Лекция 4: Кодиране

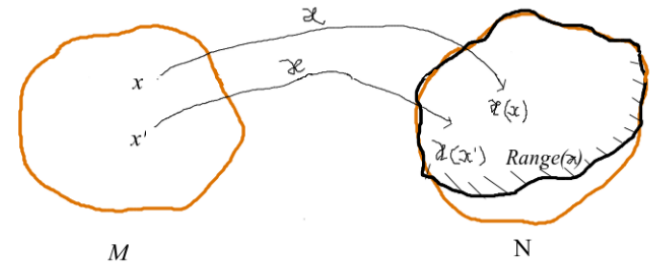
1.8 Кодиране

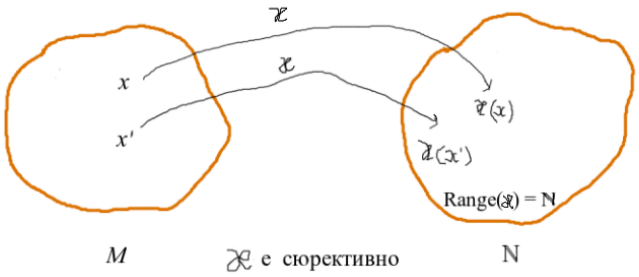
В този раздел ще дефинираме кодиране на наредени двойки, на n-торки и на крайни редици от естествени числа, посредством които по-нататък ще можем да кодираме — оператори, програми, изчисления и пр.

Нека M е произволно множество. *Кодиране* на M ще наричаме всяко инективно изображение

$$\kappa\colon\ M \longrightarrow \mathbb{N}.$$

Ясно е, че за да можем да кодираме с естествени числа едно множество M, то трябва да е най-много изброимо. Обикновено M е множество от *конструктивни обекти* — числа, низове, формули, дървета и пр.





𝒳 е сюрективно

Числото $\kappa(x)$ ще наричаме *код* на x. Инективността на κ е задължителна, с други думи, искаме всяко число да е код на *най-много* един обект. За нашите цели ще бъде удобно *всяко* число да е код на някакъв обект, затова ще се грижим конкретните кодирания, които дефинираме, да са сюрективни. И това, което е от изключителна важност — ще осигуряваме тези кодирания да са *ефективни*, което означава, най-общо казано, по кода $\kappa(x)$ да можем да възстановяваме *алгоритмично* обекта x. (Без последното условие въпросът за съществуване на кодиране на дадено множество M се решава тривиално: НДУ за съществуване на такова кодиране очевидно е изискването M да е най-много изброимо.)

1.8.1 Кодиране на наредени двойки

Оттук до края на курса с Π ще означаваме следното изображение $\Pi : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$:

$$\Pi(x, y) \stackrel{\text{деф}}{=} 2^x(2y + 1) - 1.$$

Да отбележим, че тъй като $2^x(2y+1) \geq 1$, то $\Pi(x, y) \in \mathbb{N}$ за всички естествени x и y. Да се убедим, че това изображение е кодиране на $\mathbb{N}\times\mathbb{N}$, като при това то е сюрективно. Да си спомним, че когато едно изображение е едновременно инективно и сюрективно, то се нарича *биективно* (или *биекция*).

Твърдение 1.18. Изображението Π: $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ е биекция.

Доказателство. Трябва да покажем, че за всяко $z \in \mathbb{N}$ съществуват единствени (x, y) : $\Pi(x, y) = z$.

Ще използваме, че всяко *положително* естествено число се представя по единствен начин във вида $2^x(2y + 1)$. Сега да вземем произволно $z \in \mathbb{N}$. Тогава $z + 1 > 0$ и следователно за *единствени* естествени x и y ще имаме, че

$$2^x(2y + 1) = z + 1.$$

Но тогава $2^x(2y + 1) - 1 = z$, т.е. $\Pi(x, y) = z$. □

Щом за всяко $z \in \mathbb{N}$ съществуват единствени x, y, за които $\Pi(x, y) = z$, то значи съществуват и *функции*, които за всяко z връщат тези x и y. Тези обратни функции обикновено се наричат *декодиращи функции*. Ние ще ги означаваме с L и R. По определение

$$L(\Pi(x, y)) = x \qquad \text{и} \qquad R(\Pi(x, y)) = y.$$

За наредената тройка (Π, L, R) ще казваме, че е *кодираща тройка* (или *кодираща схема*) Една такава тройка наричаме *примитивно рекурсивна* кодираща тройка, ако функциите, участващи в нея, са примитивно рекурсивни.

Твърдение 1.19. (Π, L, R) е примитивно рекурсивна кодираща тройка.

Доказателство. Изображението Π е примитивно рекурсивно, защото можем да го препишем като $\Pi(x, y) = 2^x(2y + 1) \dot{-} 1$, като всички функции, участващи в това представяне, са примитивно рекурсивни.

Сега нека $\Pi(x, y) = z$, т.е. $2^x(2y+1)-1 = z$. Тогава $2^x(2y+1) = z+1$ и следователно $x = (z + 1)_0$, с други думи

$$L(z) = (z + 1)_0$$

и значи L е примитивно рекурсивна.

