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 ２７ 年　 ５ 月 １９ 日現在

機関番号：１１３０１
研究種目：若手研究(B)
研究期間：2012～2014
課題番号：２４７０００２１

研究課題名（和文）高水準かつ安全なプログラミング言語間連携機構の実現

研究課題名（英文）A study on high-level and safe interoperability of programming languages

研究代表者
上野　雄大（Ueno, Katsuhiro）

東北大学・電気通信研究所・助教

研究者番号：６０５５１５５４

交付決定額（研究期間全体）：（直接経費）　　2,100,000円

研究成果の概要（和文）：複数のプログラミング言語を連携してひとつのアプリケーションを安全に開発できることを目指し，本研究では，言語間連携のための正しいコードをコンパイラが型から自動的に導出することで，型付き関数型言語からそれ以外の言語を直接かつ安全に利用可能にする方式を開発した．その方式に基づき，関数型言語SML#にシェルスクリプトをそのまま埋め込む拡張を実装した．さらにRubyとの直接連携を目指し，Rubyの高水準な形式的意味論を与える基礎理論を構築した．

研究成果の概要（英文）：Towards constructing a secure application by combining multiple programming languages, this reserch has developed a new approach for a strongly typed functional language to interoperate with an untyped language by generating a secure glue code between the two languages from type annotations of shell scripts. This research also has implemented an extension to SML#, a variant of Standard ML, that allows the programmer to embed shell scripts directly in an ML program. In addition, towards interoperability with Ruby, this research has presented a formal operational semantics that can serve as a high-level specification of Ruby.

研究分野：プログラミング言語

キーワード：プログラム言語論　プログラミングパラダイム　コンパイラ　関数型言語

１．研究開始当初の背景

少ない労力で高機能なプログラムを構築するための手段のひとつとして，または大規模なデータを容易に統合・分析するための方法として，複数のプログラミング言語で書かれたモジュールを結合してひとつのアプリケーションを構成することは，今日の多くの計算機利用の局面において行われている．例えば，Web アプリケーションの開発では，Ruby などのスクリプト言語でアプリケーションロジックを記述し，データベースへの問い合わせは SQL で行い，ユーザーインターフェース部分は Ajax などの技術に基づき XML と Javascript を用いて記述する．マッシュアップやクラウドコンピューティング，またはクラスタを用いた並列計算などを含むような，より大規模かつ有機的な構造を持つアプリケーションでは，そのアプリケーション全体を記述する言語の数はさらに増加する．高度な計算機の活用には，様々なプログラミング言語で書かれたモジュールを連携・統合する技術は必須と言える．

しかし残念ながら，今日行われている言語間の連携は，その連携の正しさが保証された形で行われていると言える状況には至っていない．これらの連携の多くは，データやプログラムを構造の無い文字列として扱うことで実現されている．文字列による連携は自由度が高く，各言語の機能を最大限に活用したプログラミングが可能である．しかしながら，その自由度の高さから，その連携の正しさを保証することは難しい．これは，たとえ記述言語が安全であったとしても，言語間連携を行う箇所で致命的な欠陥やセキュリティー上の問題が生じる可能性があることを意味する．適切なエスケープ処理を行なっていないことによる SQL インジェクションやクロスサイトスクリプティングなどの脆弱性は，この問題の典型的な例である．一方，この問題を系統的に解決するための手段として，連携先の言語の機能を連携元の言語に埋め込む，または連携元の言語を連携先の言語に変換するなどのアプローチが提案されている．例えば代表的な成果に，Heller らによる Caml-Shcaml や Cooper らによる Links などがある．しかし，埋め込みや変換に基づく連携は，埋め込み先の言語で記述可能な範囲に限定され，２つの言語の特長を最大限に活用したプログラムを記述することは難しく，連携のメリットが得にくい．その結果，今日のプログラミング環境では，依然，自由度は高いが脆弱な文字列ベースの連携が主流となっているのが現状である．この状況は，ソフトウェアが担う社会的役割の大きさに対してあまりに脆弱と言える．

