様式 C-19

科学研究費補助金研究成果報告書

平成　２３年　６月　１日現在

機関番号：３２６１９
研究種目：若手研究（Ｂ）
研究期間：2007 ～ 2010
課題番号：１９７００００２１
研究課題名（和文）　プログラム変換に基づく対話的プログラム開発システム

研究課題名（英文）　Interactive program development system based on program transformation

研究代表者
篠埜　功（SASANO ISAO）
芝浦工業大学・工学部情報工学科・助教
研究者番号：１０３６２０２１

研究成果の概要（和文）：関数型言語によるプログラム開発をプログラム変換に基づき支援する環境の構築を目指し、基本的な支援機能である変数名補完機能について、基本方式の提案および実装を行った。プログラムは最初から順に記述されていくという前提で、暗に型付けられた関数型言語を対象とした変数名補完問題を定義し、それを解くアルゴリズムを提案した。提案したアルゴリズムは､候補となるべき変数のみが全て列挙されるという望ましい性質を持つ。提案手法に基づき核言語を対象として変数名補完を行う Emacs モードを実装し、その有効性を確認した。

研究成果の概要（英文）：We aimed at developing integrated development systems for implicitly typed functional languages based on program transformation techniques. With investigating the current status of the systems, we have developed basic mechanism of variable completion systems. As a first step toward developing practical systems, we considered a simple case: up to the cursor position the program text is given completely. With this assumption we specified a variable completion problem for an implicitly typed core functional language with let polymorphism and then developed an algorithm for solving the problem. The algorithm has a desirable property that all and only the variables that should be candidates are computed as the candidates. Based on the algorithm we have implemented a variable completion system for the language as an Emacs mode.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２００７年度	900,000	0	900,000
２００８年度	600,000	180,000	780,000
２００９年度	600,000	180,000	780,000
２０１０年度	600,000	180,000	780,000
年度			
総　計	2,700,000	540,000	3,240,000

研究分野：プログラミング言語
科研費の分科・細目：情報学・ソフトウェア
キーワード：プログラム開発システム、プログラム変換、変数名補完

１．研究開始当初の背景

プログラム変換は古くから研究されており、仕様からのプログラムの導出、コンパイラにおける最適化などにおいて有用であることが知られている。プログラム変換とは、プログラムに対して変換規則を適用することにより、意味を変えることなく別のプログラムを得ることである。正しいが効率は良いとは限らない素朴なプログラムから出発し、変換規則を適用することにより、正しくかつ効率

の良いプログラムを得られれば、仕様から効率のよいプログラムが得られたことになる。変換規則としては、関数の展開、畳み込み、具体化、一般化といった基本的なものや、それらを組み合わせて得られる規則、また、関数融合、組化等を行うための、より複雑な規則までさまざまである。これまで、人間の洞察を用い、人手を介することにより広範囲のさまざまな問題に対する効率の良いプログラムの導出の研究が行われており、またプログラミング言語のコンパイラに融合変換を組み入れる研究など活発に行われてきている。コンパイラに入れるプログラム変換は有用であるが、すべてのプログラムに一様に適用されるため、効率が悪くならないことを保証するために変換機能が限定されてしまうという一面がある。本研究においては、プログラミング時に、プログラム変換を知っているプログラマが紙の上で行うことをコンピュータ上で行えるように支援することを考える。

２．研究の目的

本研究では、プログラム変換に基づく対話的プログラム開発システムの作成を目的とする。

３．研究の方法

(1) 既存の開発環境の調査

プログラム変換に適していると考えられる関数型言語を対象として開発環境の調査を行った。Haskell, Standard ML, OCaml 等の代表的型付き関数型言語の開発環境では、通常 Java 等の言語の開発環境に備わっている変数名補完機能が提供されておらず、この機能の整備をする必要があると考えた。

