ソフトウェア構成特論 第2回

大学院理工学研究科 電気電子情報工学専攻　篠埜 功

2014年4月17日

1　はじめに

今回は、集合の帰納的定義および帰納法について解説する。今回の目標としては、集合の帰納的定義とはどういうことかを理解し、何らかの帰納的に定義された集合のすべての要素について成り立つ性質を帰納法を用いて証明することを体験し、実際に簡単な例について自分で証明ができるようになることを目指す。

2　集合の帰納的定義および帰納法

帰納的に定義された集合の要素が何らかの性質を満たすことを示す際に帰納法が用いられる。プログラミングの世界では、プログラム（の集合）は帰納的に定義され、また関数型言語においてよく用いられる再帰的データ型も帰納的に定義される。帰納的に定義される集合について成り立つ性質を証明するときに用いられるのが構造帰納法である。構造帰納法や高校等で勉強する数学的帰納法は整礎帰納法の例である。

2.1　数学的帰納法 (mathematical induction)

中学、高校等で習う数学的帰納法は、自然数について成り立つ性質を証明する際に用いる。自然数の集合の要素（つまり自然数）について成り立つ性質を証明する際に用いられるのが数学的帰納法である。

例 1　任意の自然数 n について、0 から n までの和は $n(n+1)/2$ と等しい。（ただし、自然数は 0 から始まるものとする。）

一般に、任意の自然数 n について $P(n)$ が成立することを示したい場合、以下の 2 つを示せばよい。

- $P(0)$ が成立する
- 任意の自然数 m について、$P(m) \Rightarrow P(m+1)$ が成立する

これが数学的帰納法である。つまり、

$$\{P(0) \wedge \forall m \in \mathbf{Nat}.\ \{P(m) \Rightarrow P(m+1)\}\} \Rightarrow \forall n \in \mathbf{Nat}.\ P(n)$$

が数学的帰納法である。$P(m)$ の部分を帰納法の仮定 (induction hypothesis)、$P(0)$ を帰納法の基本ケース (basis)、$\forall m \in \mathbf{Nat}.\ \{P(m) \Rightarrow P(m+1)\}$ を帰納法のステップ (induction step) と呼ぶ。

上記の例の場合、示したい性質は、

$$P(n):\ 0 + \cdots + n = n(n+1)/2$$

である。$\cdots$ の部分は直感的な記述だが、気になる人は漸化式などを使って定義すればよい。まず、$P(0)$ を示す。$P(0)$ の左辺が 0、右辺が 0 で、成立する。次に、$\forall m \in \mathbf{Nat}.\ \{P(m) \Rightarrow P(m+1)\}$ を示す。まず、$P(m)$ の成立を仮定する。つまり、

$$0 + \cdots + m = \frac{m(m+1)}{2}$$

が成り立つとする。この仮定のもとで、$P(m+1)$ が成立することを示せばよい。$P(m+1)$ の左辺は、$0 + \cdots + m + m + 1$ である。上記の仮定より、

$$0 + \cdots + m + m + 1 = \frac{m(m+1)}{2} + m + 1$$

である。この右辺をさらに変形していくと、

$$\begin{aligned}
0 + \cdots + m + m + 1 &= \frac{m(m+1)}{2} + m + 1 && (1)\\
&= \frac{m(m+1) + 2(m+1)}{2} && (2)\\
&= \frac{(m+1)(m+2)}{2} && (3)\\
&= \frac{(m+1)(m+1+1)}{2} && (4)
\end{aligned}$$

となり、$P(m+1)$ が成立する。

よって数学的帰納法より、$\forall n \in \mathbf{Nat}.\ P(n)$ が成立する。

練習問題 1 任意の自然数 n について、0 から n までの二乗和が $n(n+1)(2n+1)/6$ に等しいことを示せ。つまり、性質 $P(n)$ を

$$0^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

としたとき、

$$\forall n \in \mathbf{Nat}.\ P(n)$$

が成り立つことを示せ。

$P(m)$ を仮定しただけでは $P(m+1)$ を示すことができない場合があり、m 以下のすべての自然数について $P(m)$ が成り立つことを仮定することにより $P(m+1)$ を示すことができるようになる場合がある。このような場合に使えるのが以下の帰納法である。

$$\{\forall m \in \mathbf{Nat}.\ \{\forall k < m.\ P(k) \Rightarrow P(m)\}\} \Rightarrow \forall n \in \mathbf{Nat}.\ P(n)$$

