『離散構造』 6 章 (帰納) の演習問題 (亀山)

問題 1 (文字列の集合の帰納的定義)

(a) 「a,b,c の 3 文字からなる文字列で，a が (ちょうど) 3 回出現する文字列」の集合を，帰納的に定義せよ．

解答例. いろいろな定義があるが一例は以下の通り．

まず,「a が 1 回もあらわれない文字列」の集合 S_1 を定義する．

- $\Lambda \in S_1$.
- $s \in S_1 \Rightarrow bs \in S_1$.
- $s \in S_1 \Rightarrow cs \in S_1$.

次に,「a が 3 回出現する文字列」の集合 S_2 を定義する．(この定義は，実際には帰納的定義でなく，通常の定義で済ませることもできる．)

- $s_1 \in S_1 \wedge s_2 \in S_1 \wedge s_3 \in S_1 \wedge s_4 \in S_1 \Rightarrow s_1 a s_2 a s_3 a s_4 \in S_2$.

(b) 「a,b,c の 3 文字からなる文字列で，$a^n b^n c^n$ の形の文字列」の集合を，帰納的に定義せよ．ただし，n は自然数，a^n は a が n 回連続して出てくる文字列とする．

解答例. これも，いろいろな定義があるが一例は以下の通り．

まず,「$a^n b^n$ の形の文字列」の集合 S_3 を次のように帰納的に定義する．

- $\Lambda \in S_3$.
- $s \in S_3 \Rightarrow asb \in S_3$.

「c^n の形の文字列」の集合 S_4 を次のように帰納的に定義する．

- $\Lambda \in S_4$.
- $s \in S_4 \Rightarrow cs \in S_4$.

また，文字列 s の長さ $|s|$ を表す関数を次のように定義する．

$$|s| = \begin{cases} 0 & \text{if } s = \Lambda \\ |s_1| + 1 & \text{if } s = x s_1 \end{cases}$$

これらを使って，求める集合 S_5 を次のように定義する．

- $s_3 \in S_3 \wedge s_4 \in S_4 \wedge |s_3| = 2|s_4| \Rightarrow s_3 s_4 \in S_5$.

問題 2 (式の集合の帰納的定義) 文字列としての、type-1 の式と type-2 の式を、次のように帰納的に定める．(ただし、これらの定義で $e_1 + e_2$ 等は、実際の数の加算ではなく、「e_1 という文字列のあとに "+" という記号をかき、e_2 という文字列をかいたもの、という意味である。つまり 3+5 は 8 ではない。)

type-1 の式の帰納的定義:

- 0 から 9 までの整数は type-1 の式である．
- e_1, e_2 が type-1 の式であれば，$e_1 + e_2$ は type-1 の式である．
- e_1, e_2 が type-1 の式であれば，$e_1 * e_2$ は type-1 の式である．

type-2 の式 (補助的に「type-2a の式」を定義している)

- 0 から 9 までの整数は type-2a の式である．
- e_1, e_2 が type-2a の式であれば，$e_1 * e_2$ は type-2a の式である．

- e_1 が type-2a の式であれば，e_1 は type-2 の式である．
- e_1, e_2 が type-2 の式であれば，$e_1 + e_2$ は type-2 の式である．

問 2-a. 3+5*7 が type-1 の式であることを導出せよ。(具体的に、どの規則をどういう風に適用して言ったか、その過程を全部書きなさい。) また，これが、type-2 の式であることの導出も書きなさい。異なる導出が 2 個以上あるときは，その全てを列挙せよ．

解答例. type-1 については 2 通りの導出，type-2 については 1 通りの導出がある．(e が type-N の式であることを $e \in t_N$ と書くことにする．)

(type-1 の導出 1)

(1) $3 \in t_1$ (定義の 1 行目より)

(2) $5 \in t_1$ (定義の 1 行目より)

(3) $3 + 5 \in t_1$ (上記 (1),(2) と定義の 2 行目より)

(4) $7 \in t_1$ (定義の 1 行目より)

(5) $3 + 5 * 7 \in t_1$ (上記 (3),(4) と定義の 3 行目より)

(type-1 の導出 2)

(1) $5 \in t_1$ (定義の 1 行目より)

(2) $7 \in t_1$ (定義の 1 行目より)

(3) $5 * 7 \in t_1$ (上記 (1),(2) と定義の 3 行目より)

(4) $3 \in t_1$ (定義の 1 行目より)

(5) $3 + 5 * 7 \in t_1$ (上記 (3),(4) と定義の 2 行目より)

(type-2 の導出 1)

(1) $5 \in t_{2a}$ (type-2a の定義の 1 行目より)

(2) $7 \in t_{2a}$ (type-2a の定義の 1 行目より)

(3) $5 * 7 \in t_{2a}$ (上記 (1),(2) と type-2a の定義の 2 行目より)