統合開発環境(以下では IDE と略記) は大規模なソフトウェア開発において重要な役割を担っている。IDE は自動字下げやキーワードへの色付け、変数名補完などの機能を提供する。変数名補完はそれらの中でも最も基本的で便利な機能のひとつである。変数名補完とは変数名を入力する時に入力中の文字列を接頭辞に持つ変数名をポップアップウィンドウ等に表示し、その中からプログラマが選んだ変数名を自動で入力する機能である。大規模なプログラムの開発においてはプログラムを読みやすくするために長い変数名を用いることがよくあり、そのような場合に変数名補完により変数名を思い出す時間や入力の時間、綴りミスを削減することが出来る。広く使われている C や C++、Java などの言語の IDE においては様々な機能が提供されているが、関数型言語については十分な機能が提供されていない状況にある。関数型言語の利用者は近年徐々に増えつつあり、十分な機能を備えた関数型言語用の IDE の開発が望まれている。

これまでに変数名補完機能を備えた統合開発環境は多く開発されてきている。それらのうちのいくつかは編集中のファイルに入力された単語に基づいた簡易な変数名補完機能を備えている。他のモジュールやクラス等の中で定義された識別子に基づいて変数名補完を行うものもある。Visual Studio に搭載されている Intellisense や、Eclipse の content assist、vim の omni completion のようにさらに高度な補完を行うものもある。それらは編集中のプログラムの文脈に合うものを候補として表示する。例えば intellisense と content assist は変数のスコープを考慮している。文脈を考慮することで変数名補完はより便利になってきてはいるが、我々の知る限り型推論がある言語において候補の絞込みに型情報を用いるものはこれまでに存在しない。

(2) 変数名補完システムの基本方式の提案および実装

変数の型情報を得るためには型推論を行う必要があり、不完全なプログラムテキストに対して型推論を行うには様々な方法が考えられるが、我々はプログラムの先頭からカーソル位置までが完全に与えられているという仮定のもとで考えた。つまり、変数名補完の候補の計算にカーソル以前のテキストのみを利用し、カーソルより後ろのテキストは利用しない。ML 系の言語では（(相互）再帰関数の宣言を除いて）全ての変数は使用される位置よりも前に宣言される為、この仮定の下でも補完候補となる変数の情報は得ることが出来る。本研究ではこの仮定のもとで補完方式を考案する。候補となるべき変数が候補に入っていないことがあると、表示される候補の他に候補が存在する可能性を考えなければならず、そのような変数名補完システムは不便である。その為、変数名補完システムは候補となるべき変数は全て表示されるという性質（完全性）を持っていることが強く望まれる。また、選択された候補によって補完した場合に型エラーが起きるような候補は含まないという性質（健全性）も望まれる。我々の先行研究で提案した素朴なアルゴリズムは健全性は備えていたが、完全性は備えていなかった。本研究では、連続する関数適用を抽象化するマーク式を導入することにより健全性と完全性を備えたアルゴリズムを構築する。さらに、我々のアルゴリズムはカーソル位置で有効な変数の型を適切に抽象化することで冗長な計算を減らしている。

①変数名補完問題の定義

変数名補完問題は、let 多相のλ式を対象とし

て行い、カーソル位置以降の情報は用いずに補完候補を計算するという前提で定式化した。具体的には、カーソル位置以降に何か適切なプログラムを補うことにより型が付くような補完候補を求める問題として変数名補完問題を定義した。

②素朴な手法
カーソル位置以降に補えるプログラムは無数に存在するため、考慮すべきプログラムを有限個に抑える必要がある。まず、素朴な考えとして、ダミー式を導入することが考えられる。これにより考慮すべきプログラムは減少する。この素朴な考えのもとで補完システムを試験的に実装したが、ダミー式の導入のみでは考慮すべきプログラムがまだ無限にあるため、構文木の深さに関して閾値を設けることにより有限に抑え、実装を行った。この素朴な手法では、計算時間が膨大にかかる場合があり、また必ずしもすべての望む候補が提示されるとは限らないという問題があった。