По-нататък, от $2^x(2y+1) = z+1$ ще имаме

$$2y+1 = \frac{z+1}{2^x} = \frac{z+1}{2^{L(z)}},$$

и следователно

$$y = \frac{\frac{z+1}{2^{L(z)}} - 1}{2}.$$

Сега примитивната рекурсивност на R следва от това, че R можем да препишем като суперпозицията

$$R(z) = y = qt(2, qt(2^{L(z)}, z+1) \dot{-} 1),$$

в която всички участващи функции са примитивно рекурсивни. □

Забележка. Да обърнем внимание на един на пръв поглед дребен факт, който обаче ще е важен при някои дефиниции по рекурсия, а именно, това, че за всяко z:

$$L(z) \leq z \qquad \text{и} \qquad R(z) \leq z,$$

като равенство се достига само при $z = 0$ и $z = 1$. За тези две числа имаме, че:

$$\Pi(0,0) = 2^0.(2.0+1) - 1 = 0 \quad \text{и} \quad \Pi(1,0) = 2^1.(2.0+1) - 1 = 1.$$

1.8.2 Кодиране на $\mathbb{N}^n$

За всяко фиксирано $n \geq 1$ ще въведем изображение

$$\Pi_n \colon \mathbb{N}^n \longrightarrow \mathbb{N},$$

за което ще докажем, че е кодиране на наредените n-торки от естествени числа. Дефиницията е с индукция по n:

Определение 1.21.

$$\Pi_1(x_1) \stackrel{\text{деф}}{=} x_1$$
$$\Pi_{n+1}(x_1, \ldots, x_{n+1}) \stackrel{\text{деф}}{=} \Pi(\Pi_n(x_1, \ldots, x_n), x_{n+1}). \tag{1.8}$$

От тази дефиниция веднага получаваме, че

$$\Pi_2(x_1, x_2) \stackrel{\text{деф}}{=} \Pi(\Pi_1(x_1), x_2) = \Pi(x_1, x_2),$$

с други думи, Π_2 съвпада с кодирането Π на наредените двойки естествени числа, което въведохме по-горе.

В случай, че се питате защо не дефинирахме кодирането по този начин:

$$\Pi_{n+1}(x_1, \ldots, x_{n+1}) \;=\; \Pi(x_1, \Pi_n(x_2, \ldots, x_{n+1})) :$$

ами да, може и така да се дефинира, но ние предпочетохме другото определение, защото има малки технически предимства ⌣̈.

Най-напред да съобразим, че Π_n наистина е кодиране на $\mathbb{N}^n$, при това — отново примитивно рекурсивно.

Твърдение 1.20. За всяко $n \geq 1$ изображението $\Pi_n : \mathbb{N}^n \longrightarrow \mathbb{N}$ е биективно и примитивно рекурсивно.

Доказателство. Индукция по n. Пропускаме като очеваден случая $n = 1$ и приемаме, че за произволно $n \geq 1$, Π_n е биективно и примитивно рекурсивно. Тогава от

$$\Pi_{n+1}(x_1, \ldots, x_{n+1}) \;=\; \Pi(\Pi_n(x_1, \ldots, x_n), x_{n+1})$$

и *Твърдение* 1.19 веднага следва, че и Π_{n+1} ще е примитивно рекурсивно.

Да видим, че то е и биективно, т.е. за всяко $z \in \mathbb{N}$ съществуват единствени $(x_1, \ldots, x_{n+1})$, за които $\Pi_{n+1}(x_1, \ldots, x_{n+1}) = z$. Наистина, да изберем произволно естествено z. От *Твърдение* 1.18 знаем, че за единствена двойка числа (x, y) ще е вярно, че $\Pi(x, y) = z$. От индуктивната хипотеза, приложена за x, ще съществуват единствени $x_1, \ldots, x_n$, такива че

$$\Pi_n(x_1, \ldots, x_n) \;=\; x.$$

Но тогава

$$\Pi_{n+1}(x_1, \ldots, x_n, y) \;=\; \Pi(\Pi_n(x_1, \ldots, x_n), y) \;=\; \Pi(x, y) \;=\; z.$$

□

Щом изображението $\Pi_n : \mathbb{N}^n \longrightarrow \mathbb{N}$ е биекция, значи съществуват обратните му функции, които ще означаваме с J_i^n, $i = 1, \ldots, n$. По определение

$$J_i^n(\Pi_n(x_1, \ldots, x_n)) \;=\; x_i.$$

Тези функции сигурно ви напомнят на проектиращите функции I_i^n. Приликата не е само в означението, всъщност декодиращите J_i^n действат върху *кодовете* на n-торките точно както проектиращите I_i^n действат върху самите n-торки.

Задача 1.16. Докажете, че за всяко n и $1 \leq i \leq n$ функциите J_i^n са примитивно рекурсивни.

