑さおよび正しさの重要性から，型検証器は「全体証明」とも言えるこの方法論の代表例になれるという意識もあった．

３．研究の方法
方法として、完全に証明されたOCamlのインタープリターの新たな実装を目指した。現在の実装が複雑過ぎて、それ自体を証明するのが不可能という認識から来る判断であった。具体的には、型検証器定理証明支援系Coqで型検証器を含むインタープリターを書き、静的意味（型付け）および動的意味の完全な検証を目指した。
インタープリターがプログラムを実行するまでの流れを考えると，まず，入力されたプログラムをパーザで分析する。次に，型検証器が入力されたプログラムの型情報を再構築し，型システムに準拠していることを確認する．最後に抽象機械で型の付いたプログラムを実行する．
最初のパーザは理論が今や確立されており，パーザは手で書くのではなく，仕様から生成されるのが一般的なので，この研究の対象とはしなかった．
しかし，次に型検証器が仕様通りに動いていること，すなわち型推論の健全性と完全性を証明しなければならなかった．
最後の実行について、実行が安全であることを証明する為に，型健全性の証明が利用できるが，その証明で扱われている実行はプログラム全体への代入など，簡単に実装できない方式をとっている．効率の良い抽象機械にするために，改めてその機械での実行が安全であること，あるいは挙動が言語の定義に沿っていることも証明しなければならなかった．
インタープリターに含む機能として、Core MLに加えて、OCaml 特有の構造的多相性（オブジェクトおよび多相ヴァリアント）と構造的サブタイピングを扱おうとした。また、時間の許す限り、可変なデータの型付けやモジュールシステムなど、他の進んだ機能も対応する予定であった。

４．研究成果
当初の計画がかなり野心的だったが、機能を加えることに連れて、Coq での証明が大きくなり、結局全てを入れることはできなかった。最初に行ったいわゆるCore MLと構造的多相性を含めた言語の実装はうまく行った。Aydemir等のEngineering Metatheoryという手法に基づいて言語を形式化し、型システムの健全性、型推論の健全性と完全性、スタックベースのインタープリターの正しさが形式的に証明できた。実装は完全にCoqで行われ、Coq の抽出機能によって型検証器を含むインタープリターがOCamlのプログラムとして得られる。元のOCamlの実装と違い、こちらは正しさが完全に保証されている。
ここで注目するべき点が二つある。まず、構造的多相性については既に論文で型システムの健全性および型推論の中心である型の単一化の証明を与えていたが、型推論の健全性・完全性は未着手だった。理論的な飛躍がないものの、形式化と同時にその証明を行う必要があった。また、構造的多相性は通常のMLの同型再帰型と違い、それより強い同値再帰型を提供している。この証明は同値再帰型の最初の形式化となったはずである。
第２ステップとしてはまず、可変なデータの型付けを組み込もうとした。こちらに関しては2004年に私が論文を発表し、OCamlの処理系の中で実装している「緩和された値限定多相性」を採用した。「緩和された値限定多相性」は通常の ML 系言語に比べて柔軟性があるが、その証明が「意味的な型」を利用するのが特徴的である。フランスから修士課程の研修で来たENS Lyon の Thomas Leventis 氏と一緒に証明方針を考え、それをCoqに移した。途中結果として、その理論的な基礎であるFrancois Pottierが提案したBT(X)という型体系の健全性の証明も得られた。
また、タリン工科大学の中田景子氏と一緒に、再帰型定義の整礎性・正則性を確認するアルゴリズムの形式化を行った。構造的多相性では既に再帰型が扱われていたが、型定義と組み合わせて使うと、実際の型が型定義を展開