２．研究の目的

本研究では，この課題に対し，連携する２つの言語の性質を型理論やプログラムの意味論に基づいて分析し，二者の違いを形式的に明確にすることで，言語の違いを吸収するコードをコンパイラが系統的に導出する，というアプローチを提案する．このアプローチでは，言語を変更・制限・翻訳することなく，かつ正しい方法で言語間の連携を実現できる．また，言語を変更しないため，連携対象の言語の全機能をそのまま活用することができる．さらに，連携に必要なコードは，理論的背景に基づく系統的な方法でコンパイラが自動生成するため，その構成上正しいことが保証される．連携コードの自動生成は，プログラマを言語間の差異や煩雑な文字列操作から開放し，従来よりも信頼性の高いアプリケーションをより簡単に記述することをも可能とする．

本研究の一般的な目標は，以上のアプローチの実現可能性を追究し，任意の２つのプログラミング言語に対して，型理論や意味論などの分析を通じて，二者の違いを吸収するコードを系統的に導出するための方法論を確立することである．申請者による先行研究によってすでに得られている，本研究の基礎となる洞察は以下の通りである．

・プログラミング言語の種類を問わず，モジュールのインターフェースにのみ注目するならば，関数型言語の型システムを用いて，そのモジュールの型付けや，ある種の型安全性を考えることができる．

・型の無い言語におけるある種の多相的な振る舞いは，関数型言語のパラメトリック多相性の理論で記述できる場合がある．

・以上の洞察に基づいて連携対象のモジュールに型を付けることができるならば，型主導コンパイルの考え方に基づいたコンパイルアルゴリズムによって，言語間の違いを吸収するコードを系統的に生成することができる．

本研究では，以上の洞察に基づき，すでに理論研究が広く行われており，その理論的な取り扱いが確立している静的型付き関数型言語を核として，関数型言語と他言語の系統的な連携の実現を目指す．さらに，本研究の一般的な目標の達成に向けた有用な知見を得るためには，連携対象の言語として，型付き関数型言語から遠い言語，例えば強い型付けを持たず，かつオブジェクト指向など関数型でないパラダイムに基づいて設計された言語を考えるべきである．また，本研究の実用的側面をアピールするためには，実用アプリケーションの記述に広く用いられている言語との連携を考えるのが好ましい．そこで本研究では，既存の実用スクリプト言語を関数型言語の型システムの考え方に基づき分析し，関数型言語との系統的連携を可能とするコンパイル方式とその性質を明らかにすることを目指す．

３．研究の方法

本研究課題の目標である高度な言語間連携機構を実現するには，その理論上の性質を

明らかにすることに加え，この連携機構を有するコンパイラを作成するための実装技術の確立や，実例を通じた評価が必須である．そこで，本研究では，連携対象の実用言語を具体的にいくつか選び，連携機構を実用関数型言語のコンパイラに実装し，実例を通じて連携機構の利便性を評価する．

本研究では，まず，連携対象の言語としてシェルを選択する．言語としてのシェルは取り扱うデータ型や言語機能が比較的小さいため，本研究の最初の対象として妥当である．提案するアプローチによるシェルとの連携を達成するために取り組むべき課題は以下の３点である．

(1) シェルスクリプトのインターフェースの決定とその型付け．型付き関数型言語では，処理内容やデータ形式が型として表現されており，型の整合性を検査することでプログラムのある種の正しさを保証できる．一方，シェルスクリプトはほとんどのデータをテキストとしてやり取りする自由な記述が特長であり，これに無理に型を付けることは，その長所を殺してしまいかねない．本研究では，シェルスクリプト全体の入出力では型が付くような構造を持つデータを扱っているとみなすことで，この課題の克服を試みる．

(2) シェルスクリプトと関数型言語の連携を実現する実装手法と意味論．シェルスクリプトを関数型言語から呼び出し実行するためには，関数型言語から呼び出されるシェルスクリプトの評価モデルが必要である．これら２つの言語の評価モデルを繋げるためには，ML の値とシェルスクリプトの入出力文字列を相互に変換するコードが必要になる．本研究では，このコードを型情報から系統的に導出する枠組みを構築し，安全かつ高水準な言語間連携の確立を目指す．

(3) 型主導コンパイルアルゴリズムの構築と実装．(1)および(2)を自動的に実現するコンパイルアルゴリズムを，型情報をコードにコンパイルする方式として構築し，コンパイラに実装する．

