

## 研究課題別評価

### 1 研究課題名:

プログラミング言語の制御構造の意味論的分析

### 2 研究者氏名:

長谷川 真人

### 3 研究のねらい:

現在用いられているプログラムおよびプログラムによって表現されたアルゴリズムは、一般に、とても複雑な構造をもっています。その構造は、多くの場合、プログラムの中で用いられるデータの構造、そしてプログラムの実行過程をつかさどる制御の構造の、巧みな組み合わせから生みだされます。前者の、データ構造の特性を生かした効果的なアルゴリズム設計は、古くから計算機科学の中心的な話題のひとつでした。一方、プログラミング言語の制御構造の活用も、また古典的な問題ですが、高度な経験則や抽象的な思考を必要とするため、必ずしも広く深く理解されてきたとは言いがたいのが実情です。古くは goto 文のようなジャンプの使用の是非に関する議論が有名ですが、その一般化である継続(コンティニュエーション)、自己言及的なプログラミングを許す再帰、あるいはプログラムをデータとしてやりとりする高階のプログラムなど、いずれも表現力に富み、プログラムに明快な構造を与える高度な制御構造が、実際のプログラミングの現場では、むしろいたずらにプログラムをわかりにくくするものとして敬遠されたり、理論屋のおもちゃに過ぎないと揶揄されたりしているのが実情です。

確かに、高度な制御構造は、習得に時間を要します。理解不足のまま用いれば、いともたやすく「スパゲッティプログラム」の出来上がりです。けれども、正しく用いれば、プログラムの開発・保守・改良に際し、大変有効な道具です。使わないのはもったいない。問題は、その「正しく使う」ための指針が、必ずしも誰にでもわかるかたちで与えられていないことです。これは、単に良い教科書がないとか良い先生がいらないなどということではありません。実のところ、プログラムの制御構造の相互作用をきちんと解き明かした理論がまだないのです。個々の制御構造を取り出してきちんと調べたり解説した例は数多くありますが、現実のプログラミング言語でおきている、様々な制御構造がお互いに干渉しあって生じる現象は、まだ理論的に十分には解明されてはいないのです。

本研究は、そのような、制御構造間の関係を、プログラミング言語の基礎理論の最新の成果を駆使して、明らかにしようというものです。特に、継続、再帰、高階プログラム、そして多相型プログラムの関係に焦点を当て、プログラム間に成り立つ等式の理論を導きました。

### 4 研究成果:

#### 4. 1 継続の意味論

継続は、もともと、代表的な制御構造のひとつであるジャンプを意味論的にとらえるために考えられたものですが、近年、さまざまなプログラミング言語で、一般化されたジャンプとして「(第一級の)継続」が用いられるようになりました。「継続」とは、「これから継続して行う残りの計算」のことです。プログラムの実行中に、その時点での継続に適当なラベル  $L$  をつけて覚えておき、しばらく計算がすすん

でからLのあらわす継続を呼び出して使うことにより、ラベルLへのジャンプを表現することができます。そのほか、コルーティンなど、多様な制御を、継続を用いて表現することができます。

しかし、継続を多用したプログラムの意味を理解することは、大変困難です。goto 文を濫用して不要に複雑になったプログラムのことを、よく「スパゲッティ」と呼びますが、継続を安直に使うと、さらにはたちの悪い「高階のスパゲッティ」(高階、というのは、継続を引数として用いるプログラムは高階の汎関数とみなせるからです)が簡単にできてしまうのです。一方で、継続の現実的な使い方の多くでは、継続は「線形」に、つまりつねに一回しか用いられないことが指摘されています。本研究では、この、線形にもちいられる、いわば「筋の良い」継続のための意味論を整備しました。また、その重要な応用として、亀山幸義氏(つくば大学、当領域 2 期生)とともに、「限定継続」とよばれる精密な継続を扱う機構についてなりたつ等式理論を導きました。これは、限定継続では、メタ継続という常に線形に用いられる継続があることに着目したことが大きな鍵となりました。以下に挙げるのは、この研究によってつきとめられた、限定継続を持つ(値呼び)ラムダ計算の等式理論の公理系です。

