メタレベル・プログラミングの信頼性

(研究課題名：プログラムのメタレベルを表現・操作する言語機構)

「機能と構成」領域　　亀山　幸義

要　旨

現代のソフトウェアは、高機能、高性能であるとともに、高信頼であることが強く要求されています。本研究では、プログラミング言語の基礎研究の立場から、高機能かつ高信頼なプログラミング方法論について考察しました。具体的には、高度に複雑な制御や環境の切換えなどを簡潔に記述する『メタレベル機構』に着目しました。最新のメタレベル機構を使ったプログラムに対する検証方法について研究し、いくつかの重要なケースについて、検証のための基礎理論を確立しました。本研究で得られた成果は、将来的に、プログラム変換プログラムやプログラム生成プログラムなどのいわゆるメタプログラムの検証に応用可能と考えられます。

1．研究のねらい

コンピュータ上でプログラムを実行するとき、プログラムのコード自身のほかに、環境や制御などの情報を利用します。環境は、変数とその値の対応であり、制御はプログラムカウンタ、すなわち、次に実行するプログラム中の番地の情報です。環境や制御は、通常のプログラム言語では、プログラムから直接操作可能な対象物ではなく、直接操作ができないレベル（メタレベル）にあるもの、すなわち、メタレベルの概念です。

メタレベル概念をプログラムの中から直接操作する機能、つまり、『メタレベル機構』を使うことにより、従来は極めて複雑な記述が必要だったプログラムを、簡潔にわかりやすく記述できることが知られており、研究されてきました。しかし、プログラム言語の安易な拡張は信頼性を損ねる可能性があります。たとえば、メタレベル機構より更に強力なものとして、プログラムの『自己反映』があります。これは、インタープリタなどの言語処理系をプログラムの中から変更できるため、「どんな計算でも出来る」機構となっており、静的な解析や検証をすることが困難です。

本研究は、メタレベル機構による高機能性・高記述性を、高信頼性と両立させた形で利用できるようにすることを目標としました。ここでポイントとなるのは、考察の対象とするメ

タレベル機構を、通常のプログラム言語の意味論に即した概念に対応するもののみに限定することです。このような概念は、自己反映計算のような無制限の拡張と異なり、拡張した言語の意味を、元の言語の意味から構築できると言う利点があります。本研究では、このように、意味論の助けを借りて、形式的体系によるモデリングと、それに対する検証方法の確立を行うことを目的としました。

本研究では、応用上重要なメタレベル機構として、精細な制御を実現する限定継続(delimited continuation) とその発展形、および、メタ変数と文脈を取り上げて、この目的を達成するための研究を行いました。

2. 研究方法と成果

2.1 非局所脱出の体系に関する研究

『さきがけ』研究の前段階として、非局所脱出（non-local exit）の機構の定式化と理論的解析に関する研究を行っており、『さきがけ』研究の期間に論文出版をしました。

非局所脱出は、多くの現代的プログラム言語で実装されている例外処理の機能に相当します。たとえば、Javaのtry/catch, Lisp系言語におけるcatch/throw, ＭＬ言語におけるexceptionの機構などです。これは、プログラムの制御を切り替えるという意味でメタレベル機構の一種と考えられます。

ここで取り扱ったのは、非局所脱出機構を持つプログラムの型システムと停止性の関係です。ループなどがない単純なプログラムでも、非局所脱出の機構を何の制限もなく導入すると、停止しないプログラムを書けてしまいます。脱出先を表すタグが動的な範囲（extent）である場合には停止しないプログラムが存在することを足がかかりにして、タグが静的な範囲を持ち、適当な型付け規則があるときには、非局所脱出機構を導入したプログラムが停止性（強正規化可能性、どのような順番で計算しても、実行がいつか停止すること）を持つことを証明しました。ここで用いた型システムは通常のプログラム言語とも両立するものとなっており、Javaのtry/catchなど広い範囲で応用がきくものです。

