Лекция 10: Програмна характеризация на примитивно рекурсивните функции

3.9 Програмна характеризация на примитивно рекурсивните функции

Ще завършим тази глава с описанието на един учебен език, на който се програмират всички едноместни примитивно рекурсивни функции (и само те). Ще построим транслатор от този език към езика на МНР, както и интерпретатор, който ще се явява универсална функция за едноместните примитивно рекурсивни функции. Основната ни цел ще бъде да наблюдаваме къде влизат в действие най-важните теореми, които доказахме дотук.

3.9.1 Езикът SL

Езикът SL, който ще въведем, е предложен от проф. Димитър Скордев. Той е съвсем прост функционален език. Програмите на този език ще наричаме *SL-изрази* (или *SL-програми*).

Понятието *SL-израз* определяме със следната индуктивна дефиниция:

Определение 3.7.

1) SL-изрази са $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{O}}$, $\mathbf{I}$, $\mathbf{L}$ и $\mathbf{R}$ (ще ги наричаме *базисни*).

2) Ако E_1 и E_2 са SL-изрази, то и $(E_1 \circ E_2)$ е SL-израз.

3) Ако E_1 и E_2 са SL-изрази, то и $\mathbf{\Pi}(E_1, E_2)$ е SL-израз.

4) Ако E е SL-израз, то и $\boldsymbol{\rho}(E)$ е SL-израз.

SL-изразите можем да зададем по-кратко чрез следната граматика:

$$E \ ::= \ \boldsymbol{\mathcal{S}} \mid \boldsymbol{\mathcal{O}} \mid \mathbf{I} \mid \mathbf{L} \mid \mathbf{R} \mid (E \circ E) \mid \mathbf{\Pi}(E,E) \mid \boldsymbol{\rho}(E)$$

Примери за SL-изрази: $\mathbf{S}$, $(\mathbf{S} \circ \boldsymbol{\mathcal{O}})$, $\mathbf{\Pi}(\boldsymbol{\rho}((\mathbf{S} \circ \boldsymbol{\mathcal{O}})), \mathbf{L})$.

SL-изразите са синтактични обекти. Но какво се крие зад всеки SL-израз? За да определим смисъла (семантиката) на всеки такъв израз, най-напред ще дефинираме две нови операции над едноместни функции. Първата се нарича *свечетаване* на две функции; ще я означаваме с $\mathsf{\Pi}$. За всяка $f, g \in \mathcal{F}_1$ полагаме $\mathsf{\Pi}(f,g)$ да е функцията, дефинирана като:

$$\mathsf{\Pi}(f,g)(x) \stackrel{\text{деф}}{=} \Pi(f(x), g(x)),$$

където $\Pi(x,y) = 2^x(2y+1) - 1$ е кодирането на наредени двойки от числа, което въведохме в раздел 1.8.1. Да напомним, че декодиращите функции на това кодиране означавахме с L и R.

Забележка. Правете разлика между записите $\mathsf{\Pi}(f,g)$ и $\Pi(f,g)$. Първият е функционалната операция $\mathsf{\Pi}$ (съчетаване), приложена върху аргументите f и g, докато вторият е суперпозицията на функциите Π, f и g.

Да отбележим, че операцията $\mathsf{\Pi}$ действа като *кодиране на функции*. Ако f и g са *тотални*, функцията $h = \mathsf{\Pi}(f,g)$ "помни" f и g, т.е. ако знаем h, можем да възстановим f и g. Това е така, защото очевидно

$$f(x) = L(\Pi(f(x), g(x))) \quad \text{и} \quad g(x) = R(\Pi(f(x), g(x))),$$

откъдето веднага $f = L \circ h$ и $g = R \circ h$.

Втората функционална операция, която ще наричаме *ротация* и ще означаваме с ρ, всъщност не е съвсем нова за нас. Тя е на практика операцията итерация, която въведохме в раздел 1.7.4, но дефинирана така, че резултатът да бъде *едноместна* функция. Ето точното определение:

За всяка $f \in \mathcal{F}_1$ дефинираме нейната ротация $\rho(f)$ както следва:

$$\rho(f)(x) \overset{\text{деф}}{\simeq} f^{L(x)}(R(x)).$$

Когато прилагаме $\rho(f)$ към аргументи от вида $\Pi(n,x)$, получаваме точно итерацията f^* на f, защото $\rho(f)(\Pi(n,x)) \simeq f^n(x) \simeq f^*(n,x)$.

Сега вече можем да дефинираме семантиката на един SL-израз, т.е. функцията, която той определя. Разбира се, дефиницията ще е по индукция, следваща индуктивната дефиниция 3.7 на SL-израз:

1) Базисните изрази $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{O}}$, $\mathbf{I}$, $\mathbf{L}$ и $\mathbf{R}$ определят съответно функциите $\mathcal{S}, \mathcal{O}, I, L$ и R, където $\mathcal{S}$ е $\lambda x.x+1$, $\mathcal{O}$ е $\lambda x.0$, I е $\lambda x.x$ (идентитетът), а L и R са декодиращите функции на кодирането Π.

