プログラミング言語における代数的及び統計的構造のシステム

松永拓也[1,a)]

概要：本発表では，変数・定数に替わる代数の概念や，代数的データ構造など，代数系の構造や理論をプログラミング言語に対応させ，数学を行うようにプログラミングをするという構造を提示する．カリー・ハワード同型対応によって圏論をプログラミング言語に対応させた関数型言語のように，本構造は，(機械) 計算的代数系たるプログラミング言語の設計の基礎となる理論である．

System of Algebraic and Statistic Structure on Programming Languages

Takuya Matsunaga[1,a)]

Abstract: In this presentation, I show the structure of algebraic systems, such as the algebraics instead of variables and constants, and algebraic data structures, by mapping them to programming languages. Like the functional languages which correspond to programming languages in terms of sphere theory by the Cary-Howard homomorphism correspondence, this structure is a fundamental theory for the design of programming languages which are (machine) computational algebraic systems.

Keywords: 代数, 代数的データ構造, 計算的代数系, 代数的整系

1. はじめに

コンピュータが誕生してから 70 年以上の時を経て，コンピュータに対して命令を与え操作するプログラミング言語は進化をし続けてきた．新たなプログラミング言語が何十何百と生まれ，進化していき，またそのなかでプログラミング言語に導入される各種機能やパラダイムが生まれ，進化してきているが，その多くは雑多であったり混沌としている[*1]ため，各々のプログラミング言語を通観するのはとても難しいし，処理系も複雑となっている．

この状況を打開するため，プログラミング言語の基礎となるべき代数的な性質を見出すことに着手した．

プログラミング言語に真に驚くべき代数的な体系を見出しているが，全て説明しようとすると時間的な余白が足りないため，本発表では基本的な論点について明らかにしていき，この体系の見通しを示す．

1.1 出発点と目標

プログラミング言語の仕様・設計及びソースコード中で可能なオブジェクト (後節で定義) を現実世界，とりわけ代数学の体系に完全に一致されるよう，プログラミングパラダイムにおいて整理・体系化することを目標とする．整理・体系化することにより明解になり，プログラミング言語の仕様・設計及びソースコードがより分かりやすくなり，したがってバグや罠を減らす効果をもたらす．

関数型及びオブジェクト指向を議論の出発点とする．現実世界とプログラムを対応付け理解しやすくし，またプログラミング言語の仕様・設計を明解にするという目標から考えると，「実世界を理解しやすくすることと，コンピュータ上への実装に関して実質的な基盤を与えることである．」とランボーにより定義される [1], (p.25) オブジェクトを取り扱うオブジェクト指向を論考の基盤の一つとすることは

[1] 日本大学理工学部応用情報工学科
Department of Computer Engineering, Nihon University, Funabashi, Chiba 274-8501
[a)] csta19097@g.nihon-u.ac.jp

[*1] 代表的なのは C++だろう．

妥当であると考えられる．

関数型は数学的な関数を基礎としてカリー・ハワード同型対応や圏論をはじめ代数学の様々な理論を用いて体系化されているが，本発表における計算的代数系はこれをさらに進めて，代数的構造を扱う代数学全体を用いて考察し直すことにより，より見通しよくすることができると信じる．

2. 基本事項

2.1 オブジェクト - Object

プログラム上における値そのものや，それらの値，そしてその値に対する演算などの処理を束ねたデータである?．概念や抽象あるいは対象となる問題に対して明確な境界と意味を持つ何ものかであり [1], (p.25)，コンピュータによる計算の対象となる実体である [2] と説明できる．

3. 代数 - Algebraics(Algebraic Models)

プログラム中で可能なオブジェクトの記号表現として，代数の概念を提唱する．代数とは，常数，定数，変数，関数を統合したものであり，これらの総称である．また，プログラム中で可能なオブジェクトの直接的な表現はリテラルである．

常数は普遍的不変性及び実行時不変性を，定数はコンパイル時不変性を保障する特別な変数ということができ，また関数は「関数型」[*2]というデータ型 (後述) が存在することから，これらを統一的に扱うことは合理的である．

