ソフトウェア構成特論 第2回補足資料
関数の帰納的定義

大学院理工学研究科 電気電子情報工学専攻　篠埜 功

この資料では関数の帰納的定義（帰納的に定義された集合から何らかの集合への関数の定義方法）について紹介する。（この講義の範囲外なので読む必要はありません。） 2.2.2節で算術式（帰納的に定義されている）を引数に取る関数 *consts* を帰納的に定義した。「関数の帰納的定義」というものが何なのか、および関数の定義になっているのかをこの資料で示す。

1　自然数上の関数の帰納的定義

Nat（自然数の集合）は帰納的に定義される。自然数が一番単純なので、この節では自然数を例として、自然数を引数に取る関数の帰納的定義について述べる。

定理 1 集合 C、集合 C の要素 a、関数 $F : \mathbf{Nat} \times C \to C$ が与えられたとき、

$$\begin{aligned} f(0) &= a \\ f(n+1) &= F(n+1, f(n)) \end{aligned}$$

を満たす **Nat** から C への関数 f が存在するとしたらただ一つである。 □

証明 以下の等式を満たす関数 $f_1 : \mathbf{Nat} \to C$、$f_2 : \mathbf{Nat} \to C$ が存在するとする。

$$\begin{aligned} f_1(0) &= a \\ f_1(n+1) &= F(n+1, f_1(n)) \\ f_2(0) &= a \\ f_2(n+1) &= F(n+1, f_2(n)) \end{aligned}$$

このとき、数学的帰納法により、任意の $n \in \mathbf{Nat}$ について $f_1(n) = f_2(n)$ を示すことができる。
Base case: $f_1(0) = a = f_2(0)$
Induction step: $f_1(k) = f_2(k)$ を仮定すると、

$$f_1(k+1) = F(k+1, f_1(k)) = F(k+1, f_2(k)) = f_2(k+1)$$

が成り立つ。よって、数学的帰納法より、任意の自然数 n について $f_1(n) = f_2(n)$ が成立する。つまり、上記等式を満たす関数は、存在するとしたらただ一つということが証明された。 □

次に、上記等式を満たす関数が少なくとも 1 つ存在するということを示す。

定理 2 集合 C、集合 C の要素 a、関数 $F : \mathbf{Nat} \times C \to C$ が与えられたとき、

$$f(0) = a$$
$$f(n+1) = F(n+1, f(n))$$

を満たす **Nat** から C への関数 f が存在する。 □

これを示すために、まず以下の性質 $Q(x)$ が任意の $x \in \mathbf{Nat}$ で成り立つことを数学的帰納法で示す。

$Q(x)$: 以下を満たす関数 f_xが存在する。

$$f_x : \{n \in \mathbf{Nat} \mid n \leq x\} \to C$$
$$f_x(0) = a$$
$$f_x(n+1) = F(n+1, f_x(n)) \quad (n < x)$$

Base case:

$$f_0 : \{0\} \to C$$
$$f_0(0) = a$$

という関数が存在するので、$Q(0)$ が成立する。

Induction step:

$Q(k)$ を仮定する。つまり

$$f_k : \{n \in \mathbf{Nat} \mid n \leq k\} \to C$$
$$f_k(0) = a$$
$$f_k(n+1) = F(n+1, f_k(n)) \quad (n < k)$$

を満たす関数 f_k が存在することを仮定する。以下のように f_{k+1} を作る。

$$f_{k+1} = f_k \cup \{(k+1, F(k+1, f_k(k)))\}$$

これは、$\{n \in \mathbf{Nat} \mid n \leq k+1\}$ から C への関数である。まず、f_{k+1} の作り方から、

$$f_{k+1}(0) = f_k(0) = a$$

である。$n < k$ のとき、

$$f_{k+1}(n+1) = f_k(n+1) = F(n+1, f_k(n)) = F(n+1, f_{k+1}(n))$$

である。$n = k$ のとき、

$$f_{k+1}(n+1) = f_{k+1}(k+1) = F(k+1, f_k(k)) = F(n+1, f_{k+1}(k)) = F(n+1, f_{k+1}(n))$$

となる。$n < k$ のときと $n = k$ のときをまとめると、$n < k+1$ のとき

$$f_{k+1}(n+1) = F(n+1, f_{k+1}(n))$$

である。以上より、数学的帰納法により、$\forall x \in \mathbf{Nat}.\ Q(x)$ が成立する。ここで、

$$f = \bigcup_{x \in \mathbf{Nat}} f_x$$

とする。$x \leq y$ のとき $f_x \subseteq f_y$ であることより、f の中に左側が同じ対は存在しないので、f は **Nat** から C への関数である。$x \leq y$ のとき $f_x \subseteq f_y$ であることは、任意の $x \in \mathbf{Nat}$ について$f_x \subseteq f_{x+1}$ であることが f_x の構築法から言え、$f_x \subseteq f_{x+n}$ が任意の $n \in \mathbf{Nat}$ について成り立つことが自然数 n に関する帰納法で言えるので成り立つ。 □

定理 1 と定理 2 より、以下の定理が成り立つ。

定理 3 集合 C、集合 C の要素 a、関数 $F : \mathbf{Nat} \times C \to C$ が与えられたとき、

$$f(0) = a$$
$$f(n+1) = F(n+1, f(n))$$

を満たす **Nat** から C への関数 f がただ一つ存在する。 □

この定理は、

$$f(0) = a$$
$$f(n+1) = F(n+1, f(n))$$

により、1 つの関数が定義されていることを言っている。この 2 つの等式の形の関数定義が自然数上の帰納的な関数定義である。

2 整礎な関係の入った集合上の関数の帰納的定義

前節と同様にして、整礎な関係の入った集合上の関数を帰納的に定義できる。以下の本の p.177 の定理 10.19(well-founded recursion) を参照。

- Glynn Winskel, *The Formal Semantics of Programming Languages — An Introduction*, The MIT Press, 1993.

これは、2.2.2 節で提示した *consts* など、あらゆる帰納的な関数定義が数学的な関数を定義していることの根拠になる。