この帰納法を累積帰納法 (course-of-values induction あるいは complete induction) という。

2.2 構造帰納法 (structural induction)

プログラムや、関数型言語においてよく用いられる再帰的なデータ型について成り立つ性質を証明する際に用いられるのが構造帰納法である。ここでは、算術式 (arithmetic expression) を例として帰納的定義および構造帰納法について説明する。

2.2.1 算術式の帰納的定義

まず算術式を以下のように定義する。

```
t ::= true
    | false
    | if t then t else t
    | 0
    | succ t
    | pred t
    | iszero t
```

これは BNF 記法とほとんど同じである。(BNF 記法だと t を<>で囲むが。) 上記で定義しようとしているのは算術式 (算術式の木) の集合である。(通常 BNF で定義するのは文字列の集合だが、ここでは木の集合を定義している。) 上記の定義においては、定義しようとしている t が右辺にも現れており、これでは定義になっていないのではないかという疑問があるだろうが、この表記はこれから説明する、集合の**帰納的定義**を簡潔に書いたものである。

定義 1 (算術式の集合の帰納的定義) 算術式 (の木) の集合は以下の 3 つの条件を満たす最小の集合 $\mathcal{T}$ である。

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$
2. $\texttt{t}_1 \in \mathcal{T}$ ならば $\{\texttt{succ t}_1, \texttt{pred t}_1, \texttt{iszero t}_1\} \subseteq \mathcal{T}$
3. ($\texttt{t}_1 \in \mathcal{T}$ かつ $\texttt{t}_2 \in \mathcal{T}$ かつ $\texttt{t}_3 \in \mathcal{T}$) ならば ($\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}$)

この定義は次のように推論規則を使った形で書き直してもよい。

定義 2 (算術式の集合の推論規則による定義)

$$\texttt{true} \in \mathcal{T} \qquad \texttt{false} \in \mathcal{T} \qquad \texttt{0} \in \mathcal{T}$$

$$\frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{succ t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{pred t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{iszero t}_1 \in \mathcal{T}}$$

$$\frac{\texttt{t}_1 \in \mathcal{T} \quad \texttt{t}_2 \in \mathcal{T} \quad \texttt{t}_3 \in \mathcal{T}}{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}}$$

上記 2 つの定義は定義したい集合が満たすべき性質を与え、それらを満たす集合の中で最小の集合として定義するという間接的な定義である。これに対し、以下のように定義すると直接集合を構築することができる。

定義 3 (算術式の集合を直接構築する定義) 各自然数 i について、集合 $\mathcal{S}_i$ を以下のように定義する。

$$\begin{aligned}\mathcal{S}_0 &= \emptyset \\ \mathcal{S}_{i+1} &= \{\texttt{true}, \texttt{false}, \texttt{0}\} \\ &\quad \cup \{\texttt{succ t}_1, \texttt{pred t}_1, \texttt{iszero t}_1 \mid \texttt{t}_1 \in \mathcal{S}_i\} \\ &\quad \cup \{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \mid \texttt{t}_1, \texttt{t}_2, \texttt{t}_3 \in \mathcal{S}_i\}\end{aligned}$$

これらの集合を用いて、集合 $\mathcal{S}$ を以下のように定義する。

$$\mathcal{S} = \bigcup_i \mathcal{S}_i$$

上記の２つ（３つだが、定義１と２は同じ）の定義による算術式の集合は等しい。

命題 1

$$\mathcal{T} = \mathcal{S}$$

証明 この講義では省略する。

2.2.2 算術式に関する関数の帰納的定義

上記で述べたように算術式は帰納的に定義されるが、これに対応して算術式を引数にとる関数を帰納的に定義することができる。例えば、算術式の中にある定数すべてからなる集合を返す関数は以下のように帰納的に定義できる。