③開発した手法
素朴な手法では閾値以下の式を全部生成するため計算量が大きくなっていた。また、閾値があるためにすべての可能性が考慮されず、候補となるべき変数が候補にならない場合があった。これらを解決するため、ダミー式を引数に取る 0 回以上の関数適用を表すマーク式を導入する。これにより、無限個の式を生成せずに全ての候補を得ることができる。任意の式は（構文上は）引数を取ることが出来る為、素朴な手法ではカーソル式を含んでいる全ての部分式にダミー式を引数として補っていた。結果としてダミー式を引数にとる関数適用が連続する式が生成されることになるが、このような関数適用の連続をマーク式によって抽象化できるということが提案手法の考え方である。これら、ダミー式、マーク式、および現在入力中の変数に対応するカーソル式を用いてカーソル位置以降を補うことにより、一つの抽象化されたプログラムを得ることができ、これに対して型推論を行う。型推論アルゴリズムは Milner の考案したアルゴリズム W をもとに構築した。この型推論アルゴリズムにより、カーソル位置の型および型環境が得られる。これをもとに、候補の絞り込みを行う。絞り込む前の候補はカーソル位置の型環境 Γ の定義域にある変数全てとする。それら候補をカーソル式の型と単一化を行うことにより絞り込む。どのカーソル式の型とも単一化が成功しなかった変数は候補から取り除かれる。候補の変数の型が τ である時は、それぞれのカーソル式の型と τ の単一化を行い、変数の型が $\forall\, \alpha_1 \ldots \alpha_k.\, \tau$ という多相型である場合には、具体化した型 $[\beta_1/\alpha_1] \ldots [\beta_k/\alpha_k]\, \tau$ ($\beta_1, \ldots, \beta_k$ fresh)とそれぞれのカーソル式の型で単一化を行う。
このアルゴリズムは、得られた変数集合が問題の条件を満たすという性質（健全性）および、問題の条件を満たす変数はすべてアルゴリズムによって得られる変数集合に含まれるという性質（完全性）を満たす。これら 2 つの性質はまだ証明していないが、具体例で試した限りにおいてこの 2 つの性質に反する例は見つかっていない。健全性は、型の制約に反する変数は全て候補から取り除かれることを保証する。この性質により、型の制約を考えない場合に比べて大幅に候補の数を減らすことができる。候補に漏れがあると不便であるため、候補となるべき変数はすべて候補になるということを保証する完全性は変数名補完においてとても重要な性質である。これらの 2 つの性質により変数名補完システムは実用に適したものとなる。
このアルゴリズムの計算量については、マーク式の型を推論するときに関数型のネストの分だけ計算量が増えるが、通常は関数型のネストはあまり深くならない。マーク式のネストの深さに関して指数的に計算量が増えるが、実際のプログラムではネストが深くなることはあまり無く、ネストの深さは実質的に定数と見なせる。さらにカーソル式の型を推論する場合において型を抽象化することで計算量を抑えている。

④実装
提案した変数名補完方式は Emacs というエディタのモードとして、Emacs lisp を用いて実装し、効果を確認した。図 1 にそのスクリーンショットを示す。

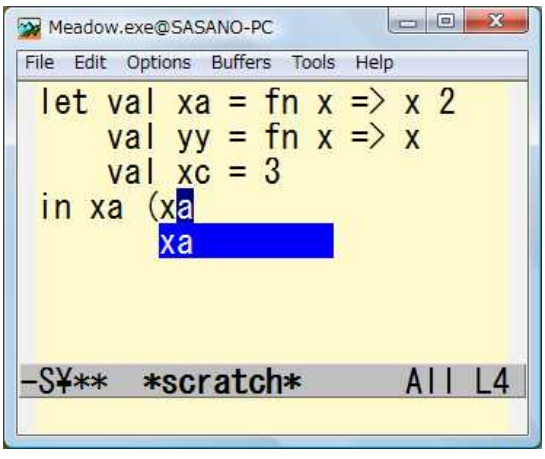

図 1: 変数名補完システム

実装においては、字句解析、構文解析を行う必要がある。通常のプログラムの解析と異なるのは、カーソル位置までのプログラムの解

析をしなければならない点である。プログラムの先頭からカーソル位置まで解析し、途中段階の構文木が出来、それに対してカーソル式、ダミー式、マーク式を用いてカーソル位置以降のプログラムを補う。その後、型推論、候補の絞り込みを行い、プログラマに候補を提示する。実行速度は、対象の核言語に対しては十分に実用になる速度が得られている。