代数は群の性質を持つ．関数に関して後節で明らかにしていくが，その他の常数，定数，変数についても同様に考えることができる．

3.1 型体系 - The Type System

型 (データ型) はそれ自身オブジェクトであり，またオブジェクトを定義する集合であると見做すことができる [5]．データ型は包含的派生型による型ツリーを構築する．

型は集合であるので，包含的派生型は部分集合に対応付けることができ，また型に関する集合演算が可能であり，演算結果を新たな型として定義したり，利用することができる．

集合たる型について考えることは不正なデータという重要なバグの一群を説明するのに役立つ [5]．型理論は圏論を用いて考察されるが，本発表では主題として取りあげないので，型体系に関する論考は省く．

3.2 関数 - Functions

関数は，データ型からデータ型への写像であり，具体的には計算や処理の定義である．独立代数を引数，従属代数を返り値とする．

[*2] 関数型プログラミングの関数型ではない．

引数及び返り値はいついかなる時でも一つである．複数の代数を受け渡しする必要がある場合はタプル型を，受け渡しする代数がない場合は Void 型を用いる．これは，写像は一対一対応であることと対応する．また引数の型は定義域，返り値の型は値域と対応する．

ここで論じる関数は，数学における関数と同じように合成することが可能であるものとする．この関数合成は，乗法演算と考えることができる，

以下の各節では，関数の満たす性質について考察する．

3.2.1 交換律

まず初めに，本節では関数が交換律を満たす場合があると考えられることについて述べる．

ソースコード 1 関数合成 A

```
1 fun1().fun2()
```

と

ソースコード 2 関数合成 B

```
1 fun2().fun1()
```

が等しいという命題は必ずしも成り立たない．

例えば，次例のように

(1) 参照透過性をもつ
(2) 片方の引数・返り値ともう一方の返り値・引数が対応する (または void である)
(3) 処理内容が双対的である

を満たす場合であれば可換であるし，

ソースコード 3 可換である例

```
1 int->int fun1(x) := x+1
2 int->int fun2(x) := x-1
```

次例のようにそうでない場合は可換ではない．

ソースコード 4 可換でない例

```
1 int->int fun1(x) := x+1
2 int->int fun2(x) := x*2
```

関数外で定義された外部変数の変更と，外部ファイルへの書き込みアクセスは，その後の計算環境を変更してしまうという点 [4], (2015-11-23 16:45 コメント) から，関数を

