『離散構造』 6 章 (帰納) の演習問題の解答

2008.2.29

問題 1 (集合の帰納的定義) $\Sigma = \{a, b, c\}$ として，Σ^* の部分集合として以下のものを帰納的に定義せよ．

(a) 「同じ文字が 2 文字以上連続して現れない文字列の集合.」

この集合を A とすると，以下のように定義できる．

- $\Lambda \in A$.
- $x \in \Sigma \Rightarrow x \in A$.
- $(s \in A \wedge s \neq \Lambda \wedge x \in \Sigma \wedge strhead(s) \neq x) \Rightarrow xs \in A$.

注. 2/29 の演習では $strhead$ という関数を $head$ と書いているのに気付かず,「正解」と言ってしまった．また、はっきり記憶していないが、上記の 3 番目の節の $s \neq \Lambda$ という部分も書いていなかったかもしれない。$head$ は (本授業のテキストでは) リストに対する部分関数であり，$strhead$ が文字列に対する部分関数であるので、上記で使うべきは $strhead$ でないといけない。さらに、$strhead$ は部分関数であり、$s = \Lambda$ に対しては値が定義されていないので、上記のように書くべきである。ここでお詫びして修正する．

集合 A は，以下のように定義することもできる．

- $\Lambda \in A$.
- $x \in \Sigma \Rightarrow x \in A$.
- $(ys \in A \wedge x \in \Sigma \wedge y \neq x) \Rightarrow xs \in A$.

(b) 回文の集合．ただし，回文とは「しんぶんし」「わたしまけましたわ」のように，上下どちらから読んでも同じ文 (語) のことである．空文字列 Λ も回文であるとする．

この集合を P とすると，以下のように定義できる．

- $\Lambda \in P$.
- $x \in \Sigma \Rightarrow x \in P$.
- $(x \in \Sigma \wedge s \in P) \Rightarrow xsx \in P$.

問題 2 (関数の帰納的定義)

(a) 自然数のリストが与えられた時，その総和を返す関数 $\mathrm{sum} : List_{\mathcal{N}} \to \mathcal{N}$.

$$\mathrm{sum}(L) = \begin{cases} 0 & \text{if } \ L = \langle\rangle \\ x + \mathrm{sum}(M) & \text{if } \ L = \mathrm{cons}(x, M) \end{cases}$$

(b) 自然数のリストが与えられた時，その末尾の要素を返す関数 $\mathrm{last} : List_{\mathcal{N}} \to \mathcal{N}$ (引数が空リストのときは 0 を返すとする．)

$$\mathrm{last}(L) = \begin{cases} 0 & \text{if } \ L = \langle\rangle \\ x & \text{if } \ L = \mathrm{cons}(x, M) \wedge L = \langle\rangle \\ \mathrm{last}(M) & \text{if } \ L = \mathrm{cons}(x, M) \wedge L \neq \langle\rangle \end{cases}$$

(2008/3/8 訂正) 上記の関数の定義には誤殖があることを水野君が指摘してくれた。右辺の $L = \langle\rangle$ と $L \neq \langle\rangle$ はともに、L ではなく M でなければならない。確かにその通りであるので、以下のように訂正する。(水野君に感謝!)

$$\mathrm{last}(L) = \begin{cases} 0 & \text{if } L = \langle\rangle \\ x & \text{if } L = \mathrm{cons}(x, M) \wedge M = \langle\rangle \\ \mathrm{last}(M) & \text{if } L = \mathrm{cons}(x, M) \wedge M \neq \langle\rangle \end{cases}$$

これが私が用意していた答えだが，演習で出てきた、次の解答も正解である．

$$\mathrm{last}(L) = \begin{cases} 0 & \text{if } L = \langle\rangle \\ \mathrm{head}(\mathrm{rev}(M)) & \text{if } L = \mathrm{cons}(x, M) \end{cases}$$

ここで rev と head はテキストで定義済みの関数，部分関数とする．

(c) 長さが同じ実数のリストが 2 つ与えられた時，それらをベクトルと見なして，内積を返す関数 ip : $List_{\mathcal{R}} \times List_{\mathcal{R}} \to \mathcal{R}$．(2 つのリストの長さが異なるときは，0 を返す．)
ここでは補助関数 f を使うことにする。

$$\mathrm{ip}(L_1, L_2) = \begin{cases} f(L_1, L_2) & \text{if } \mathrm{length}(L_1) = \mathrm{length}(L_2) \\ 0 & \text{otherwise} \end{cases}$$

otherwise という単語は「その他」の場合を指す。

$$\mathrm{f}(L_1, L_2) = \begin{cases} 0 & \text{if } L_1 = \langle\rangle \\ x * \mathrm{head}(L_2) + \mathrm{f}(M, \mathrm{tail}(L_2)) & \text{if } L_1 = \mathrm{cons}(x, M) \end{cases}$$