$$\begin{aligned}&\mathit{consts}(\texttt{true}) = \{\texttt{true}\} \\ &\mathit{consts}(\texttt{false}) = \{\texttt{false}\} \\ &\mathit{consts}(\texttt{0}) = \{\texttt{0}\} \\ &\mathit{consts}(\texttt{succ t}_1) = \mathit{consts}(\texttt{t}_1) \\ &\mathit{consts}(\texttt{pred t}_1) = \mathit{consts}(\texttt{t}_1) \\ &\mathit{consts}(\texttt{iszero t}_1) = \mathit{consts}(\texttt{t}_1) \\ &\mathit{consts}(\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3) = \mathit{consts}(\texttt{t}_1) \cup \mathit{consts}(\texttt{t}_2) \cup \mathit{consts}(\texttt{t}_3)\end{aligned}$$

また、算術式の大きさを返す関数は以下のように帰納的に定義できる。

$$\begin{aligned}&\mathit{size}(\texttt{true}) = 1 \\ &\mathit{size}(\texttt{false}) = 1 \\ &\mathit{size}(\texttt{0}) = 1 \\ &\mathit{size}(\texttt{succ t}_1) = \mathit{size}(\texttt{t}_1) + 1 \\ &\mathit{size}(\texttt{pred t}_1) = \mathit{size}(\texttt{t}_1) + 1 \\ &\mathit{size}(\texttt{iszero t}_1) = \mathit{size}(\texttt{t}_1) + 1 \\ &\mathit{size}(\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3) = \mathit{size}(\texttt{t}_1) + \mathit{size}(\texttt{t}_2) + \mathit{size}(\texttt{t}_3) + 1\end{aligned}$$

また、算術式の深さを返す関数は以下のように定義できる。

$$
\begin{aligned}
&depth(\texttt{true}) = 1 \\
&depth(\texttt{false}) = 1 \\
&depth(\texttt{0}) = 1 \\
&depth(\texttt{succ t}_1) = depth(\texttt{t}_1) + 1 \\
&depth(\texttt{pred t}_1) = depth(\texttt{t}_1) + 1 \\
&depth(\texttt{iszero t}_1) = depth(\texttt{t}_1) + 1 \\
&depth(\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3) = max(depth(\texttt{t}_1), depth(\texttt{t}_2), depth(\texttt{t}_3)) + 1
\end{aligned}
$$

$size$ と $consts$ には以下のような関係がある。

命題 2

$$|consts(\texttt{t})| \leq size(\texttt{t})$$

証明 算術式の構造に関する帰納法で証明する。（補助資料参照）

定理 1 (算術式の構造に関する帰納法)

$$
\begin{aligned}
&P(\texttt{true}) \wedge P(\texttt{false}) \wedge P(\texttt{0}) \\
&\wedge \forall \texttt{t}_1 \in \mathcal{T}.\ \{P(\texttt{t}_1) \Rightarrow P(\texttt{succ t}_1)\} \\
&\wedge \forall \texttt{t}_1 \in \mathcal{T}.\ \{P(\texttt{t}_1) \Rightarrow P(\texttt{pred t}_1)\} \\
&\wedge \forall \texttt{t}_1 \in \mathcal{T}.\ \{P(\texttt{t}_1) \Rightarrow P(\texttt{iszero t}_1)\} \\
&\wedge \forall \texttt{t}_1, \texttt{t}_2, \texttt{t}_3 \in \mathcal{T}.\ \{P(\texttt{t}_1) \wedge P(\texttt{t}_2) \wedge P(\texttt{t}_3) \Rightarrow P(\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3)\} \\
&\Rightarrow \forall \texttt{t}_1 \in \mathcal{T}.\ P(\texttt{t}_1)
\end{aligned}
$$

算術式の構造に関する帰納法については整礎帰納法の特別の場合であり、それを以下で説明する。

2.3 整礎帰納法 (well-founded induction)

上記でのべた数学的帰納法や構造帰納法は整礎帰納法の特別な場合である。整礎帰納法は整礎な順序が定義されている集合の要素について成り立つ性質を証明する際に用いる。整礎帰納法を理解すれば必要に応じて様々な帰納法を自分で作り上げて使うことができる。