S2S 春セミナー 2013
「Scott の逆極限構成—計算機科学と圏論—」

Tomoki UDA (@t_uda)

2013-03-27 Wed.

概要

In this lecture, we consider fixpoint equations $X \cong F(X)$ for a set function F. For example, the set of natural numbers satisfies the recursive equation $\mathbb{N} = \{0\} \cup \{\, n+1 \mid n \in \mathbb{N} \,\}$, which can be easily derived from the recursive definition. On the other hand, λ-terms also have the recursive structure which looks like $\Lambda \cong V + [\Lambda \to \Lambda]$. However, in fact, any set X does not satisfies $X \cong A \cup (X \to X)$. Thus, set-theoretically, we cannot easily find a mathematical object which represents λ-terms. Our goal is to construct the solution to this equation in categorical settings.

Keywords　再帰方程式 (recursive equation)、再帰的定義 (recursive definition)、圏 (category)、関手 (functor)、F-代数 (algebra)、不動点 (fixpoint or fixed point)、始対象 (initial object)、ω-鎖 (chain)、余極限 (colimit)、逆極限 (limit or inverse limit)、逆極限法

圏 (category)　$\mathcal{C}, \mathcal{D}, \ldots$

対象 (object)　$A, B, \ldots$

　$A \in \mathcal{C}$

射 (morphism, arrow)　$f, g, h, \ldots$

　$f \colon A \to B$ in $\mathcal{C}$

　$g \colon A \to B$

　$h \in \mathcal{C}(A, B)$

関手 (functor)　$F \colon \mathcal{C} \to \mathcal{D}$

0　Preliminaries

この講義では、「圏論もしくは（関数型）プログラミング言語の経験」があることを仮定している。圏論の基本的な用語が分かるならば、講義の内容は難しくない。ここでは、プログラミング言語経験者が講義を理解しやすいように、圏論の用語とこの講義のモチベーションを簡単に説明する。なお厳密な説明はしないので、詳しい定義は参考文献を見ること。

0.1　圏の基本的な用語

講義では、圏を計算機科学的な視点で捉えるため、圏論の用語は以下のように理解して差し支えない。

対象	型 （型のついた集合）
射	副作用のない関数・プログラム （入力と出力の型が定められた写像）
圏	プログラムの集まり （型と関数の集まり）
関手	型から別の型を作る対応

E.g. 1

例えば、**Set** は集合を対象、写像を射とする圏である。整数型 Int は、整数全体の集合だと思えば、**Set** の対象と見なせる。関数 $\mathrm{succ} \colon \mathrm{Int} \to \mathrm{Int}; n \mapsto n+1$ は **Set** の射である。集合 A に対して、A の有限列全体の集合を $\mathrm{List}\, A$ と書くと、$\mathrm{List} \colon \mathbf{Set} \to \mathbf{Set}$ は **Set** の間の関手になる。

また、関数 $f \colon A \to B$ に対し、新しい関数 $\mathrm{List}\, f \colon \mathrm{List}\, A \to \mathrm{List}\, B; (x_i) \mapsto (f(x_i))$ を対応

させることができる。(註: 関数型言語では List f を map f と書くことが多い。)

この時、任意の $f\colon A \to B$ と $g\colon B \to C$ に対し、定義から明らかに $\mathrm{List}(g \circ f) = \mathrm{List}\, g \circ \mathrm{List}\, f$ が成り立つ。(実は、このように合成に関して分配則が成り立つ対応 F を関手と呼ぶのである。)

0.2 半順序集合

$\sqsubseteq$ を集合 P 上の二項関係とする。この時、$(P, \sqsubseteq)$ が半順序集合であるとは、二項関係 $\sqsubseteq$ が反射律、対称律、推移律を満たすことを言う。この順序について増大列 $p_0 \sqsubseteq p_1 \sqsubseteq \ldots$ を考える。$p \in P$ がこの増大列の上界であるとは、任意の p_i に対して $p_i \sqsubseteq p$ なることを言う。上界の中で最小のものが存在すれば、それを上限といい $\sqcup p_i$ と書く。

実は、半順序集合 $(P, \sqsubseteq)$ はそれ自体を圏と見なすことができる。つまり、この圏は、P の元を対象として、$p \sqsubseteq q$ が成り立つことで以てこの圏の射と見なすのである。この意味において、圏は半順序集合の一般化になっている。後の第 3 節では、増大列や上界、上限を圏の上で一般化した概念を扱う。

E.g. 2
計算機科学的な半順序集合の捉え方についても話そう。P を何らかの「プログラム」の集合とする。この時、半順序 $\sqsubseteq$ を上手く定めて、$p \sqsubseteq q$ が成り立つことを、プログラム p よりも q の方がより多くの情報を計算すると解釈することができる。

この解釈の下で、(非自明な) 増大列はプログラムの反復回数を増やすことに相当し、上限は反復回数を可能な限り増やした理想的なプログラムと思える。

例えば、素数の集合を入力として取り、新しい素数を計算し付け加え出力するプログラムを p とすると、p を反復適用することでより多くの情報が得られ (増大列)、無限回反復する理想的なプログラムは全ての素数を計算する (上限)。もちろん、素数全てを書き出すプログラムは現実には存在しないが、有限回の反復で打ち切れば現実的にもプログラムは作れるであろう。このように、上限への増大列は、理想的なプログラムを近似する列と捉えられる。

0.3 始対象

任意の集合 $A \in \mathbf{Set}$ に対して、空集合 $\emptyset$ は A への写像 (空写像) をただ一つ持つ。一般の圏 $\mathcal{C}$ において、任意の対象 $A \in \mathcal{C}$ への写像がただ一つだけ定まるような対象 $0 \in \mathcal{C}$ がある時、これを始対象 (initial object) と呼ぶ。どうせ A への射が一つしかないのだから、これのためにわざわざ $f\colon 0 \to A$ などと記号を用意してやるのはシャクである。そのため、この射を $!_A\colon 0 \to A$ と書くことがある。

E.g. 3
半順序集合 $(P, \sqsubseteq)$ の場合の例を挙げよう。P は最小元 0 を持つとする。これはつまり、任意の $p \in P$ に対して $0 \sqsubseteq p$ なることを意味するから、0 は圏として見た P の始対象である。このように、始対象は最小元の一般化になっている。

前提知識についての軽い (?) 説明はここまでにしよう。以下に、この後の講義の章立てだけ載せておこう。

1 Introduction

2 F-algebras and fixpoints

3 ω-chains and limits

4 **CPO**

講義の内容は主に [2] を基にした。計算機科学と圏論の関係がコンパクトにまとめられているので、興味があれば参照されたい。

参考文献

[1] Richard Bird and Oege de Moor. *Algebra of Programming.* Prentice Hall, 1997.

[2] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists.* The MIT Press, 1991.

[3] Benjamin C. Pierce. *Types and Programming Languages.* The MIT Press, 2002.

[4] John C. Reynolds. *Theories of Programming Languages.* Cambridge University Press, 2009.