Решение. Индукция по n. При $n = 1$ имаме $J_1^1(z) = z$.
Да допуснем, че за някое n всички $J_i^n, 1 \leq i \leq n$, са примитивно рекурсивни. Тогава за $n+1$ ще имаме следното:

ако $1 \leq i \leq n$, то очевидно

$$J_i^{n+1}(z) \;=\; J_i^n(L(z))$$

и значи J_i^{n+1} е примитивно рекурсивна, съгласно индукционната хипотеза. Ако пък $i = n+1$, то по определение $J_{n+1}^{n+1}(z) = R(z)$ и значи отново J_{n+1}^{n+1} е примитивно рекурсивна. □

Тази задача, заедно с *Твърдение* 1.20 ни дават, че $(\Pi_n, J_1^n, \ldots J_n^n)$ е примитивно рекурсивна кодираща схема. На нас, обаче, ще ни трябва нещо по-силно от факта, че функциите $J_1^n, \ldots J_n^n$ са примитивно рекурсивни, а именно — ще ни трябва техния *явен вид*, изразен чрез декодиращите L и R. Да се убедим, че той е следният:

Твърдение 1.21. За всяко $n \geq 1$ и $1 \leq i \leq n$:

$$J_i^n \;=\; \begin{cases} R \circ L^{n-i}, & \text{ако } 1 < i \leq n \\ L^{n-1}, & \text{ако } i = 1. \end{cases}$$

Доказателство. Ясно е, че отново ще трябва да разсъждаваме с индукция относно n.

По-горе видяхме, че $J_1^1(z) = z$, т.е. $J_1^1 = I_1^1$, което се съгласува с $J_1^1 = L^0 \overset{\text{деф}}{=} I_1^1$. Така базата $n = 1$ е проверена.

Сега да приемем, че за някое $n \geq 1$ всяка от функциите J_i^n има горния вид. За $n+1$ разглеждаме следните два случая:
1 сл. $i = n+1$. От определението на Π_{n+1} имаме

$$J_{n+1}^{n+1}(z) \;=\; R(z) \;=\; R(\underbrace{L^{(n+1)-(n+1)}(z)}_{z}).$$

2 сл. $1 \leq i \leq n$. По-горе съобразихме, че $J_i^{n+1}(z) = J_i^n(L(z))$. Прилагаме индуктивната хипотеза и получаваме

$$J_i^{n+1}(z) \;=\; J_i^n(L(z)) \overset{\text{и.х.}}{=} \begin{cases} R(L^{n-i}(L(z))), & \text{ако } 1 < i \leq n \\ L^{n-1}(L(z)), & \text{ако } i = 1. \end{cases}$$

С други думи,

$$J_i^{n+1}(z) \;=\; \begin{cases} R(L^{n+1-i}(z)), & \text{ако } 1 < i \leq n \\ L^n(z), & \text{ако } i = 1, \end{cases}$$

което заедно с полученото по-горе за $i = n+1$ довършва доказателството на твърдението. □

Ще усилим още малко горното твърдение, като докажем, че всъщност $J_i^n(z)$ зависи "примитивно рекурсивно" не само от z, но и от индексите си n и i, по-точно:

Твърдение 1.22. Функцията

$$F(n,i,z) \;=\; \begin{cases} J_i^n(z), & \text{ако } n \geq 1 \;\&\; 1 \leq i \leq n \\ 0, & \text{в останалите случаи} \end{cases}$$

е примитивно рекурсивна.

Доказателство. За F можем да запишем, използвайки току-що доказаното *Твърдение* 1.21, че

$$F(n,i,z) \;=\; \begin{cases} R(L^*(n-i,z)), & \text{ако } n \geq 1 \;\&\; 1 < i \leq n \\ L^*(n-1,z), & \text{ако } n \geq 1 \;\&\; i = 1 \\ 0, & \text{в останалите случаи.} \end{cases}$$

Сега примитивната рекурсивност на F следва от факта, че итерацията запазва примитивната рекурсивност (*Твърдение* 1.17). □

1.8.3 Кодиране на $\mathbb{N}^*$

В този курс с $\mathbb{N}^*$ ще означаваме множеството на крайните *непразни* редици от естествени числа, с други думи

$$\mathbb{N}^* \;=\; \bigcup_{n=1}^{\infty} \mathbb{N}^n.$$

Да отбележим, че това означение се разминава със стандартното $A^* \overset{\text{деф}}{=} \bigcup_{n=0}^{\infty} A^n$, което включва и празния низ в A^*.

Ние няма да включваме ε в $\mathbb{N}^*$, защото идеята ни е $\mathbb{N}^*$ да е множеството от редиците, които са кодове на операторите на нашите програми (програмите ни ще бъдат просто крайни редици от оператори). И тъй като всяка програма има поне един оператор, празният низ очевидно няма да е от този вид.

Всяка *непразна* редица от естествени числа ще означаваме със счупени скобки по ето този начин:

$$\langle x_0, \ldots, x_n \rangle,\; n \geq 0.$$

Тук $n \geq 0$, т.е. започваме броенето от нула, за да осигурим, че n пробягва всички естествени числа, включително и нулата. Разбира се, всичко това е изцяло заради технически удобства.

Определение 1.22. В множеството на всички непразни редици с елементи от $\mathbb{N}$ дефинираме следното изображение $\tau\colon\ \mathbb{N}^* \longrightarrow \mathbb{N}$:

$$\tau(\langle x_0, \ldots, x_n\rangle) \ = \ \Pi(n, \Pi_{n+1}(x_0, \ldots, x_n)). \tag{1.9}$$

От определението се вижда, че ако

$$z = \tau(\langle x_0, \ldots, x_n\rangle) \stackrel{\text{деф}}{=} \Pi(\underbrace{n}_{L(z)}, \underbrace{(\Pi_{n+1}(x_0, \ldots, x_n))}_{R(z)}),$$

то $L(z)+1$ е дължината на редицата с код z, а $R(z)$ е кодът на тази редица (разглеждана като $L(z)+1$-орка). С други думи, $L(z)$ "помни" дължината на редицата с код z, докато $R(z)$ "помни" елементите на тази редица.