2) Ако SL-изразите E_1 и E_2 определят функциите f и g, то SL-изразът $(E_1 \circ E_2)$ определя тяхната композиция $f \circ g$.

3) Ако SL-изразите E_1 и E_2 определят функциите f и g, то SL-изразът $\boldsymbol{\Pi}(E_1, E_2)$ определя тяхното съчетаване $\mathsf{\Pi}(f,g)$.

4) Ако SL-изразът E определя функцията f, то SL-изразът $\boldsymbol{\rho}(E)$ определя ротацията $\rho(f)$ на f.

Скобите в изразите от вида $(E_1 \circ E_2)$ са заради еднозначния синтактичен анализ — за да е ясно как се чете, примерно, изразът $E_1 \circ E_2 \circ E_3$ — дали той е $((E_1 \circ E_2) \circ E_3)$ или $(E_1 \circ (E_2 \circ E_3))$. Но тъй като композицията е асоциативна операция, за семантиката това не е от значение. Затова по-надолу обикновено ще пропускаме тези скоби.

Правете разлика между *израз* и *функция*, която този израз определя. Например изразите $\mathbf{S}$ и $\mathbf{S} \circ \mathbf{I}$ са различни, но имат една и съща семантика, т.е. определят една и съща функция, в случая $\lambda x.x+1$. За такива изрази ще казваме, че са *еквивалентни*.

Определение 3.8. Функцията $f \in \mathcal{F}_1$ наричаме *SL-определима*, ако тя се определя от някой SL-израз.

Съвкупността от тези функции ще означаваме с $\mathcal{SL}$, т.е.

$$\mathcal{SL} = \{f \mid f \text{ е } SL\text{-определима}\}.$$

Другояче казано, функциите от класа $\mathcal{SL}$ са точно тези, които се получават от базисните функции $\mathcal{S}, \mathcal{O}, I, L$ и R чрез краен брой прилагания на функционалните операции $\circ$, $\mathsf{\Pi}$ и ρ.

Ясно е, че ако f е SL-определима, има безброй много изрази, които я определят. Наистина, ако f се определя от израза E, то и всеки от изразите $E \circ \underbrace{\mathbf{I} \circ \cdots \circ \mathbf{I}}_{n \text{ пъти}}$ определя f (като, разбира се, това далеч не са всички изрази, еквивалентни на E.).

Нашата близка цел ще бъде е да покажем, че функциите, определими в езика SL, са точно едноместните примитивно рекурсивни функции.

Теорема 3.10. (Скордев) Едноместната функция f е SL-определима тогава и само тогава, когато тя е примитивно рекурсивна.

Доказателство. Нека f се определя от SL-израза E. С индукция по дефиницията на E ще покажем, че f е примитивно рекурсивна.

Наистина, ако E е базисен SL-израз, то той определя някоя от функциите $\mathcal{S}, \mathcal{O}, I, L$ и R, които са примитивно рекурсивни. Нека E е от вида $E_1 \circ E_2$, като за E_1 и E_2 е вярно, че функциите f и g, които те определят, са примитивно рекурсивни. Но $E_1 \circ E_2$ определя $f \circ g$, която очевидно също е примитивно рекурсивна. Ако $E = \mathbf{\Pi}(E_1, E_2)$ и E_1 и E_2 определят примитивно рекурсивните f и g, то $\mathbf{\Pi}(E_1, E_2)$ определя функцията $\mathsf{\Pi}(f, g) \overset{\text{деф}}{=} \Pi(f, g)$, която също ще е примитивно рекурсивна. Накрая, ако $E = \boldsymbol{\rho}(E_1)$, като по индуктивната хипотеза E_1 определя примитивно рекурсивната функция f, то $\boldsymbol{\rho}(E_1)$ определя функцията $\lambda x.f^*(L(x), R(x))$, която по *Твърдение* 1.17 е примитивно рекурсивна.

За обратната посока на твърдението не можем да разсъждаваме с индукция по дефиницията на класа на примитивно рекурсивните функции. Проблемът е, че с тази дефиниция се въвеждат *всички* примитивно рекурсивни функции, а на нас ни трябват само *едноместните*. Този технически проблем ще преодолеем, като се възползваме от понятието *представяща* на дадена функция f.