停止性そのものは極めて理論的な成果ですが、停止性を満たすかどうかは、型を持つ計算体系が適切に構築されたかどうかの一種の試金石となっています。非局所脱出の体系においても、停止性の証明をうまく構成するためには、むしろ計算規則が弱すぎる（非局所脱出の機構を使う規則が弱い）という逆説的な結果が得られ、言語設計への示唆が得られました。また、従来不明だった『脱出タグをあらわす型』の論理的意味付けが得られました。これら

の成果はTCS誌で発表しました。

静的なケースで停止性が成立するという結果は、脱出先を表すタグが動的か静的かに依存して、プログラムの停止性が変わってくることを意味しています。そこで、動的なタグを持つ体系について型システムを若干拡張した形で定式化し、その論理的意味付けを考察し、APLAS'02国際会議で発表しました。

2.2 精密な継続を扱う機構の公理化

『継続』(continuation) は、元来は、表示的意味論においてジャンプ命令を解釈するために発明された概念であり、「計算の残りの部分」を意味しています。プログラムの実行のある時点Aで、その時点での継続を保存し、計算が進んだ時点Bにおいて、その保存した継続Aを呼び出すとすれば、AからBへのジャンプ命令をあらわしていると考えられます。このように「現在の継続」を保存し、後に呼び出せるようにすることにより、非局所脱出、ループ、コルーティンなど多様な制御を表現することができます。「現在の継続」を保存する機能は、Scheme言語のcall/cc (call with current continuation) やStandard ML/NJ言語のcallccとして実現されており、これらに対する研究も多くの蓄積があります。

call/ccによる継続の機能は非常に強力と考えられていましたが、近年、call/ccではうまく記述できないプログラムのパターンがあることもわかってきました。call/ccの機構は、(1)「残りの計算の全て」を一括して保存することしかできず、「残りの計算の一部」を指定することができない、(2)保存した継続を呼び出すと、その時点の継続が捨てられてしまう (継続が入れ替わる)、の2つの欠点があり、これらの欠点により表現できないプログラムが存在しています。そこで、より精密な継続を表現するため、限定継続 (delimited continuation) の機構が提案されました。これは、部分継続、あるいは、合成可能継続という別名を持つことからわかるように、残りの計算のうちの一部を指定して保存することができ、さらに、保存した継続を呼び出すとき、現在の継続は捨てられずに2つの継続が合成されるというものです。本研究では、限定継続の機構のうち最も広く使われているshift/reset機構について研究を行いました。

$$
\begin{aligned}
1+(2\times(\text{call/cc}\ \lambda k.3+(k\,4))) &\rightarrow 1+(2\times(3+(k\,4)))\ \text{where}\ k=(1+(2\times\square)) \\
&\rightarrow 1+(2\times 4) \rightarrow 9 \\
1+\langle 2\times(\text{shift}\ k.3+(k\,4))\rangle &\rightarrow 1+\langle 3+(k\,4)\rangle\ \text{where}\ k=\langle 2\times\square\rangle \\
&\rightarrow 1+\langle 3+\langle 2\times 4\rangle\rangle \rightarrow 12
\end{aligned}
$$

図1 call/ccとshift/resetの違い (<…>がresetをあらわす)

従来、shift/reset機構については、CPS変換（関数型プログラム言語のコンパイラで使われる変換）による意味論、応用例、効率的な実装などが知られていましたが、検証のための規則はわかっていませんでした。また、操作的意味論はinformalなものとして与えられていただけであり、CPS変換による意味論に照らして十分であるかどうか等はわかっていませんでした。したがって、実装は存在しても本当に正しい実装かどうかの証明を与えることができませんでした。

本研究では、shift/resetを含む２つのプログラムが等しいことを証明する公理（等式）の集合を求めることに成功しました。この公理の集合は、既に与えられていたCPS変換による意味論に対して健全かつ完全（必要十分）なものであり、かつ、変換によらずに直接的にshift/resetに対して推論が可能なものであるため、ソフトウェア検証への応用が可能なものです。この結果を得るためには、CPS変換に対して、適切な逆変換を見つけ出すという点が鍵となりました。