Най-напред да покажем, че τ е *кодиране* на $\mathbb{N}^*$, което при това покрива цялото $\mathbb{N}$, т.е. τ е биективно.

Твърдение 1.23. Изображението $\tau\colon\ \mathbb{N}^* \longrightarrow \mathbb{N}$ е биекция.

Доказателство. Трябва да видим, че за всяко $z \in \mathbb{N}$ съществува единствена редица $\langle x_0, \ldots, x_n\rangle$, такава че

$$\tau(\langle x_0, \ldots, x_n\rangle) \ = \ z.$$

Наистина, да означим $L(z)$ с n. От *Твърдение* 1.20, приложено за $n+1$, съществуват единствени $x_0, \ldots, x_n$, такива че

$$\Pi_{n+1}(x_0, \ldots, x_n) \ = \ R(z).$$

Сега вече

$$\tau(\langle x_0, \ldots, x_n\rangle) \stackrel{\text{деф}}{=} \Pi(n, \Pi_{n+1}(x_0, \ldots, x_n)) \ = \ \Pi(L(z), R(z)) \ = \ z.$$

□

Ясно е, че ако знаем редицата $\langle x_0, \ldots, x_n\rangle$, алгоритмично можем да намерим нейния код $\tau(\langle x_0, \ldots, x_n\rangle)$. Обратно, ако знаем кода $z = \tau(\langle x_0, \ldots, x_n\rangle)$ на една редица $\langle x_0, \ldots, x_n\rangle$, очевидно по него можем да възстановим дължината на тази редица и нейните елементи. Но как да формализираме това "възстановяване"? При кодирането Π_n тази идея се формализираше лесно — просто съобразихме, че декодиращите функции на Π_n са примитивно рекурсивни. Тук, обаче, за декодиращи функции очевидно не можем да говорим.

Затова въвеждаме две други функции $lh(z)$ и $mem(z,i)$, които връщат дължината на редицата и нейните елементи. Ето и точните дефиниции:

Нека $z = \tau(\langle x_0, \ldots, x_n\rangle)$. Тогава

$$lh(z) \stackrel{\text{деф}}{=} n,$$

с други думи, lh връща *дължината на редицата* с код z (или по-скоро дължината минус 1, защото дължината на редицата $\langle x_0, \ldots, x_n \rangle$ е $n+1$).

Другата функция mem (от *member*) дефинираме по следния начин (отново предполагайки, че $z = \tau(\langle x_0, \ldots, x_n \rangle)$):

$$mem(z,i) \stackrel{\text{деф}}{=} \begin{cases} x_i, & \text{ако } i \leq lh(z) \\ 0, & \text{ако } i > lh(z). \end{cases} \tag{1.10}$$

Определение 1.23. Казваме, че едно кодиране $\tau : \mathbb{N}^* \longrightarrow \mathbb{N}$ е *ефективно*, ако функциите lh и mem са рекурсивни. Ако тези функции са примитивно рекурсивни, ще казваме, че τ е *примитивно рекурсивно* кодиране.

Сега ще покажем, че нашето кодиране τ е ефективно, като при това неговите функции lh и mem са примитивно рекурсивни.

Твърдение 1.24. Декодиращите функции lh и mem на кодирането τ, дефинирано чрез (1.9), са примитивно рекурсивни.

Доказателство. За функцията "дължина" вече забелязахме, че $lh(z) = L(z)$ и значи lh е примитивно рекурсивна.

Да се убедим, че и mem е примитивно рекурсивна. Наистина, нека $z = \tau(\langle x_0, \ldots, x_n \rangle)$, т.е.

$$z = \Pi(n, \Pi_{n+1}(x_0, \ldots, x_n)).$$

Тогава $R(z) = \Pi_{n+1}(x_0, \ldots, x_n)$ и значи за всяко $x_i, 0 \leq i \leq n$ ще имаме

$$x_i = J_{i+1}^{n+1}(R(z)).$$

Тук е моментът да си спомним, че съгласно *Твърдение* 1.22, функцията $F(n,i,z) = J_i^n(z)$ е примитивно рекурсивна. Следователно за mem можем да запишем:

$$\begin{aligned} mem(z,i) &= \begin{cases} J_{i+1}^{n+1}(R(z)), & \text{ако } i \leq lh(z) \\ 0, & \text{ако } i > lh(z) \end{cases} \\ &= \begin{cases} F(L(z)+1, i+1, R(z)), & \text{ако } i \leq lh(z) \\ 0, & \text{ако } i > lh(z), \end{cases} \end{aligned}$$

откъдето се вижда, че mem е примитивно рекурсивна. □

Ще завършим този раздел с една друга дефиницията на функция-история, различна от функцията $\hat{f}$, която въведохме в предишния раздел. Тази функция-история ще означаваме с H_f.