$$\begin{aligned}
 \langle F[\text{shift } M] \rangle &= \langle M (\lambda x. \langle F[x] \rangle) \rangle & (1) \\
 \text{shift } (\lambda k. k \ M) &= M \ (k \text{は} M \text{に自由に出現しない}) & (2) \\
 \text{shift } (\lambda k. \langle M \rangle) &= \text{shift } (\lambda k. M) & (3) \\
 \langle \text{let } x \text{ be } \langle M \rangle \text{ in } N \rangle &= \text{let } x \text{ be } \langle M \rangle \text{ in } \langle N \rangle & (4) \\
 \langle V \rangle &= V \ (V \text{は値}) & (5)
 \end{aligned}$$

これらを用いて、限定継続を用いたプログラムについて、たとえば以下のような等式を用いた推論を行うことが可能です。

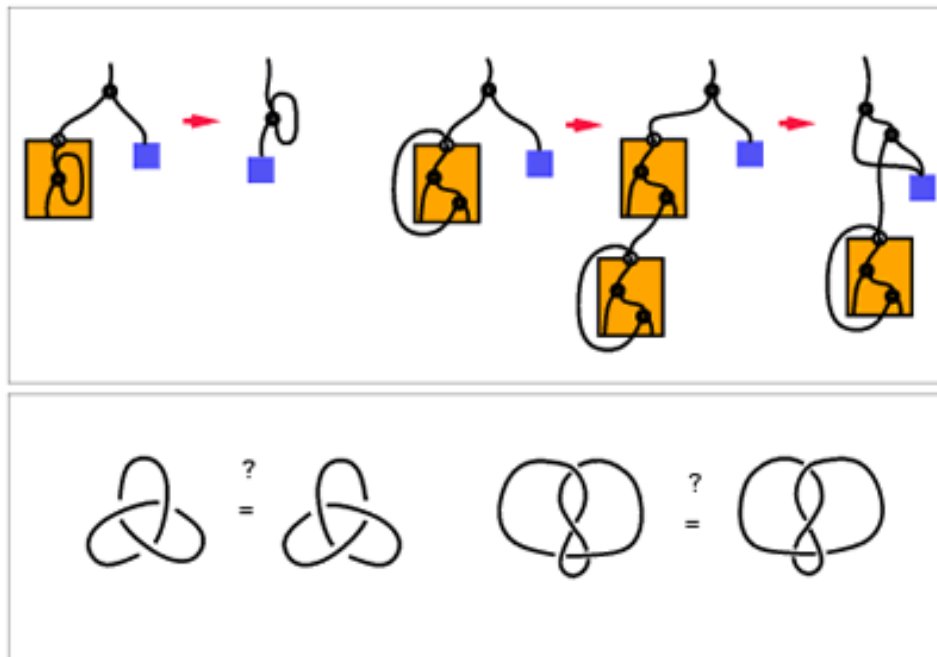
$$\begin{aligned}
 &1 + \langle 2 * (\text{shift } (\lambda k. 3 + (k \ 4))) \rangle \\
 = &1 + \langle (\lambda k. 3 + (k \ 4)) (\lambda x. \langle 2 * x \rangle) \rangle & (1) \text{より} \\
 = &1 + \langle 3 + ((\lambda x. \langle 2 * x \rangle) \ 4) \rangle & \text{値呼び } \beta \text{ 簡約} \\
 = &1 + \langle 3 + \langle 2 * 4 \rangle \rangle & \text{値呼び } \beta \text{ 簡約} \\
 = &\dots & (5) \text{などより} \\
 = &12
 \end{aligned}$$

プログラム中、kは、xから2 \* xを求める計算(2 \* [-])につけられたラベルです。この等式推論により、プログラムの実行結果を正しく、また明快に求めることができます。

また、この「継続の線形な利用」は、より一般の制御構造の線形な利用のための、一般化された意味論の枠組みに一般化して議論できることも示しました。

#### 4. 2 再帰法の意味論

再帰法は、自己言及的なプログラムを書くことを可能にする、代表的な制御構造です。本研究に先立つ研究で、巡回的なデータ構造から生み出される再帰計算について分析するための圏論的なモデルの理論を展開しており、本研究では、この理論の発展・応用のために欠かせない、「よい性質を持つモデルを構成する技法」について成果をあげました。