当初与えた公理の集合はかなり複雑なものでしたが、さきがけ研究の領域会議などの場で当領域３期生の長谷川真人氏と議論を重ね、同氏の協力を得ることにより、公理を大幅に簡単にすることに成功しました。さらに従来よく研究されてきたcall/ccの理論との関係を明確にしました。この成果はICFP'03国際会議で発表しました。

その後、私たちの公理に基づき、操作的意味論も（ある程度）正当化され、shift/resetに対する研究が一段と進むきっかけとなりました。実際に、その後のCW'04ワークショップでは、限定継続に関する研究発表が５件中３件あり、また、ICFP'04国際会議でも別の著者らによる限定継続の定式化の研究が発表されるなど一種のブームのような状況となりました。

２．３　継続の階層化へ

上記の研究は理論的には非常にすっきりしたものでしたが、shift/resetを使ったプログラムの検証には不十分なものでした。なぜなら、shift/resetが１種類しかないことが前提になっているため、shift/reset機構を異なる目的で使っている２つのプログラムを組み合わせることができないからです。たとえば、shift/resetの代表的利用例であるバックトラッキングと複数解の収集は、それぞれ単独では問題なく動作するものですが、それぞれのプログラムを合成して１つの大きなプログラムの中で使おうとすると２種類の使い方が干渉してしまい、期待した結果が得られません。したがって大規模プログラムにおいては事実上使えない機能となっていました。この解決策として、shift/resetをレベル付けして、異なる使い方には

異なるレベルを与えるという階層化が提案され、レベルの数（階層の数）に応じて繰り返しCPS変換を施すという変換により、意味論が与えられていました。しかし、この変換は極めて複雑であり、これに対する理論的解析はほとんどなされておらず、したがって、階層化したshift/resetもあまり使われていませんでした。

本研究では、この階層化されたshift/reset機構について、変換によらずに直接に検証が容易になる規則の発見を目指しました。最初は、２種類のshift/reset（レベル２までのshift/reset）を持つプログラム言語に対する健全かつ完全な検証規則の集合を発見し、CW'04ワークショップで発表しました。しかし、ここで得られた集合は、極めて複雑な形をした多数の公理から構成されていたため、直接検証に使うことは困難で、その応用は限定的でした。

その後さらに研究を継続し、最終的に、任意のレベルnのshift/resetに対して、簡潔な検証規則の集合を発見しました。この検証規則の健全性および完全性の証明（必要十分な規則であることの証明）では、階層化されたCPS変換により変換されたプログラムを、型システムの構造により整理したことと、それに基づく逆CPS変換の発見が本質的に貢献しています。簡潔さの目安として、次の表に公理の個数の比較表を掲載します。ここでは公理の数だけを示していますが、公理の形自体も前節のものとほぼ同等（レベルに関する添え字を除くと完全に一致する）になっています。

表２　公理（等式）の個数

	λ計算に対する公理	resetに対する公理	shiftに対する公理
前節のもの（レベル１）	３個	２個	３個
本節のもの（レベルｎ）	３個	３個	３個

この成果をCSL'04国際会議で発表し、論理の立場からの討論を行いました。また、その後、上記の表のうち、resetに対する公理を更に簡単化して２個とすることに成功し、CSL'04論文の雑誌版として学術雑誌に投稿しました。

２．４　メタ変数と文脈に関する研究

入力としてプログラムを受け取ったり、プログラムを生成したりするプログラムを、簡単のために『メタプログラム』と総称します。メタプログラムは特別なものではなく、コンパイラなどの言語処理系、プログラム変換システムなどのほか、動的なwebページを作成するためのプログラムなど、様々なソフトウェアが含まれます。一方、メタプログラムの正しさ

を系統的に示したり、正しいメタプログラムを構成したりするための方法論は十分に研究されているとはいえません。そこで、本研究では、通常のレベル（オブジェクトレベル）とメタレベルとの区別がある簡単なプログラム言語を設計し、その上でメタレベルプログラミングがどのように行われるか基礎的な考察をしました。