Нека $f : \mathbb{N} \longrightarrow \mathbb{N}$ е тотална функция. *Историята* H_f на f дефинираме по следния начин:

$$H_f(x) \stackrel{\text{деф}}{=} \tau(\langle f(0), \ldots, f(x) \rangle). \tag{1.11}$$

Твърдение 1.25. Ако $f : \mathbb{N} \longrightarrow \mathbb{N}$ е примитивно рекурсивна, то и нейната история H_f е примитивно рекурсивна.

Доказателство. Искаме да напишем примитивно рекурсивна схема за H_f. За целта да видим как са свързани $H_f(x+1)$ и $H_f(x)$. Имаме

$$\begin{aligned} H_f(x+1) &= \tau(\langle f(0), \ldots, f(x), f(x+1)\rangle) \overset{\text{деф } \tau}{=} \\ &\Pi(x+1,\ \Pi_{x+2}(f(0), \ldots, f(x),\ f(x+1))) \overset{\text{деф } \Pi_{x+2}}{=} \\ &\Pi(x+1,\ \Pi(\Pi_{x+1}(f(0), \ldots, f(x)),\ f(x+1))). \end{aligned} \tag{1.12}$$

Но за $H_f(x)$ имаме:

$$H_f(x) = \tau(\langle f(0), \ldots, f(x)\rangle) = \Pi(x, \Pi_{x+1}(f(0), \ldots, f(x))),$$

откъдето

$$\Pi_{x+1}(f(0), \ldots, f(x)) = R(H_f(x)).$$

Оттук, катоизпозлваме (1.12), за $H_f(x+1)$ получаваме

$$H_f(x+1) = \Pi(x+1, \Pi(R(H_f(x)), f(x+1))).$$

Сега окончателно

$$\left|\begin{array}{l} H_f(0) = \tau(\langle f(0)\rangle) = \Pi(0, f(0)) \\ H_f(x+1) = \underbrace{\Pi(x+1, \Pi(R(H_f(x)), f(x+1)))}_{G(x, H_f(x))}, \end{array}\right.$$

където функцията $G(x,y) = \Pi(x+1, \Pi(R(y), f(x+1)))$ е примитивно рекурсивна. Следователно и H_f е примитивно рекурсивна. □

1.8.4 Задачи

Задача 1.17. Нека $\pi(x,y) = 2^x.(2y+1)$. Да дефинираме следното изображение κ, действащо върху множеството $\hat{\mathbb{N}}$ на *всички* крайни редици с елементи от $\mathbb{N}$ (вече включваме и празната редица)::

$$\kappa(\varepsilon) = 0; \qquad \kappa(\langle x_1, \ldots, x_n\rangle) = \pi(x_1, \kappa(\langle x_2, \ldots, x_n\rangle)).$$

Докажете, че κ е примитивно рекурсивно кодиране на $\hat{\mathbb{N}}$.

Доказателство. С пълна индукция по z да се убедим, че за всяко $z \in \mathbb{N}$ съществува единствена редица с код z.

Ако $z = 0$, то $\kappa(\varepsilon) \overset{\text{деф}}{=} 0$, и понеже $\pi(x,y) \geq 1$, то ε е единствената редица с това свойство.

Лесно се вижда, че π в биекция между $\mathbb{N}^2$ и $\mathbb{N}^+$. Тогава за $z > 0$ съществуват единствени x и y, за които $z = \pi(x, y)$. Понеже $y < z$, по индуктивната хипотеза съществува единствена редица $\langle x_1, \ldots, x_n \rangle$, такава че $\kappa(\langle x_1, \ldots, x_n \rangle) = y$. Тогава е ясно, че

$$\kappa(\langle x, x_1, \ldots, x_n \rangle) = \pi(x, \kappa(\langle x_1, \ldots, x_n \rangle)) = \pi(x, y) = z.$$

Нека l и r са декодиращите за кодирането π. (Тъй като 0 не е в областта от стойности на π, да приемем, че $l(0) = r(0) = 0$). Ясно е, че l и r са примитивно рекурсивни.

Сега за функцията $lh(z)$, даваща дължината на редицата с код z, ще имаме, че:

$$\begin{cases} lh(0) = 0 \\ lh(z+1) = 1 + lh(r(z+1)). \end{cases}$$

Тъй като при $z > 0$ $r(z) < z$, то това е дефиниция с пълна рекурсия и следователно $lh(z)$ е примитивно рекурсивна.

Нека $\kappa(\langle x_1, \ldots, x_n \rangle) = z$. Да дефинираме mem като:

$$mem(z, i) \overset{\text{деф}}{=} \begin{cases} x_i, & \text{ако } z > 0 \ \& \ 1 \leq i \leq lh(z) \\ 0, & \text{иначе} \end{cases}$$

Нека $z > 0$ и $1 < i \leq lh(z)$. Тогава очевидно i-тият елемент на редицата с код z е $i-1$-ви елемент на редицата с код $r(z)$. Значи за mem ще имаме следната рекурсивна връзка:

$$mem(z, i) = \begin{cases} mem(r(z), i-1), & \text{ако } z > 0 \ \& \ 1 < i \leq lh(z) \\ l(z), & \text{ако } z > 0 \ \& \ i = 1 \\ 0, & \text{в останалите случаи.} \end{cases}$$

Има няколко начина да се убедим, че функцията mem е примитивно рекурсивна. Най-краткият е да забележим, че тя се дефинира с *вложена* рекурсия и да се възползваме от *Задача* 1.13, която казва, че тази рекурсия се изразява чрез примитивна рекурсия.