この研究の核心となったアイデアは、再帰計算の実装(上図)の違いを適切に捉えるために、絡み目(下図)を分類する不変量のための圏論的構造を応用するというものです。特に、扱いやすいモデルからは、プログラム間の関係が自然に導き出せるので、現在では、そのようなモデルを、すでに知られているモデルをもとに構成する方法を見つけることが大きな問題になっています。

具体的には、この理論で中心的な役割を果たす、トレースつきモノイダルカテゴリとよばれる数学構造に関して、巡回構造を生み出すトレースとよばれる演算が満たす一様性原理の理論を構築しました。これは、古典的な表示的意味論(領域理論)において最小不動点演算子がみたすプロトキンの一様性原理と呼ばれるものの、一般のトレースつきモノイダルカテゴリへの拡張と考えることができます。本研究では、この一様性条件による制約を受けた要素だけからなる「使いやすい」モデルを構成する一般的な技法を与えました。さらに、この方法が自然なものであることを示す例として、古典的な意味論において再帰法について推論するための基本的なテクニックであるスコットの不動点帰納法が、このトレースつきモノイダルカテゴリの構成技法から自然に導かれることを示しました。

#### 4.3 制御構造の相互作用

現実のプログラミングにおいては、複数の制御構造が、巧妙に組み合わせられて用いられることがよく見られます。しかし、はじめに述べたように、そのような状況を明快に説明できる理論は、まだ存在しません。この方面では、本研究に先立ち、継続、繰り返しと再帰の組み合わせに関する状況を明快に説明する圏論的な枠組みを与えていました。以下に挙げるのは、その理論的考察から得られた、第一級継続と繰り返しの組み合わせによる再帰のプログラム例です。

```

(* an empty type "bot" with an initial map "abort" A : bot -> 'a *)
datatype bot = VOID of bot;
fun A (VOID v) = A v;
(* the C operator, C : (('a -> bot) -> bot) -> 'a *)
fun C f = SMLofNJ.Cont.callcc (fn k => A (f (fn x => (SMLofNJ.Cont.throw k x) : bot)));

(* basic combinators *)
fun step F x = C (fn k => F k x);      (* step : (('a -> bot) -> 'b -> bot) -> 'b -> 'a *)
fun pets f k x = k (f x) : bot;      (* pets : ('a -> 'b) -> ('b -> bot) -> 'a -> bot *)
fun switch l x = C (fn q => l (x,q));  (* switch : ('a * ('b -> bot) -> bot) -> 'a -> 'b *)
fun switch_inv f (x, k) = k (f x) : bot; (* switch_inv : ('a -> 'b) -> 'a * ('b -> bot) -> bot *)

(* an iterator, loop : ('a -> 'a) -> 'a -> bot *)
fun loop f x = loop f (f x) : bot;

(* recursion from iteration *)
fun fix F = switch (loop (step (switch_inv o F o switch)));
(* fix : (('a -> 'b) -> 'a -> 'b) -> 'a -> 'b *)

```

意味モデルの分析から得られた等式理論を用いて、このプログラム例が、再帰と繰り返しの間の一対対応を与えていることが(限定継続の例と同様の、等式を使った推論により)証明できます。

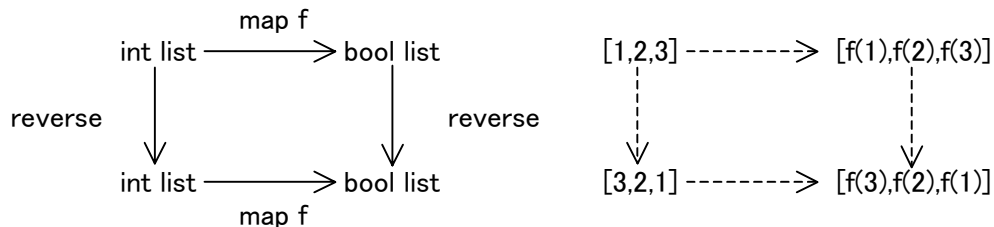
本研究では、角谷氏とともに、この枠組みでの再帰法、継続とパラメータの間に成り立つ関係を分析し、「関数完全性」とよばれる、パラメータの取り扱いに関する基本的な性質が成り立つことを示しました。また、再帰法がこの枠組みでみたすべき一様性原理を調べ、簡単な特徴づけを与えました。その結果、適切な一様性原理のもとでは、上に述べた圏論的な意味モデルはパラメータに関する必要な性質を常に満たしていることがわかり、意味モデルの満たすべき条件を、以前の研究で与えていたものから、大幅に単純化することができました。