次に，シェルとの連携機構の実現で得られた知見を応用し，より高度な連携機構を必要とする汎用プログラミング言語と関数型言語との連携を目指す．本研究では，より高度な連携対象の実用言語として Ruby を選択する．Ruby は，純粋なオブジェクト指向言語として設計されたスクリプト言語であり，関数型言語とは多くの点において直交する特徴を持つ．二者を高水準に連携することができるならば，提案する言語連携技術は，様々な種類の言語を系統的に結合する基盤技術として応用できる可能性がある．一方，シェルよりも大きく複雑な Ruby 言語との連携には，Ruby の言語機能や意味論を詳細に理解し，Ruby 言語への型注釈の与え方を緻密に設計する必要がある．本研究では，以上の調査および検討を行い，シェルとの連携技術を洗練し，実用レベルの関数型言語との連携を実現することを目指す．

本研究を通じて構築された全ての方式は，SML#コンパイラに実装し，その実用性を検証する．実用性の検証には，Web アプリケーションなど複雑なアプリケーションの構築を試みることによって行う．そのため，本研究では，高度な言語間連携機構の実現に加え，SML#コンパイラの実装技術や，関数型言語によるアプリケーション構築技術に関する研究も併せて行う．

４．研究成果

まず，関数型言語から高水準にシェルスクリプトを呼び出す方式を構築し，関数型言語 SML#コンパイラに対する予備的な実装を行った．本連携方式は，シェルスクリプトに型注釈を与えた式 _sh s _end : π を SML#言語に加えることで，SML#プログラム中にシェルスクリプトをそのまま埋め込めるようにしたものである．SML#プログラム中に含まれるシェルスクリプトには，そのスクリプトの入出力データの形式を SML#の型注釈として明示する．SML#プログラムからは，このシェルスクリプトはその型注釈で与えられた型の関数として見える．従って，型が整合する限り，シェルスクリプトを SML#プログラムと自由に組み合わせることができる．言語間連携の正しさは，シェルスクリプトに正しく型注釈を与えるかどうかで決まる．もしプログラマが正しい型注釈を書いたならば，シェルスクリプトとの連携を含めた SML#プログラム全体は型安全である．SML#コンパイラは，この型注釈から，連携に必要となるデータ変換コードを自動生成する．この連携機構を用いたプログラム例を以下に示す．

```
val users = _sh
      who | cut -f1 -d\  | sort | uniq
    _end : unit -> string list;
val diskUsage = _sh
      du -sm "/home/$1"
    _end : string -> unit -> int * string;
map (fn i => (i, #1 (diskUsage i)))
    (users ());
```

例に見られるように，この連携機構は，SML#およびシェルの両方の言語に変更を加えない．従って，従来の文字列ベースの連携方式と同様に，言語の記述力を全く制限しない．一方，従来の連携方式とは異なり，型注釈の正しさをユーザーが保証するならば，連携機構の整合性は自動的に保証される．これらの性質は，本研究の目的である高水準かつ安全な連携機構が目指す望ましい性質である．従って，SML#とシェルの組み合わせにおいては，本研究の目指す連携機構は達成できたと言える．

次に，シェルとの連携で得た知見に基づき，型付き関数型言語と Ruby の高度な連携機構の実現を試みた．この機構の実現を遂行するための最初の課題は，Ruby 言語機能とその意味を系統的に理解することである．この課題

の達成のため，JIS/ISO 標準として定められている Ruby の言語仕様の分析を通じ，Ruby の意味論の理解を試みた．しかし，Ruby は高度な言語機構が相互に依存する複雑な言語であり，JIS/ISO 標準に定められている形式では，本研究の目的である言語間連携に必要となるだけの十分な Ruby 言語の理解を得ることが困難であった．そこで本研究では，Ruby に関するより本質的な理解を得るために，相互依存する言語機能の意味規則をオラクルとして抽象化することで，Ruby の複雑な言語仕様を言語機能ごとに分割し，各機能を独立した計算系として定義することで，より系統的に理解可能な Ruby の意味論を与える手法を開発した．この研究成果は，高度な言語連携のための基礎を与えるだけでなく，実用プログラミング言語の開発における盤石な言語設計・実装基盤の実現にも発展し得るものである．しかし残念ながら，当初の目標であった Ruby との言語連携機構の開発は，研究期間内に達成することはできなかった．当初の目標は達成できなかったものの，将来の実用プログラミング言語の開発に影響を与え得る新たな知見を得ることができたことは，本研究の大きな成果の一つである．