４．研究成果
関数型言語によるプログラム開発をプログラム変換に基づき支援する環境の構築を目指し、let 多相の暗に型づけられた関数型言語に対する変数名補完問題の定式化およびアルゴリズムの考案、実装を行った。このアルゴリズムはプログラマが期待する候補をすべて提示し、かつ必ず型エラーになる候補は提示しないという最も望ましい性質を持つ。この結果は、今後暗に型付けられた関数型言語に対する開発環境を開発する際に最も基本的な機能の実現に関する基盤を与えるものである。

①国内外における位置づけとインパクト
国内では、プログラミング言語研究者の集まる国内会議で研究成果を発表し、有用性は認められている。今後、実際に使われる言語上で実装することにより、広く認知され、実際に使用されるようになることが期待される。国外ではまだ発表していないが、今後、国際会議等で研究成果を発表することを考えている。現在のところ、変数名補完方式についての研究は国外でも行われておらず、新規性の高い研究成果が得られていると考える。

②今後の展望
まず、提案アルゴリズムの健全性および完全性の証明を行うことが重要な課題である。また、実際に用いられている Standard ML, Haskell, OCaml 等の言語に対し、変数名補完システムを実現することを目指している。実言語での変数名補完システムの実装に向け、主に以下の３点が課題になると考える。まず、現在の実装では文字が入力されるたびに最初から補完候補の計算を行っているので無駄が多い。構文解析や型推論の結果を保存しておいて再利用することが考えられる。計算の再利用に関する研究は多くあり、それらを利用することで計算量を減らすことが出来ると考えられる。次に、 カーソル位置より前には構文エラーや型エラーがないという仮定をしているため、プログラムの先頭の方に小さいエラーがあるだけで変数名補完が働かなくなる。この仮定は強すぎるので構文解析時のエラー回復などの方法によってエラーのあるプログラムでも変数名補完が働くようにできると考えられる。最後に、Haskell 等の言語では変数を束縛する前に使用することができ、また、ML では（相互）再帰関数の定義の中では束縛の前に変数を使用できる。我々の現在の枠組みではカーソル位置より後のプログラムテキストを使っていないので、それらのケースにそのままでは適用できない。問題定義を拡張することにより、そのような状況でも補完が行えるようになると考えられる。その他、パターンマッチング、中置演算子、モジュール、型注釈などの言語要素を扱えるように拡張する。

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）

〔学会発表〕（計　3 件）
① 後藤 拓実, 篠埜 功, 暗に型付けられた関数型言語に対する変数名補完方式の提案, 第 13 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(PPL2011) 論文集, pp. 216-230, 北海道札幌市 定山渓ビューホテル, 2011 年 3 月 9 日～11 日.

② Isao Sasano, Model Query Language MQL, Workshop on Bidirectional Transformation (BT2010), National Institute of Informatics, Tokyo, Japan, March 15, 2010.

③ 後藤 拓実, 篠埜 功, 多相型言語の変数名補完を行う Emacs モードの開発, 第 12 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(PPL2010) 論文集, pp. 177-190, 香川県 琴平温泉, 2010 年 3 月 3 日～5 日.

〔その他〕
変数名補完システムを公開している web ページ
http://www.cs.ise.shibaura-it.ac.jp/complement/

６．研究組織
(1)研究代表者
篠埜　功（SASANO ISAO）
芝浦工業大学・工学部情報工学科・助教
研究者番号：１０３６２０２１