プログラムをときほぐす数学理論

（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）

「機能と構成」領域　長谷川 真人

要　旨

実用的なプログラムの多くは、再帰やジャンプなどの高度かつ複雑な制御構造を用いて書かれており、その正確な設計や理解は大変困難です。本研究では、プログラミング言語のモデルの理論を的確に用いることによって、そのようなプログラムについて成り立つ有用な原則を抽出し、その正しさを数学的に証明しました。特に、再帰法や継続法を用いたプログラムに関する基本的な等式を導きました。これらの成果は、今後、プログラムの設計、検証、最適化などにおいて、基礎的な役割を果たすことが期待されます。

1．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、巧妙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う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、制御構造の間の関係を、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。

2．研究方法と成果

2．1　プログラミング言語の意味論の新しい流れと本研究の位置づけ

個々の研究項目とその成果の説明のまえに、本研究の全体に共通する背景と研究方針について述べます。「プログラミング言語の意味論」とは、プログラムの意味する内容を、数学的なモデルを介して分析する研究分野です。本研究は、この、プログラミング言語の意味論の中でも、抽象度の高い数学構造を利用する、「表示的意味論」と呼ばれる手法を中心にしたものです。

ただし、抽象度をあげすぎて重要な情報が失われてはいけません。プログラムの制御構造に関する情報を最大限保っていて、かつ、余計な情報がなるべくそぎ落とされている、そのようなモデルを構築し、分析することに焦点をあてました。言い換えると、モデルが満たすべき条件をまずはっきりさせたうえで、90年代以降の新しい理論を駆使して、そのような条件を満たすモデルを構築し、その分析を行ったのです。これは、古典的な表示的意味論の舞台である「領域理論」が、90年代に、その古典的な枠組みから脱皮し、表示的意味論に必要な条件を満たすようなモデルの理論として再構築されたことと無関係ではありません。本研究は、領域理論という抽象的なレベルで起きた革命と同様のことを、より具体的なプログラムの制御構造の問題に関して試みた、といえると思います。

数学的には、「圏論」とよばれる抽象的な代数学の枠組みが、本研究の遂行のうえで大きな役割を果たしました。圏論では、個々の数学的な対象の内部構造よりも、それらの対象の間に成り立つ相互関係に焦点をあてて理論を展開します。プログラミング言語のモデルを与える際にも、そのモデルがどんな素材で作られているかよりも、モデルがどのような働きをするかのほうがずっと重要な問題であり、その求める「働き」を記述するのに、圏論は大変便利な道具となるのです。

2．2　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」などと呼びますが、継続を安直に使うと、さらにたちの悪い「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。

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

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することが可能です。

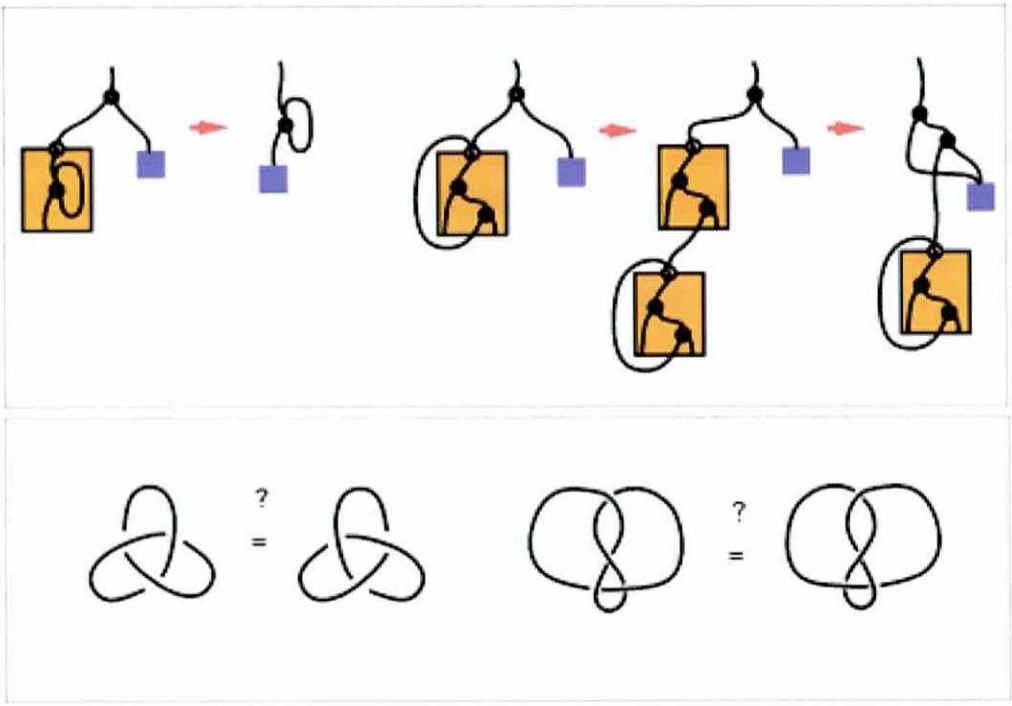
$$
\begin{aligned}
& 1+\langle 2*(\mathrm{shift}\,(\lambda k.3+(k\ 4)))\rangle \\
=\ & 1+\langle(\lambda k.3+(k\ 4))\,(\lambda x.\langle 2*x\rangle)\rangle && (1)\text{より}\\
=\ & 1+\langle 3+((\lambda x.\langle 2*x\rangle)\ 4)\rangle && \text{値呼び}\beta\text{簡約}\\
=\ & 1+\langle 3+\langle 2*4\rangle\rangle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