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１件）

①遠藤誠典, 百足勇人, 森畑明昌, 上野雄大, 大堀淳．変数参照関係を用いた関数型プログラムのコードリーディング支援．コンピュータソフトウェア, 32(1), pp.1_194-212, 2015. 査読あり

〔学会発表〕（計１４件）

① Katsuhiro Ueno, Yutaka Fukasawa, Akimasa Morihata, Atsushi Ohori. The Essence of Ruby. In: Programming Languages and Systems, 12th Asian Symposium, APLAS 2014, Lecture Notes in Computer Science, Vol. 8858, pp. 78-98, National University of Singapore, 2014 年 11 月 17 日.
②逢坂美冬, 佐々木智啓, Charles Mejia Cruz, 上野雄大, 大堀淳．関数型言語 SML#における 64 ビット対応への取り組み．日本ソフトウェア科学会第 31 回大会，名古屋大学，2014 年 9 月 10 日.
③ Katsuhiro Ueno, Atsushi Ohori. Compiling SML# with LLVM: a Challenge of Implementing ML on a Common Compiler Infrastructure. In ACM SIGPLAN ML Family Workshop 2014, Gothenburg, Sweden, 2014 年 9 月 4 日.
④Atsushi Ohori, Katsuhiro Ueno, Kazunori Hoshi, Shinji Nozaki, Takashi Sato, Tasuku Makabe, Yuki Ito. SML# in Industry: A Practical ERP System Development. In Proceedings of The 19th ACM SIGPLAN International Conference on Functional Programming, Gothenburg, Sweden, 2014 年 9 月 2 日.
⑤Katsuhiro Ueno, Atsushi Ohori. A foreign language interface from ML to shell. In pre-proceeding of the Symposium on Trends in Functional Programming 2014 (TFP 2014), Soesterberg, Netherlands, 2014 年 5 月 26 日.
⑥斎藤皓, 上野雄大, 森畑明昌, 大堀淳. SML# の SQL 統合機能への行集約機能の実装. 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，熊本県阿蘇市，2014 年 3 月 5 日.
⑦上野雄大，大堀淳．関数型言語からキーバリューストアへの型安全なアクセス機構．日本ソフトウェア科学会第 30 回大会，東京大学，2013 年 9 月 11 日.
⑧遠藤誠典, 百足勇人, 森畑明昌, 上野雄大, 大堀淳．変数参照関係を用いた関数型プログラムのコードリーディング支援．日本ソフトウェア科学会第 30 回大会，東京大学，2013 年 9 月 11 日.
⑨藤井貴啓，上野雄大，森畑明昌，大堀淳．SML#のデータベース連携機能を活用したウェブアプリケーション構築技術．第 15 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，福島県会津若松市，2013 年 3 月 5 日.
⑩相澤遥也, 徳田亮平, 上野雄大, 森畑明昌, 大堀淳. ヘッダーファイルから SML#への C 関数の自動インポート．第 15 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（ポスター・デモセッション），福島県会津若松市，2013 年 3 月 5 日.
⑪深澤優鷹，上野雄大，森畑明昌，大堀淳．Ruby の操作的意味論の形式的定義に向けて．第 15 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（ポスターセッション），福島県会津若松市，2013 年 3 月 5 日.
⑫斎藤皓, 上野雄大, 森畑明昌, 大堀淳. SML# の SQL 統合への group by の導入．第 15 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（ポスターセッション），福島県会津若松市，2013 年 3 月 5 日.
⑬朝井雄大，上野雄大，森畑明昌，大堀淳．C との連携機能を持つ関数型言語におけるプロファイラの試作．第 15 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，福島県会津若松市，2013 年 3 月 4 日.
⑭藤井貴啓，上野雄大，大堀淳．SML#を用いた Web アプリケーションの試作 - 関数型言語による Web とデータベースのシームレスな連携に向けて. 日本ソフトウェア科学会第 29 回大会（デモ・ポスターセッション），法政大学，2012 年 8 月 23 日.

〔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〕

６．研究組織
(1)研究代表者
上野　雄大（UENO KATSUHIRO）
東北大学・電気通信研究所・助教
研究者番号：60551554