Другият начин се основава на наблюдението, че при рекурсивното обръщение *и двата* аргумента на mem намаляват. Да разгледаме *представящата* $\hat{mem}$ на mem, дефинирана като:

$$\hat{mem}(t) \ = \ mem(L(t), R(t))$$

Тогава рекурсивното обръщение

$$mem(z, i) \ \longrightarrow \ mem(r(z), i-1)$$

ще се превърне в

$$m\hat{e}m(t) \longrightarrow m\hat{e}m(\Pi(r(L(t)), R(t) - 1)).$$

Така вече ще имаме $t > \Pi(r(L(t)), R(t) - 1)$. Това означава, че можем да напишем схема за пълна рекурсия за $m\hat{e}m$ и значи тя ще е примитивно рекурсивна. Оттук, поради $mem(z, i) = m\hat{e}m(\Pi(z, i))$, ще имаме, че и mem е примитивно рекурсивна.

Но може би най-краткият начин да се убедим в примитивната рекурсивност на mem е да забележим, че можем да я изразим и *явно* по следния начин:

$$mem(z, i) = \begin{cases} l(r^{i-1}(z)), & \text{ако } z > 0 \ \& \ 1 \leq i \leq lh(z) \\ 0, & \text{иначе.} \end{cases}$$

Доказателството е с рутинна индукция по $1 \leq i \leq lh(z)$. Случаят $i = 1$ е очевиден, а допускайки, че за $1 \leq i < lh(z)$ това е така, за $i + 1$ ще имаме:

$$mem(z, i) = mem(r(z), i - 1) \overset{\text{и.х.}}{=} l(r^{(i-1)-1}(r(z))) = l(r^{i-1}(z)).$$

□

Задача 1.18. Да означим с Fin множеството от всички едноместни крайни функции в естествените числа. Изображението $\kappa : Fin \to N^+$ се дефинира като:

$$\kappa(\theta) = \begin{cases} 1, & \text{ако } Dom(\theta) = \emptyset \\ \prod_{i=1}^{n} p_{x_i}^{\theta(x_i)+1}, & \text{ако } Dom(\theta) = \{x_1, \ldots, x_n\}. \end{cases}$$

Докажете, че κ ефективно кодиране на крайните функции.

Доказателство. Да видим най-напред, че всяко $z > 0$ е код на единствена крайна функция. Наистина, ако $z = 1$ то z е може да е код само на $\emptyset^{(1)}$.

Ако $z > 1$, то z се разлага по единствен начин във вида

$$p_{x_1}^{t_1} \cdot \ldots \cdot p_{x_n}^{t_n},$$

където показателите $t_1, \ldots, t_n$ са положителни. Нека θ е крайната функция с дефиниционно множество $\{x_1, \ldots, x_n\}$, такава че $\theta(x_i) = t_i - 1$ за всяко $i = 1, \ldots, n$. Тогава очевидно $\kappa(\theta) = z$.

Сега трябва да покажем, че са примитивно рекурсивни функциите lh и mem, където

$lh(z) =$ броят на точките, в които крайната функция с код z е дефинирана,

$$mem(z,i) = \begin{cases} \Pi(x_i, \theta(x_i)), & \text{ако } z > 0 \ \& \ \kappa(\theta) = z \ \& \ x_i \text{ е } i\text{-тият} \\ & \text{по големина елемент на } Dom(\theta) \\ 0, & \text{ако } z = 0. \end{cases}$$

Лесно се вижда, че

$$lh(z) \ = \ \sum_{i=0}^{z} sg((z)_i).$$

За $mem(z,i)$ може да разсъждаваме така: нека

$$h(z) = \mu x_{x \leq z}[(z)_x > 0].$$

Ясно е, че при $z \geq 2$, $h(z)$ връща първия елемент от домейна на крайната функция с код z (по-горе видяхме, че всяко такова z е код на непразна крайна функция). Тогава за mem можем да запишем:

$$mem(z,i) = \begin{cases} \Pi(h(z), (z)_{h(z)} - 1), & \text{ако } z \geq 2 \ \& \ i = 1 \\ mem\left(\frac{z}{p_{h(z)}^{(z)_{h(z)}}}, i-1\right), & \text{ако } z \geq 2 \ \& \ i > 1 \\ 0, & \text{в останалите случаи.} \end{cases}$$

Като разсъждавате както в предишната задача, покажете, че представящата на mem

$$\hat{mem}(t) \ = \ mem(L(t), R(t))$$

е примитивно рекурсивна, откъдето и самата mem ще е такава. □

Задача 1.19. (Задача за ЕК) Да означим с Fin множеството на всички *крайни* подмножества на $\mathbb{N}$. Дефинираме изображение

$$\kappa\colon \ Fin \ \longrightarrow \ \mathbb{N}$$

по следния начин: ако $A = \{x_1, \ldots, x_n\}$, то $\kappa(A)$ е числото, в чийто двоичен запис единиците са точно на позиции $x_1, \ldots, x_n$ (като броим позициите от дясно наляво, започвайки от позиция 0).

Забележка. Това кодиране е познато като *канонично кодиране* на крайните множества от естествени числа.