#### 4. 4 多相型プログラムと制御構造

以上のテーマについて研究を進めていく過程で、プログラミング言語の制御構造は、その中に(たいい暗黙のうちに)用いられている、ある種の「一様性」に注目して分析を行うのがよい、ということがわかってきました。つまり、制御構造を用いたプログラムが、様々な状況で示す一様な振る舞いに、その制御構造の本質が現れている、と気がついたのです。そのことを端的に示すものとして、本研究の後半で、制御構造を用いる多相型プログラムが満たす、「パラメトリシティ原理」と呼ばれる一様性原理について、集中的に調べました。

多相型プログラムとは、様々なデータ型に対して動くように書かれたプログラムです。ここでは、それらのデータ型の内容に一切関わりなく動くように書かれた(パラメトリックな)多相型プログラムだけを考えることにします。たとえば、リストの順番をひっくり返すプログラムは、整数型のリスト、真偽値型

のリスト、文字列型のリスト、など、どんなデータ型のリストについても使うことができる多相型プログラムです。このようなプログラムは、データ型のとりかたに関係なく一様にふるまうのですが、その一様性を数学的に定式化したのが「パラメトリシティ原理」と呼ばれるものです。リストの順番をひっくり返すプログラム `reverse` は、パラメトリシティ原理によって以下の可換性を満たすことがわかります。



ここで、`f` は、整数を入力に取り真偽値を出力する任意のプログラムです。

このような、プログラムに関する代数的な法則が、パラメトリシティ原理から自然に導かれます。これらは、プログラムの性質に関する推論や、プログラムの最適化などのために、基礎的な役割を果たします。

ところで、パラメトリシティ原理は、制御構造を全く含まない、いわゆる純関数型プログラミング言語とそのモデルの理論に関してよく調べられ、その正当性もわかっていますが、再帰や継続のような制御構造が存在する場合には、実はそのままでは正しくないのです。一般に、多相型プログラムは、制御構造を含むプログラムについては、一様に動く保証がないのです。この問題は、再帰の場合には以前から知られていました。たとえば、再帰を用いた(止まらないかもしれない)多相型プログラムは、常に定数を返すようなプログラムに関しては、先ほどの可換性を満たしません。

だからといって、パラメトリシティ原理は、制御構造を持つプログラミング言語について何も言えないか、というと、決してそうではありません。「一様に動く」ことを保証するための、よい条件を見つけることで、制御構造を用いた多相型プログラムのための、一般化されたパラメトリシティ原理を展開することができます。また、その「よい条件」は、用いる制御構造によって定まる本質的な性質のはずです。実際、再帰に限れば、すでに「線型性」という、プログラムの停止性に関する本質的な条件を用いた、「線型パラメトリシティ原理」の理論が、すでに考えられていました。(この線型性は、再帰とトレースつきモノイダルカテゴリの研究で出てきた一様性の定義に用いられるものと、同一のものと思って差し支えありません)。本研究では、線型性に代わり、ジャンプと可換であるという「焦点の合った性質」を用いることで、継続を用いても成り立つ「フォーカル(focal)パラメトリシティ原理」を提案し、その正しさを証明しました。

さらに、その後、「線型パラメトリシティ原理」「フォーカルパラメトリシティ原理」は、より一般的な制御構造に関する理論として、大きなひとつの枠組みのなかで統一的に議論をすることができるようになりました。この、多相性と制御構造の一般論に関する理解は、本研究で得られた最も深い成果だと考えています。