メタプログラムにとって本質的なことは、当然ながら、オブジェクトレベルとメタレベルを区別することです。それぞれのレベルの変数、プログラム、環境、継続などの概念は異なるものとなるため、この区別に基づいたメタレベル機構が必要となります。ここでは、メタプログラムの定式化の第一歩として、メタレベルの変数（メタ変数）やそれに対する操作を定式化しました。メタ変数は通常の変数（オブジェクトレベルの変数）と同様に、λ抽象したり値を束縛したりすることができますが、メタ変数に対する代入は通常の代入と異なり、通常の変数のキャプチャを許すという特徴があります。

$$(\lambda X.\ \lambda x.\ X)\ (xy) \rightarrow \lambda y.\ xy$$
$$(\lambda X.\ \lambda Y.\ X)\ (xY) \rightarrow \lambda Z.\ xY$$

図２　キャプチャのある代入と、キャプチャを避ける代入
（通常の変数は小文字x, yで表し、メタ変数は大文字X, Y, Zであらわす）

この現象を、より一般的に定式化するため、メタレベルのメタレベル、すなわち、メタメタレベルや、メタメタメタレベルなど、無限のメタレベル階層を持つ体系の中で、通常の代入とキャプチャを許す代入の２通りの代入を導入しました。このような体系は極めて複雑になるように予想されましたが、適当な型システム（論文ではarityと呼んでいます）の導入により、計算体系として必要な性質（合流性、正規形に関する性質、停止性などの性質）を全て満たす体系が得られることを示しました。特に、正規形に関する性質は、「計算が途中でスタックすることがない」ことを保証する重要な性質です。

また、この体系においては、私たちが従来から研究していた、プログラム文脈をうまく表現できること、特に、文脈の合成を自然に表現できることがわかりました。プログラムの文脈というのは、１つだけ穴のあいたプログラムのことであり、プログラム中のある部分に着目したとき、それを取り囲むログラムと言う意味合いです。文脈は、モバイル計算など異なる環境で動作するプログラムをモデル化する際に、『環境』の側を表現するのに使うことができると言う応用があります。この場合、移動するプログラムの側にあらわれる自由変数が、移動後の環境で値を持つと言う効果を表すため、単なる代入では表現できず、キャプチャを許す代入が必要になります。

ここで述べた成果は、京都大学佐藤雅彦教授、五十嵐淳講師、千葉大学櫻井貴文助教授との共同研究によるものです。従来からの文脈に関する成果は、雑誌JFLPやコンピュータソフトウェアで発表し、ここで述べたメタ変数と文脈に関する成果発表はCSL'03国際会議で行いました。

2.5 その他の研究

さきがけ研究は特定の目標を掲げて実行してきましたが、この期間内にも「正しさの保証」という大テーマに関連する他の研究も行ってきました。その1つとして、大学院生の中島一氏とともに、モデル検査における抽象化の研究を行いました。具体的な検証例を取り上げ、そこで用いられる抽象化写像を、より細かいいくつかの基本的抽象化に分解して正しさを保証する方法や、実時間システムに対してモデル検査の効率を改善するため一種の静的解析を組み合わせて対象範囲を絞りこむ方法の研究などを行い、研究成果を発表しました。

3. 今後の展望

さきがけ研究における私のテーマは、プログラム言語のメタレベル機構の正しさについての基礎研究でした。特に、非常に精細なコントロールを得られる限定継続の機構とメタ変数・文脈の機構について研究を行い、意味論のガイドに基づき、型システム、論理の手法を用いて研究を行いました。前者については、階層化された形での一般的な限定継続機構の検証に関して、必要十分で、かつ、簡潔な規則を発見し、後者については、メタプログラムのモデル化の基礎となる体系の定式化という成果を得ました。本研究で取った手法は、形式意味論や型システムなど、従来のプログラム言語で研究されてきた道具立てを一から作りなおすのではなく、それらと調和するようにモデル化と形式化を行うことです。少なくとも今回の研究の範囲では、この方針は適切であったと考えています。