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。

２．３　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。そして、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。この方法が自然なものであることを示す例として、古典的な意味論で再帰法について推論するために基本的な道具となるスコットの不動点帰納法が、この一様性原理が与えるトレースつきモノイダルカテゴリの構成技法の一例となっていることを示しました。

２．４　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、巧妙に組み合わさっ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 I x = C (fn q => I (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一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。

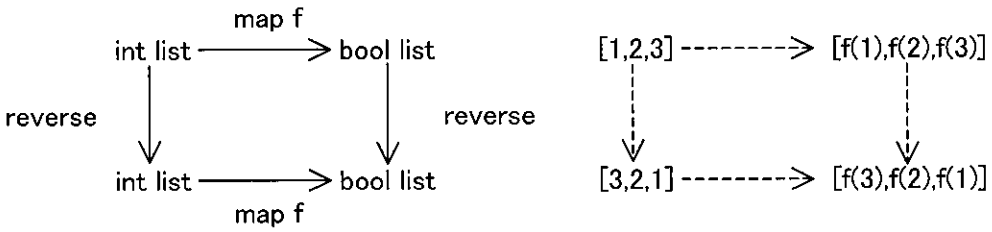
本研究では、角谷氏とともに、この枠組みでの再帰法、継続とパラメータの間に成り立つ関係を分析し、「関数完全性」とよばれる、パラメータの取り扱いに関する基本的な性質が成り立つことを示しました。また、再帰法がこの枠組みでみたすべき一様性原理を調べ、簡単な特徴づけを与えました。その結果、上に述べた圏論的な意味モデルの満たすべき条件を、大幅に単純化することができました。

２．５　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（たいてい暗黙のうちに）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るはずです。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。

この原稿を書いている時点ではまだ進行中の研究なのですが、「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ます。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。

3．今後の展望

本研究の自然な発展として、プログラムの制御構造の、明快かつ統一的な意味論が展開できると期待しています。特に、本研究の到達点である、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。

本研究を通じて感じたのは、プログラミング言語の制御構造の意味論の明快な発展のためには、以下の二点につねに注意を払うことが重要だということでした：

（1） 代数的（algebraic）：等式で書き下せる理論であること

（2） 一様性（uniformity）：線形性、多相性などに対し普遍的な性格を持つ理論であること

前者は、１９９０年代以降の計算の意味論の研究において次第にその重要性が認識されてきたポイントでもあります。一方、後者は、（このような一般化された観点では）本研究の一連の成果と経験を通じてはじめて明示的に認識されたことのように思われます。まとめますと、プログラミング言語の制御構造のための一般論は、適切な一様性を満たす代数理論として構築されるべき、ということが、本研究が示唆する重要な展望だと考えています。

謝　辞

本研究の遂行において、さまざまな側面からご支援頂いた、研究総括、領域事務所スタッフをはじめとする、本研究領域関係者の皆様に感謝いたします。

4．成果リスト

査読つき学術雑誌

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.

査読つき国際会議

1．Y. Kakutani and M. Hasegawa, Parametrizations and Fixed-point Operators on Control Categories. Proc. 6th International Conference on Typed Lambda Calculi and Applications (TLCA'03), Springer Lecture Notes in Computer Science 2701, pages 180-194, June 2003.

2．Y. Kameyama and M. Hasegawa, A Sound and Complete Axiomatization of Delimited Continuations. Proc. 8th ACM SIGPLAN International Conference on Functional Programming (ICFP2003), pages 177-188, August 2003.

3．M. Hasegawa, Semantics of Linear-Continuation Passing in Call-by-name. Proc. 7th International Symposium on Functional and Logic Programming (FLOPS2004), Springer Lecture Notes in Computer Science 2998, pages 229-243, April 2004.

4．M. Hasegawa, Relational Parametricity and Control (Extended Abstract). Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LICS2005), pages 72-81, June 2005.

口頭発表

1．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

2．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

3．M. Hasegawa, On a Complete Axiomatization of Delimited Continuations, Joint Queen Mary and Imperial College Theory Seminar, June 2004

4．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

など

表彰

第19回日本IBM科学賞（コンピューターサイエンス分野）、2005年11月