(4) $5 * 7 \in t_2$ (上記 (3) と type-2 の定義の 1 行目より)

(5) $3 \in t_{2a}$ (type-2a の定義の 1 行目より)

(6) $3 \in t_2$ (上記 (5) と type-2 の定義の 1 行目より)

(7) $3 + 5 * 7 \in t_2$ (上記 (4),(6) と type-2 の定義の 2 行目より)

問 2-b. 3*5*7 が type-1 の式であることの導出，また，type-2 の式であることの導出を書きなさい．ただし，異なる導出が 2 個以上あるときは，その全てを列挙せよ．

解答例. type-1, type-2 とも 2 通りの導出がある．ここでは，type-1 のみ示す．

(type-1 の導出 1)

(1) $3 \in t_1$ (定義の 1 行目より)

(2) $5 \in t_1$ (定義の 1 行目より)

(3) $3 * 5 \in t_1$ (上記 (1),(2) と定義の 3 行目より)

(4) $7 \in t_1$ (定義の 1 行目より)

(5) $3 * 5 * 7 \in t_1$ (上記 (3),(4) と定義の 3 行目より)

(type-1 の導出 2)

(1) $5 \in t_1$ (定義の 1 行目より)

(2) $7 \in t_1$ (定義の 1 行目より)

(3) $5 * 7 \in t_1$ (上記 (1),(2) と定義の 3 行目より)

(4) $3 \in t_1$ (定義の 1 行目より)

(5) $3 * 5 * 7 \in t_1$ (上記 (3),(4) と定義の 3 行目より)

問 2-c. type-1 の式の「長さ」をあらわす関数 len をうまく定義したい。たとえば、$len(3) = len(5) = 1$ であり、$len(3 * 5 * 7) = 5$ である。関数 len を帰納的に定義せよ.

解答例. type-1 の式 e の長さ $len(e)$ は以下のように定義できる.(ただし，下の方の「補足」を参照のこと.)

$$len(e) = \begin{cases} 1 & \text{if } e = 0,1,2,3,4,5,6,7,8,9 \\ len(e_1) + len(e_2) + 1 & \text{if } e = e_1 + e_2 \\ len(e_1) + len(e_2) + 1 & \text{if } e = e_1 * e_2 \end{cases}$$

ただし，この定義の条件部分 (if のあと) に出現する「$e_1 + e_2$」は文字列としての e_1, $+$, e_2 を連接した文字列をあらわし，左辺 ($len(e_1) + len(e_2)$ など) の $+$ は，自然数の加算をあらわす.

補足. (本問は，出題のしかたが，若干いい加減だったため，ここの補足に書くような，面倒な状況が生じている．これは，1 年生の「離散構造」の範囲をこえるので，今回は特に気にしなくてよいが，興味がある人は読んでほしい．)

実は，上記の「定義」が本当に関数の定義になっているかどうかは，検討を要する．というのは，上記の定義は，type-1 の式を，その導出のされかたによって分類しているが，type-1 の式は 2 通り以上で導出されるから，導出のされかたが異なると，その導出のされかたによって，関数 len の適用結果が異なる可能性があるからである．実際には，len の定義では「type-1 の式の集合から自然数への関数」になるので，特に困らないが，具体的に「困る」例として，以下のようなものがある.

$$f(e) = \begin{cases} 1 & \text{if } e = 0,1,2,3,4,5,6,7,8,9 \\ f(e_1) + f(e_2) & \text{if } e = e_1 + e_2 \\ f(e_1) * f(e_2) & \text{if } e = e_1 * e_2 \end{cases}$$

このような f に対しては，$f(1 + 1 * 1 + 1)$ の値は 1 つに決まらず，3 になったり 4 になったりする．これは直感的には当然であり，足し算とかけ算を含む式の値は，かっこのつけかたにより異なってくるのである.

問 2-d. どんな type-1 の式 e に対しても，$len(e)$ は奇数であることを証明しなさい.

解答例. type-1 の式 e に関する帰納法を使う.

(Base case) $e = 0,1,2,3,4,5,6,7,8,9$ のいずれかのとき．このとき $len(e) = 1$ であり，奇数である.

(Step-1) $e = e_1 + e_2$ のとき．帰納法の仮定より，$len(e_1)$ と $len(e_2)$ は奇数である．よって，$len(e) = len(e_1) + len(e_2) + 1$ は，3 つの奇数の和となり，奇数である.

(Step-2) $e = e_1 * e_2$ のとき．帰納法の仮定より，$len(e_1)$ と $len(e_2)$ は奇数である．よって，上と同様に，$len(e) = len(e_1) + len(e_2) + 1$ も奇数である.

以上より，帰納法を使って，全ての type-1 の式 e に対して，$len(e)$ が奇数であることが証明できた.