する形で作られるので、正しい構造であるかどうか展開時に調べる必要がある。しかも、チェックの中でも展開が行われるので、かなり複雑なアルゴリズムになる。具体的には、メモリ・グラフの上で定義されている OCaml のアルゴリズムを Coq で明示的なグラフ構造で実装し、証明を始めた。型の完全な展開の停止性および展開がグラフの性質を崩さないことが証明されている。
モジュールシステムに関して、同じく中田氏と一緒に研究して来たパスによる再帰モジュールの形式化の一部を Coq で実装した。その証明には非常に追いにくい帰納法が使われており、Coq での実装により証明の信頼度が大きく上がったと言える。本来、このパスによる手法は OCaml のモジュールの形式化とは異なっていたが、最近になって OCaml のモジュールでもパスを使うようになったので、将来的にはこの手法で OCaml のモジュールシステムの正しさも証明できるようになるだろう。
この Coq での形式化と並行して、OCaml の型システムの開発も止まっていない。特に、この４年間では OCaml に一般化代数的データ型(GADT)を追加した。GADT は Coq などにある依存型の一部の機能を普通の関数型プログラミング言語で導入するための機構である。これにより、データ構造の様々な不変量が表現できるようになり、より安全なプログラムが書けるようになる。まず、Jacques Le Normand と一緒に基本的な実装を行った。型の安全性は保存されたが、型推論の完全性が損なわれていた。これは GADT を導入した他のプログラミング言語とは共通の課題であった。続いて、Didier Rémy と一緒に、型推論の完全性を取り戻す方法を考案した。これも OCaml で実装され、論文も発表された。
残念ながら、研究期間中に GADT の形式化を始める時間がなかった。Didier Rémy との共著論文で使われているいる枠組みが構造的多相性のものに近いので、可能なはずである。今回も型推論の健全性・完全性がかなり込み入ったものになっているので、形式化のありがたみは格別であろう。

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 2 件）

Jacques Garrigue, Keiko Nakata : Path resolution for recursive nested modules. Higher-Order and Symbolic Computation (2012) 24:207-237.
DOI 10.1007/s10990-012-9083-6.
Jacques Garrigue : A certified implementation of ML with structural polymorphism and recursive types. Mathematical Structures in Computer Science, to appear.

〔学会発表〕（計 11 件）

査読論文あり
Jacques Garrigue : A certified implementation of ML with structural polymorphism. In 8th Asian Symposium on Programming Languages and Systems, Shanghai, November 2010. Springer-Verlag LNCS 6461, pages 360-375.
DOI 10.1007/978-3-642-17164-2_25
Hyonseung Im, Keiko Nakata, Jacques Garrigue and Sungwoo Park : A syntactic type system for recursive modules.In ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, Portland, Oregon, October 2011.
DOI 10.1145/2048066.2048141
Jacques Garrigue and Didier Rémy : Ambivalent types for principal type inference with GADTs. In {\em 11th Asian Symposium on Programming Languages and Systems}, Melbourne, December 2013. Springer-Verlag LNCS 8301, pages 257-272.
DOI 10.1007/978-3-319-03542-0_19

査読論文なし
Jacques Garrigue and Alain Frisch : First-class modules and composable signatures in Objective Caml 3.12.
ACM SIGPLAN Workshop on ML, Baltimore, 2010.
Jacques Garrigue and Jacques Le Normand : Adding GADTs to OCaml: a direct approach. ACM SIGPLAN Workshop on ML}, Tokyo, 2011.
Jacques Garrigue and Didier Rémy : Tracing ambiguity in GADT type inference. ACM SIGPLAN Workshop on ML, Copenhagen, 2012.
Jacques Garrigue : On variance, injectivity, and abstraction. OCaml Workshop, Boston, 2013.
Jacques Garrigue and Grégoire Henry : Runtime types in OCaml. OCaml Workshop, Boston, 2013.
Jacques Garrigue and Pierre-Marie Pédrot, Simpoulet: an attempt at proving environmental bisimulations in Coq.
第 6 回定理証明および定理証明系に関する研究集会，名古屋大学，2010.
Jacques Garrigue : More Logic More Types. ML Nagoya，名古屋ルーセントタワー，2012.
Jacques Garrigue and Thomas Leventis, Avoiding binders: rooted recursive modules and semantic polymorphism. 第 8 回定理証明および定理証明系に関する研究集会，千葉大学，2012.

〔図書〕（計 0 件）

〔産業財産権〕

○出願状況（計０件）
○取得状況（計０件）

〔その他〕
ホームページ等
http://www.math.nagoya-u.ac.jp/~garrigue/

６．研究組織
(1)研究代表者
ジャック・ガリグ（Jacques GARRIGUE）
名古屋大学・多元数理科学研究科・准教授
研究者番号：80273530