Примери: $\kappa(\emptyset) = 0_{(2)}, \quad \kappa(\{0,1\}) = 11_{(2)}, \quad \kappa(\{1,3,4\}) = 11010_{(2)}.$

1) Докажете, че κ е биекция.

2) Нека $\kappa(A) = z$. Докажете, че са примитивно рекурсивни функциите lh и mem, дефинирани като:

$$lh(z) = |A|$$

$$mem(z,i) = \begin{cases} i\text{-тия по големина елемент на } A, & \text{ако } 1 \leq i \leq lh(z) \\ 0, & \text{иначе.} \end{cases}$$

Задача 1.20. (Функцията на Акерман) Докажете, че съществува единствена функция F, определена с равенствата:

$$\left|\begin{array}{rcl} F(0,y) & = & y+1 \\ F(x+1,0) & = & F(x,1) \\ F(x+1,y+1) & = & F(x,F(x+1,y)) \end{array}\right. \tag{1.13}$$

и тази функция е тотална.

Решение. Нека f удовлетворява (1.13). С индукция по x ще покажем, че

$$\forall x \underbrace{\forall y\ F(x,y) \text{ е еднозначно определена}}_{P(x)}.$$

База $x=0$. Следва от това, че

$$F(0,y) = y+1.$$

Да приемем, че за някое $x, P(x)$ е вярно, т.е.

$$\forall y\ F(x,y) \text{ е еднозначно определена.}$$

Трябва да покажем, че и $P(x+1)$ е вярно, т.е. вярно е, че

$$\forall y\ F(x+1,y) \text{ е еднозначно определена).}$$

Да означим

$$Q(y) \overset{\text{деф}}{\iff} F(x+1,y) \text{ е еднозначно определена.}$$

Сега с индукция относно y ще покажем, че $\forall y Q(y)$, което е точно $P(x+1)$. Наистина, при $y=0$ ще имаме

$$F(x+1,0) = F(x,1) \overset{\text{и.х. } P(x)}{=} \text{еднозначно определена.}$$

Допускайки, че за някое y е вярно $Q(y)$, за $y+1$ получаваме последователно

$$F(x+1,y+1) = F(x,F(x+1,y)) \overset{\text{и.х. } Q(y)}{=} F(x,z)) \overset{\text{и.х. } P(x)}{=} \text{е еднозначно определена.}$$

□

Да направим съвсем незначителна промяна в базисното условие в дефиницията на функцията на Акерман, като вместо $y+1$ пишем y. Получаваме функция, която е почти константа! Да видим:

Задача 1.21. Нека за функцията g е изпълнено:

$$\left|\begin{array}{rcl} g(0,y) & = & y \\ g(x+1,0) & = & g(x,1) \\ g(x+1,y+1) & = & g(x,g(x+1,y)) \end{array}\right. \tag{1.14}$$

Докажете, че g има следния явен вид:

$$g(x,y) = \begin{cases} y, & \text{ако } x = 0 \\ 1, & \text{иначе.} \end{cases}$$

Решение. Ясно е, че $g(0,y) = y$. Трябва да покажем, че

$$\forall x_{x\geq 1} \forall y \; g(x,y) = 1.$$

За целта с индукция по $x \geq 1$ да се убедим, че $\forall x P(x)$, където

$$P(x) \overset{\text{деф}}{\iff} \forall y \; g(x,y) = 1.$$

База $x = 1$, т.е. доказваме, че

$$\forall y \; \underbrace{g(1,y) = 1}_{Q(y)}.$$

Ще докажем, че $\forall y \; Q(y)$ с индукция относно y. При $y = 0$ от (1.14) получаваме:

$$g(1,0) \; = \; g(0,1) \; = \; 1.$$

Да допуснем, че за някое y е вярно $Q(y)$. Тогава за $Q(y+1)$ ще имаме, съгласно (1.14):

$$g(1,y+1) \; = \; g(0,g(1,y)) \overset{\text{и.х. } Q(y)}{=} \; g(0,1) \; = \; 1.$$

С това приключва проверката на $\forall y \; Q(y)$, или все едно — на твърдението $P(1)$. Сега да допуснем, че за някое $x \geq 1$ е изпълнено $P(x)$. Трябва да покажем, че и $P(x+1)$ е вярно, т.е.

$$\forall y \; \underbrace{g(x+1,y) = 1}_{R(y)}.$$

Трябва да покажем, че $\forall y \; R(y)$. Действаме отново с индукция относно y.

При $y = 0$ ще имаме

$$g(x+1,0) \; = \; g(x,1) \overset{\text{и.х. } P(x)}{=} \; 1.$$

Сега да приемем, че $R(y)$ е вярно за някое y. Тогава за $y+1$ ще имаме, съгласно (1.14):

$$g(x+1,y+1) \;=\; g(x,g(x+1,y)) \overset{\text{и.х. } R(y)}{=} g(x,1) \overset{\text{и.х. } P(x)}{=} 1.$$

□

Нека f_k е едноместната функция, която се получава от функцията на Акерман при фиксиран първи аргумент, равен на k, т.е.

$$f_k(y) \;=\; F(k,y)$$

за всяко $y \in \mathbb{N}$. Следват няколко задачи за някои основни свойства на тези функции.