## 5 自己評価:

さきがけ研究における私の仕事は、およそ10年前からとりくんできた、プログラミング言語における

制御構造に関する基礎研究の、自然な継続です。プログラム意味論の最新の成果を利用してプログラムの制御構造に関する、正確かつ有用な原則を抽出する、という当初の目的を十分に達成したのみならず、そのプログラム意味論そのものを大きく前進させる結果を出せたと考えています。長く取り組んできた再帰の意味論についてはその掘り下げを行い、また、比較的最近になって着手した継続の意味論については、亀山氏との共同研究の成功を含め予想以上の成果をあげることができました。そしてなにより、再帰、継続、線形性、一様性といったこれまでに取り組んできた問題を自然に総括するかたちで、プログラムの多相性と制御構造に関する、本質的なアイデアにたどりつくことができました。この、本研究の到達点ともいえる、多相型プログラムと制御構造のパラメトリシティ原理の理論は、今後、この分野の中心的な話題になっていく可能性があると考えています。

#### 6 研究総括の見解:

長谷川氏の研究は、プログラムの振舞いを表す制御構造を数学的に解明することを目的としたものである。現在のプログラム言語は種々の制御構造を持ち、それによって高度な計算を表現できる一方、その組み合わせによって複雑な振舞いが生じ、プログラムの理解を妨げ、誤りの原因になることが指摘されている。長谷川氏は、再帰、継続、多相性をもった高階関数型言語の振舞いを正確に表現するプログラム意味論を、カテゴリ理論などを用いて数学的に展開し、これまでのプログラム意味論を大きく前進させたものであり、この研究から生まれたパラメトリシティ原理は、今後の意味論研究で重要な位置を占めるものと考えられる。国際的な評価も高く、IBM科学賞を受賞するなど、非常に高く評価できる研究である。

#### 7 主な論文等:

##### (1) 学術雑誌

1. M. Hasegawa, The Uniformity Principle on Traced Monoidal Categories. Publications of the Research Institute for Mathematical Sciences 40(3): 991-1014, September 2004.
2. Y. Kakutani and M. Hasegawa, Parametrizations and Fixed-point Operators on Control Categories. Fundamenta Informaticae 65(1-2): 153-172, March 2005.
3. M. Hasegawa, Classical Linear Logic of Implications. Mathematical Structures in Computer Science 15(2):323-342, April 2005.
4. J.R.B. Cockett, M. Hasegawa and R.A.G. Seely, Coherence of the Double Involution on \*-Autonomous Categories. To appear in Theory and Applications of Categories.

##### (2) 査読つき国際会議

5. Y. Kakutani and M. Hasegawa, Parametrizations and Fixed-point Operators on Control Categories. Proc. 6<sup>th</sup> International Conference on Typed Lambda Calculi and Applications (TLCA'03), Springer Lecture Notes in Computer Science 2701, pages 180-194, June 2003.
6. Y. Kameyama and M. Hasegawa, A Sound and Complete Axiomatization of Delimited

Continuations. Proc. 8<sup>th</sup> ACM SIGPLAN International Conference on Functional Programming (ICFP2003), pages 177-188, August 2003.

7. M. Hasegawa, Semantics of Linear-Continuation Passing in Call-by-name. Proc. 7<sup>th</sup> International Symposium on Functional and Logic Programming (FLOPS2004), Springer Lecture Notes in Computer Science 2998, pages 229-243, April 2004.
8. M. Hasegawa, Relational Parametricity and Control (Extended Abstract). Proc. 20<sup>th</sup> Annual IEEE Symposium on Logic in Computer Science (LICS2005), pages 72-81, June 2005.

(3) 口頭発表

9. 長谷川真人、Traces in Computer Science、トポロジーとコンピュータ研究集会、2002 年 11 月
10. 長谷川真人、Recursive Programs in the Abstract、第 6 回プログラミング及びプログラミング言語ワークショップ(PPL2004)、2004 年 3 月
11. M. Hasegawa, On a Complete Axiomatization of Delimited Continuations, Joint Queen Mary and Imperial College Theory Seminar, June 2004
12. 長谷川真人、Geometry of Non-deterministic Interaction、第 8 回シンポジウム「代数・言語・計算」、2005 年 2 月

(4) 受賞

2005 年 11 月 第 19 回日本 IBM 科学賞(コンピューターサイエンス分野)