本研究の次のステップとして、ここで得た理論を基にして、実際に検証するためのシステムを実装し、具体的な例題を検証することが考えられます。より具体的には、種々のプログラム変換（たとえば、1期生のRobert Glück氏、小川瑞人氏、2期生の胡振江氏の研究など）などが検証対象です。また、以前からのテーマである証明からのプログラム抽出なども一種のメタプログラムであり、題材として適当であると考えています。これらに対して適当な論理体系の上で検証を展開することが今後の研究課題です。

このほかに、メタレベル機構を使った応用例を数多く構成することも重要だと考えていま

す。メタレベル機構は、現実のプログラム言語で直接使えるものは多くありませんが、概念として使われているケースは多いと考えられます。そのようなケースについてメタレベル機構を使って書きなおし、本研究の成果を生かす形になれば、プログラムの信頼性を向上させることが可能であると考えられます。この方向にそって研究を進めるためには、プログラム言語の設計が必要であり、そのためには、たとえば、shift/reset機構と型システムの関係など解決すべき課題も残っています。これらについては今後の検討課題としたいと思います。

終わりに：個人研究である「さきがけ」の研究期間を通じて、逆に、研究者交流の重要性を感じました。本研究は、領域会議における他研究員との交流、外国研究者との相互訪問などがなければ遂行できませんでした。これらのために様々な面からサポートしていただいたＪＳＴと、片山卓也先生はじめ「機能と構成」領域の先生方・１～３期の研究員の方々・職員の方々に深謝します。

４．成　　果

書籍（編）

1. Yukiyoshi Kameyama, Peter J. Stuckey (editors), FLOPS 2004 Proceedings, Lecture Notes in Computer Science Volume 2998, Springer, 307 pages, April, 2004.

学術雑誌

1. Yukiyoshi Kameyama, Masahiko Sato, "Strong Normalizability of the Non-deterministic Catch/Throw Calculi", Theoretical Computer Science 272(1-2), pp. 223-245, 2002.
2. Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, "A Simply Typed Context Calculus with First-Class Environments", Journal of Functional and Logic Programming 2002(4), pp. 1-41, 2002.
3. Azza A. Taha, Masahiko Sato, Yukiyoshi Kameyama, "A Second Order Context Calculus", コンピュータソフトウェア 19: 3, pp. 2-19, 2002.
4. 中島　一, 亀山幸義, 「抽象化と精密化による実時間モデル検査の改善」, 情報処理学会論文誌: プログラミング Vol. 45, No. SIG12 (PRO23), pp. 11-24, 2004.

査読つき国際会議

1. Yukiyoshi Kameyama, Masahito Hasegawa, "A Sound and Complete Axiomatization for Delimited Continuations", Proc. Eighth ACM International Conference on Functional Programming (ICFP'03), pp. 177-188, 2003.
2. Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, Atsushi Igarashi, "Calculi of Meta-Variables", Proc. 17th International Workshop on Computer Science Logic (CSL'03), Lecture Notes in Computer Science Volume 2803, pp. 484-497, 2003.
3. Yukiyoshi Kameyama, "Axiomatizing Higher-Level Delimited Continuations", Proc. Fourth ACM-SIGPLAN Continuation Workshop (CW'04), pp. 49-53, 2004.

4．Yukiyoshi Kameyama, "Axioms for Delimited Continuations in the CPS Hierarchy", Proc. Annual Conference of the European Association for Computer Science Logic (CSL'04), Lecture Notes in Computer Science Volume 3210, pp. 442-457, 2004.

口頭発表

1．Yukiyoshi Kameyama, "Dynamic Control Operators in Type Theory", Second Asian Workshop on Programming Languages and Systems, Daejon, Korea, Dec., 2001.
2．Yukiyoshi Kameyama, "Partial Continuation and CPS-Translation", Workshop on Foundation of Software, Hangzhou, China, Sep., 2003.
3．亀山幸義，中島　一，『ケーススタディ：モデル検査と定理証明を用いた信号制御システムの検証』，システム検証の科学技術シンポジウム，産業技術総合研究所システム検証研究ラボ，2004年２月．

など。