問 2-e. (発展課題) 「type-3 の式」の集合は以下を満たす。これを定義せよ.(1) type-1 の式の集合と type-3 の式の集合は等しい.(2) type-3 の定義には曖昧さがない。つまり、「e が type-3 の式であることの導出」は，どんな e に対してもたかだか 1 つである.

解答例. type-1 と type-2 の定義を比較すると，type-2 の方が「曖昧さ」は減少している．つまり，type-2 の定義では，かけ算と足し算が別のレベルになっているので，$1+2*3$ のような式では，必ず $2*3$ が先に結合し，そのあと 1 あ結合するような導出しかない．しかし，type-2 もまだ曖昧さがあり，$1+2+3$ のように足し算同士 (あるいはかけ算同士) がつながっているとき,「左から結合」とも「右から結合」とも決まっていないので，曖昧さが生じる．そこで,「左から結合」することにして以下の定義を考えた．

type-3a の式 (type-3 の式を定義するための，補助的な定義):

- 0 から 9 までの整数は type-3a の式である．
- e_1 が type-3a の式で，x が 0 から 9 までの整数であれば，$e_1 * x$ は type-3a の式である．

type-3 の式の定義:

- e_1 が type-3a の式であれば，e_1 は type-3 の式である．
- e_1 が type-3 の式で，e_2 が type-3a の式であれば，$e_1 + e_2$ は type-3 の式である．

このように定義した，type-3 の式の集合が，問題文の性質 (1),(2) を満たす理由は，各自で考えられたい．

問題 3 (関数の帰納的定義)

自然数のリストの集合を S とする．関数 $f : S \to \mathcal{N}$ を次のように帰納的に定義する．(右辺の「+」は文字列ではなく、自然数上の本当の加算である。)

$$f(L) = \begin{cases} 0 & \text{if } L = \langle\rangle \\ x + f(L_1) & \text{if } L = \text{cons}(x, L_1) \end{cases}$$

問 3-a. $f(\langle 1,2,3\rangle)$ つまり、$f(\text{cons}(1,\text{cons}(2,\text{cons}(3,\langle\rangle))))$ を計算しなさい.

解答例. 以下の通り.

$$\begin{aligned} f(\text{cons}(1,\text{cons}(2,\text{cons}(3,\langle\rangle)))) &= 1 + f(\text{cons}(2,\text{cons}(3,\langle\rangle))) && f \text{ の定義の 2 行目より} \\ &= 1 + (2 + f(\text{cons}(3,\langle\rangle))) && f \text{ の定義の 2 行目より} \\ &= 1 + (2 + (3 + f(\langle\rangle))) && f \text{ の定義の 2 行目より} \\ &= 1 + (2 + (3 + 0)) && f \text{ の定義の 1 行目より} \\ &= 6 \end{aligned}$$

問 3-a. 任意の $L_1, L_2 \in S$ に対して $f(L_1 \oplus L_2) = f(L_1) + f(L_2)$ が成立することを帰納法で証明しなさい. ただし、$\oplus$ はリストの連接 (concatenation) である (テキストを参照のこと)。

解答例. リスト L_1 に関する帰納法で証明する.

(Base case) $L_1 = \langle\rangle$ のとき.

$\oplus$ の定義より，$f(L_1 \oplus L_2) = f(\langle\rangle \oplus L_2) = f(L_2)$ となる．また， $f(L_1) + f(L_2) = f(\langle\rangle) + f(L_2) = 0 + f(L_2) = f(L_2)$ である．よって，$f(L_1 \oplus L_2) = f(L_1) + f(L_2)$ である.

(Step) $L_1 = \text{cons}(a, L_3)$ のとき.

左辺:

$$\begin{aligned} f(L_1 \oplus L_2) &= f(\text{cons}(a, L_3) \oplus L_2) \\ &= f(\text{cons}(a, (L_3 \oplus L_2))) && \oplus \text{ の定義より} \\ &= a + f(L_3 \oplus L_2) && f \text{ の定義より} \\ &= a + (f(L_3) + f(L_2)) && \text{帰納法の仮定より} \end{aligned}$$

ここで，帰納法の仮定を使ってよいのはなぜかといえば，仮定の方では $f(L_3 \oplus \cdots)$ となっていて，L_3 は，L_1 より真に小さいので，使ってよいのである．

右辺:

$$\begin{aligned} f(L_1) + f(L_2) &= f(\mathrm{cons}(a, L_3)) + f(L_2) \\ &= (a + f(L_3)) + f(L_2) \end{aligned}$$

よって左右両辺は等しい．

以上から，帰納法により，$f(L_1 \oplus L_2) = f(L_1) + f(L_2)$ が成立する．

補足. この問題を，L_2 に関する帰納法で解こうと思うと失敗する．(このように，同じように見えても，どちらか一方だけ証明がうまくいくことも多く，試行錯誤が必要である．)