(d) ラベルつき 2 分木 ($\mathrm{BinTree}_A$) の高さを返す関数 height : $\mathrm{BinTree}_A \to \mathcal{N}$. ただし，木の高さとは，根から葉までの道の長さの最大値である．(根だけからなる木の高さを 0 とせよ．)

$$\mathrm{height}(T) = \begin{cases} 0 & \text{if } T = \circ \\ 1 + \mathrm{height}(L) & \text{if } T = \langle L, x\rangle \\ 1 + max(\mathrm{height}(L), \mathrm{height}(R)) & \text{if } T = \langle L, x, R\rangle \end{cases}$$

ここで max(x,y) は自然数 x,y のうち小さくない方を取る関数である。

問題 3 (自然数に対する帰納法) 自然数 a, b, c に対して $a \equiv b \mod c$ という表記は,「$a - b$ が c で割り切れる」ことを表す.

p が素数ならば，任意の自然数 n に対して $n^p \equiv n \mod p$ であることを n に関する帰納法で証明せよ．ただし，$(n+1)^p \equiv n^p + 1 \mod p$ であることを証明せずに使ってよい.

答え (証明).

「p が素数ならば，任意の自然数 n に対して $n^p \equiv n \mod p$ である」ことを n に関する帰納法で証明する。

(base case) $n = 0$ のとき、$n^p = 0$ であり、$0 \equiv 0 \mod p$ なので、$n = 0$ に対して、「。。。」は成立する。

(induction step) $n = k$ のとき、「。。。」が成立すると仮定する (帰納法の仮定)。$n = k + 1$ のときも「。。。」が成立することを示したい。つまり、$(k+1)^p - (k+1)$ が p の倍数であればよい。

問題文に、$(n+1)^p \equiv n^p + 1 \mod p$ と書いてあるので、$(k+1)^p = k^p + 1 + ap$ となる自然数 a が存在する。そこで、

$$(k+1)^p - (k+1) = (k^p + 1) + ap - (k+1) = k^p + ap - k$$

となり、帰納法の仮定より $k^p - k$ は p の倍数であるので、この式は p の倍数である。

以上から、全ての自然数 n に対して、上記の「。。。」は成立する。

問題 4 (リストに対する帰納法)

lengthと $\oplus$ をテキスト 48 ページの関数とするとき，任意の $L_1 \in List_A$, $L_2 \in List_A$ に対して，以下のことを証明せよ.

$$\mathrm{length}(L_1 \oplus L_2) = \mathrm{length}(L_1) + \mathrm{length}(L_2)$$

この問題の解答はテキストに詳細に書いてあるので、そちらを参考にしてほしい。

問題 5 (関数の「再帰」的定義，余力のある人のみ)

以下のように関数 $f : \mathcal{N} \to \mathcal{N}$ を定義しようとしたが，これは講義で説明した帰納的な関数定義の形にはなっていない．どのような点が帰納的定義になっていないかを説明せよ．また，それにもかかわらず，以下の等式を満たす関数 $f : \mathcal{N} \to \mathcal{N}$ が存在することを示せ.

$$f(n) = \begin{cases} n - 10 & (n > 100 \text{ の時}) \\ f(f(n+11)) & (0 \leq n \leq 100 \text{ の時}) \end{cases}$$

答. 関数 f の帰納的定義では、「n が最も基本的な値の場合」には、f の値を (f 自身を使わずに) 定義し、「n がそれ以外の (一般的な) 値の場合」には、f の値を使って定義してもよい。ただし、後者の場合、使う f の引数は、現在の n より簡単な値のみである。

さて、上記の関数 f は、いろいろな点で、帰納的定義の原則を破っている。

- $n > 100$ を「最も基本的な値の場合」と見なすことはできない。(自然数は小さいものから順番に構成されるので、$n < 100$ なら良いが、$n > 100$ では、基本的ではない。)
- $0 \leq n \leq 100$ の場合を「一般的な値の場合」と見なしたとき、f の定義の中で f を使ってもよいが、その引数は n より簡単な値でなければならない。しかし、上記の f の場合、$f(f(n+11))$ となっていて、外側の f の引数は $f(n+11)$ である。これは、n より簡単な値である保証がないので、帰納的定義になっていない。さらに、内側の f も引数が $n+11$ であり、n より簡単とは言えない。

このように、上の形で書いたとしても、これが「関数」の定義になっているかどうかは、はっきりしない。(部分関数であることは明らかだが、$f(n)$ を何回も展開すると、有限回で展開が停止するかどうか、保証がない。)

しかし、実は、上記の関数は「McCarthy の 91 関数」と呼ばれており、以下の関数と同じであることが知られている。(興味がある人は、理由を考えてみるとよい。)

$$f(n) = \begin{cases} 91 & (n \leq 101 \text{ の時}) \\ n - 10 & (n > 102 \text{ の時}) \end{cases}$$

この関数は明らかに「自然数から自然数への関数」である。

[補足] John McCarthy は、Lisp というプログラム言語や、プログラム理論、人工知能 (知識推論) の分野で極めて著名な計算機科学者である。世界の Computer Science をリードする大学であるアメリカ・スタンフォード大学の先生を長年勤められ、少し前に引退された。