Задача 1.22. Докажете, че при всяко $k \geq 0$ функцията f_{k+1} се получава от f_k по следната примитивно рекурсивна схема:

$$\left|\begin{array}{l} f_{k+1}(0) = f_k(1) \\ f_{k+1}(y+1) = f_k(f_{k+1}(y)). \end{array}\right. \tag{1.15}$$

Докажете още, че всяка от функциите f_k е примитивно рекурсивна.

Доказателство. Това, че f_k удовлетворява горната примитивно рекурсивна схема следва непосредствено от дефиницията (1.13) на функцията на Акерман:

$$f_{k+1}(0) \overset{\text{деф}}{=} F(k+1,0) \overset{(1.13)}{=} F(k,1) \;=\; f_k(1) \quad \text{и}$$
$$f_{k+1}(y+1) \;=\; F(k+1,y+1) \overset{(1.13)}{=} F(k,F(k+1,y)) \;=\; f_k(f_{k+1}(y)).$$

Сега с тривиална индукция по k получаваме, че f_k е примитивно рекурсивна. □

Задача 1.23. Докажете, че за всички естествени k и y са изпълнени равенствата:

1) $f_{k+1}(y) = f_k^{y+1}(1)$;

2) $f_{k+1}(1) = f_k(\ldots f_1(f_0(2)))\ldots)$.

Упътване. 1) Отново разсъждаваме с индукция, този път по y.

База $y = 0$:

$$f_{k+1}(0) = f_k(1) = f_k^{y+1}(1).$$

Сега ако допуснем, че 1) е вярно за някое y, за $y+1$, от равенствата (1.15) ще имаме:

$$f_{k+1}(y+1) \overset{(1.15)}{=} f_k(f_{k+1}(y)) \overset{\text{и.х.}}{=} f_k(f_k^{y+1}(1)) = f_k^{y+2}(1).$$

□

Задача 1.24. Намерете явния вид на функциите f_k за $k = 0, 1, 2, 3, 4$.

Доказателство. По определение за всяко y:

$$f_0(y) = F(0, y) = y + 1,$$

т.е. първата функция е $f_0(y) = y + 1$.
За следващите функции ще използваме, че f_k удовлетворяват схемата (1.15) от *Задача* 1.22:

$$\left| \begin{array}{l} f_{k+1}(0) = f_k(1) \\ f_{k+1}(y+1) = f_k(f_{k+1}(y)). \end{array} \right.$$

Така за f_1 ще имаме:

$$\left| \begin{array}{lll} f_1(0) & = & f_0(1) \ = \ 2 \\ f_1(y+1) & = & f_0(f_1(y)) \ = \ f_1(y) + 1, \end{array} \right.$$

и оттук лесно стигаме до извода, че $f_1(y) = y + 2$.

За f_2 получаваме:

$$\left| \begin{array}{lll} f_2(0) & = & f_1(1) \ = \ 3 \\ f_2(y+1) & = & f_1(f_2(y)) \ = \ f_2(y) + 2. \end{array} \right.$$

Ясно е, че трябва да търсим f_2 във вида $f_2(y) = ay + b$. Това равенство е в сила за всяко y. Заместваме с възможно най-малките стойности $y = 0$ и $y = 1$ и получаваме следната система за a и b:

$$\left| \begin{array}{lll} f_1(0) & = & f_2(0) \ = \ a.0 + b \ = \ 3 \\ f_1(y+1) & = & f_2(1) \ = \ a.1 + b \ = \ a + 3 \ = \ 5. \end{array} \right.$$

Оттук $a = 2, b = 3$, и значи $f_2(y) = 2y + 3$..

Условията за f_3 са:

$$\left| \begin{array}{lll} f_3(0) & = & f_2(1) \ = \ 5 \\ f_3(y+1) & = & f_2(f_3(y)) \ = \ 2.f_3(y) + 3. \end{array} \right.$$

Тук вече търсим f_3 във вида $f_3(y) = 2^{ay+b} + c$. Както по-горе, заместваме в горните равенства с различни стойности на y (достатъчно е да вземем например $y = 0, 1$ и 2), за да получим система за a, b и c. От нея получаваме, че $f_3(y) = 2^{y+3} - 3$.

Да се опитаме да пресметнем и f_4. За начало имаме:

$$\left| \begin{array}{lll} f_4(0) & = & f_3(1) \ = \ 13 \\ f_4(y+1) & = & f_3(f_4(y)) \ = \ 2^{f_4(y)+3} - 3. \end{array} \right.$$

Виждаме, че $f_4(y+1) \approx 2^{f_4(y)}$ и значи f_4 трябва да е от вида $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{y \text{ пъти}}$.

По-точният отговор е $f_4(y) = \underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{y+3 \text{ пъти}} - 3$. Довършете подробностите.

□

Задача 1.25. (Задача за ЕК) Докажете, че функцията на Акерман не е примитивно рекурсивна.

Упътване. Опитайте се да покажете, че за всяка примитивно рекурсивна функция g съществува k, такова че

$$g(x_1, \ldots, x_n) < F(k, max(x_1, \ldots, x_n))$$

за всички естествени $x_1, \ldots, x_n$.

За целта преди това покажете следните свойства на F:

(1) $F(x+1, y) > y+1$ за всяко x и y;

(2) F е монотонно растяща по двата си аргумента;

(3) $F(x+1, y) \geq F(x, y+1)$ за всяко x и y.

□