評価する順序を変更した場合に結果の同一であることを保障できないため，参照透明性に関するこの制約は妥当である．

また，引数・返り値の対応関係は，関数間で値の受け渡しをすることが可能か否かに，また双対性は関数が互いに対称的で関数の意味が全体として等しいかどうかに関係する．

3.2.2 結合律

次に，本節では関数が結合律を満たすと考えられることについて述べる．

関数合成の定義より

ソースコード 5 結合律 A

```
1 (fun1().fun2()).fun3()
```

及び

ソースコード 6 結合律 B

```
1 fun1().(fun2().fun3())
```

が等しい事は明らかである．

従って結合律を満たす．

3.2.3 分配律

続いて，本節では関数が分配律を満たすと考えられることについて述べる．

関数における分配律について述べる前に，説明のための演算子を 1 つ導入する．その演算子は”:>”で，式と式を並列し結合した式とする[*3]．

ソースコード 7 カスケード構文

```
1 fun1()..fun2()
2       ..fun3()
```

というカスケード構文より

ソースコード 8 分配律 A

```
1 fun1().(fun2():>fun3())
```

は

ソースコード 9 分配律 B

```
1 fun1().fun2()
2 fun1().fun3()
```

すなわち

ソースコード 10 分配律 C

```
1 (fun1().fun2()):>(fun1().fun3())
```

と等しい

従って分配律を満たす

3.2.4 単位元

なにもしない関数 nothing() を考えれば単位元となる．

3.2.5 逆元

関数を引数にとり作用を逆にした関数を返す高階関数 reverse() を考えれば，

ソースコード 11 逆元 A

```
1 fun1()
```

に対する逆元は

ソースコード 12 逆元 B

```
1 fun1().reverse()
```

と表せる

3.2.6 小括

結合律を満たし，単位元および逆元を有することから関数は群である．場合により交換律も満たし得ることから可換群ともなり得る．

[*3] 並列実行はランタイムにより同期的になることも非同期になることもあるがここでの議論には関わらないので未定義とする．

4. 代数的データ構造

代数的データ構造には共用体，構造体，関数体などがある．

4.1 関数体

関数体とは関数族，すなわち関数の集合である．またこの関数体において，ミクスインを加法演算と考えることができる．ミクスインの指定の並び順を変えてもその関数体に含まれる関数は等しいため，加法に関する可換群であるといえる．また，関数定義が何も含まれない関数体をミクスインしても変化はないことから，これを単位元とすることができる．さらに，ミクスインの逆の操作，すなわち指定した関数体に含まれる関数を取り除くことを考えることができ，これは逆元となる．したがって，関数体は環である．

加えて，前章で示した通り関数は群であるため，関数体は体である[*4]と言える．

4.2 小括

共用体，構造体にも同様に考えることができ，従ってそれらも同様に体である．

5. 総括

以上のように，プログラミング言語を代数的構造にあてはめて体系的に示すことができることが明らかになった．

5.1 代数的整系

本発表で示したような，プログラミング言語を代数的構造にあてはめて体系的に示すことができるシステムを代数的整系と呼ぶことにする．

5.2 計算的代数系

対応関係を見出すことによって代数的構造にあてはめる事が可能な，すなわち代数的整系である計算機システムを計算的代数系と呼ぶ．

[*4] 故に関数体という命名がなされている．

6. 終わりに

本発表では，プログラミング言語の基礎となるべき代数的な性質があること，そしてその核となる部分について示した．この洞察は，フェルマーによる定理とそれに対するワイルズによる証明のように，プログラミング言語における一大転換点となったであろう．ラングランズプログラムの様[*5]に，ここで示せなかった各論についても考察していければ幸いである．

さらに，プログラミング言語の基礎となるべき代数的な性質が存在するならば，代数的整系であるプログラミング言語を構築することも可能であろう[*6]．整った言語であれば．学習し理解するのにも難くないはずである．

謝辞

私自身が所属する日本大学理工学部応用情報工学科の教授，松野裕先生には日頃より型理論やソフトウェア工学など情報工学についてご教示を頂き，さらに相談やアドバイスをして頂いたりと，様々なご支援を頂いたことを深謝する．

また，アイデアに関する的確な指摘やアドバイスなど，日常的に研究をサポートして頂いた友人の皆，特に村田慶悟君，尾上遼太朗君にも感謝している．

参考文献

[1] J. ランボー/M. ブラハ/W. プレメラニ/F. エディ/W. ローレンセン; オブジェクト指向方法論 OMT, プレンティスホール (1992).

[2] 株式会社オージス総研: 第 2 回 - オブジェクトって何ですか？ -, トレーニングの現場から, 入手先 ⟨https://www.ogis-ri.co.jp/otc/hiroba/others/trainerEssay/tressay02.html⟩ (1999).

[3] Pikawaka：オブジェクトとは何か？基本となる知識をマスターしよう！, 入手先 ⟨https://pikawaka.com/ruby/object⟩ Pikawaka(2019).

[4] sasanquaneuf: 参照透過性と副作用についての提言, 入手先 ⟨https://qiita.com/sasanquaneuf/items/3df1001a027e868e9e0e⟩ Qiita (2015).

[5] Elm-jp コミュニティ: 集合としての型, 入手先 ⟨https://guide.elm-lang.jp/appendix/types_as_sets.html⟩ Elm 公式ガイド日本語訳 (1993).

[*5] プログラミング言語版ラングランズプログラムともいうべきプロジェクトは，本研究プロジェクトの協力者である村田慶悟君の通称名から名を取って「がーねっとプログラム」と命名する．

[*6] 私が設計に取り組んでいる，FunCobal family プロジェクトに属する FunCobal 言語もその一つである．