3　構文の定義方法 (BNF と帰納的定義)

プログラムや論理式などの文法 (構文) を定義するため，BNF (Backus Normal Form, Backus-Naur Form) による方法がよく使われる．

例 5 (簡単なプログラム言語のプログラム)

```
<プログラム> ::= <宣言部> <関数定義部>
<宣言部> ::= "" | <宣言> ";" <宣言部>
<関数定義部> ::= "" | <関数定義> ";" <関数定義部>
<関数定義> ::= <型宣言> <関数名> "(" <引数部> ")" <ブロック>
<ブロック> ::= "{" <宣言部> <文の列> "}"
...
```

BNF はコンパクトな表現で、無限集合を定義できるという利点があるが，比較的単純な文法しか表現できない．

BNF による定義法を含む，より一般的な定義方法として帰納的定義 (inductive definition) がある．

例 6 葉が自然数のラベルを持つ 2 分木に対する帰納的定義

$$\frac{n : \text{自然数}}{\mathit{leaf}(n) : 2\text{ 分木}}\ \mathit{leaf} \qquad \frac{T1 : 2\text{ 分木}\quad T2 : 2\text{ 分木}}{\mathit{node}(T1, T2) : 2\text{ 分木}}\ \mathit{node}$$

この定義は，leaf と node の 2 つの定義節から構成される．leaf 規則は,「n が自然数であれば leaf(n) が 2 分木である」ということを意味し,「初めの要素」を導入するものである．node 規則は,「$T1, T2$ が 2 分木であれば，node$(T1, T2)$ が 2 分木である」ということを意味し,「ある要素から次の要素を作る操作」を導入するものである．

例: 2 分木 (葉が自然数のラベルを持つもの) の帰納的定義

$$\frac{n : \text{自然数}}{\texttt{leaf}(n) : 2\text{ 分木}}\ \texttt{leaf} \qquad \frac{T1 : 2\text{ 分木}\quad T2 : 2\text{ 分木}}{\texttt{node}(T1, T2) : 2\text{ 分木}}\ \texttt{node}$$

上記の 2 つのルールが 2 分木に対する帰納的定義は，(明示的には書かないが，いつでも) 以下のルールを含む．

上記の 2 つのルールを有限回適用してできたものだけが，2 分木である．

この最後のルールは，帰納法 (2 分木に関する構造帰納法) として表現される．

例 7 $P(T)$ を 2 分木 T に関する命題とする．すると，以下の帰納法が成立する．

$$\frac{\begin{array}{cc} n : \text{自然数} & P(T1)\quad P(T2) \\ \vdots & \vdots \\ P(\mathit{leaf}(n)) & P(\mathit{node}(T1, T2)) \end{array}}{\forall T : 2\text{ 分木}.P(T)}\ \text{帰納法}$$

このルールは,「すべての 2 分木 T に対して P という性質が成立することを証明するためには，(1) n が自然数であるという仮定のもとで leaf(n) に対して P が成立すること，(2) $T1$ と $T2$ に対して P が成立するという仮定のもとで node$(T1, T2)$ に対して P が成立すること，の 2 つを証明すればよい，というものである．これが，自然数に対する数学的帰納法の拡張になっていることは容易に想像できるだろう．