Да напомним, че ако f е произволна функция на n аргумента, *представяща* на f е едноместната функция $\hat{f} \in \mathcal{F}_1$, която действа върху *кодовете* на аргументите на f така, както f действа върху самите аргументи, с други думи

$$\hat{f}(\Pi_n(x_1, \ldots, x_n)) \simeq f(x_1, \ldots, x_n).$$

"Официалната" дефиниция на $\hat{f}$ е следната: за всяко $z \in \mathbb{N}$:

$$\hat{f}(z) \overset{\text{деф}}{\simeq} f(J_1^n(z), \ldots, J_n^n(z)).$$

Тук $J_1^n, \ldots, J_n^n$ са декодиращите функции за кодирането Π_n от раздел 1.8.2. При $n = 1$ очевидно $\hat{f} = f$.

Нашата цел ще е да покажем, че ако f е примитивно рекурсивна, то нейната представяща $\hat{f}$ е SL-определима. В частност, когато f е едноместна, от $\hat{f} = f$ ще получим, че f е SL-определима.

Оттук до края на доказателството на теоремата целта ни ще бъде да покажем на импликацията

$$f \text{ е примитивно рекурсивна} \implies \hat{f} \in \mathcal{SL}. \tag{3.16}$$

Това ще направим с индукция по дефиницията на класа на примитивно рекурсивните функции. Ако f е $\mathcal{S}$ или $\mathcal{O}$, няма какво да се доказва. Ако f е проектиращата функция I_k^n, то нейната представяща е функцията J_k^n. От *Твърдение* 1.21 знаем, че за J_k^n имаме следното представяне:

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

То показва, че и в двата случая $k = 1$ и $1 < k \leq n$ — ще имаме, че J_k^n ще принадлежи на $\mathcal{SL}$.

Сега нека за $f \in \mathcal{F}_n$ е изпълнено $f = g(h_1, \ldots, h_k)$, като по индукционната хипотеза $\hat{g}, \hat{h}_1, \ldots, \hat{h}_k$ лежат в $\mathcal{SL}$. Трябва да покажем, че и $\hat{f} \in \mathcal{SL}$. Да разпишем дефиницията на $\hat{f}$:

$$\begin{aligned} \hat{f}(\Pi_n(\bar{x})) \overset{\text{деф}}{=} f(\bar{x}) \;&=\; g(h_1(\bar{x}), \ldots, h_k(\bar{x})) \;=\; \hat{g}(\Pi_k(h_1(\bar{x}), \ldots, h_k(\bar{x}))) \\ &= \hat{g}(\Pi_k(\hat{h}_1(\Pi_n(\bar{x})), \ldots, \hat{h}_k(\Pi_n(\bar{x})))) \;=\; \hat{g}(\Pi_k(\hat{h}_1, \ldots, \hat{h}_k))(\Pi_n(\bar{x})). \end{aligned}$$

Следователно $\hat{f} = \hat{g}(\Pi_k(\hat{h}_1, \ldots, \hat{h}_k))$. Съгласно индуктивната хипотеза $\hat{g}, \hat{h}_1, \ldots, \hat{h}_k$ са от $\mathcal{SL}$. Сега за да се убедим, че и $\hat{f} \in \mathcal{SL}$, е достатъчно да си спомним дефиницията на кодирането Π_k от раздел 1.8.2. Имаме

$$\Pi_k(\hat{h}_1, \ldots, \hat{h}_k) \overset{\text{деф}}{=} \underbrace{\Pi(\ldots\Pi}_{k-1 \text{ пъти}}(\hat{h}_1, \hat{h}_2), \ldots, \hat{h}_k) = \underbrace{\mathsf{\Pi}(\ldots\mathsf{\Pi}}_{k-1 \text{ пъти}}(\hat{h}_1, \hat{h}_2), \ldots, \hat{h}_k).$$

От това представяне се вижда, че функцията $\Pi_k(\hat{h}_1, \ldots, \hat{h}_k)$ се получава след $k-1$-кратно прилагане на операцията $\mathsf{\Pi}$ към функции, които са в класа $\mathcal{SL}$, следователно и тя е в $\mathcal{SL}$, откъдето веднага и $\hat{f} \in \mathcal{SL}$.

Нека сега f се получава с примитивна рекурсия от функциите g и h, като отново по индуктивната хипотеза $\hat{g}$ и $\hat{h}$ са в класа $\mathcal{SL}$. Трябва да

покажем, че и $\hat{f} \in \mathcal{SL}$. Да разгледаме най-напред случая, когато f е на 2 аргумента. Тогава тя удовлетворява примитивно рекурсивната схема

$$\begin{aligned} f(x,0) &= g(x) \\ f(x,y+1) &= h(x,y,f(x,y)). \end{aligned} \tag{3.17}$$

Ще покажем, че примитивната рекурсия може да се изрази чрез операцията ρ. За целта да разгледаме изображението

$$(x,y,z) \;\mapsto\; (x,\; y+1,\; h(x,y,z)).$$

При $y=0$ и $z=g(x)$ ще имаме

$$(x,\,0,\,g(x)) \;\mapsto\; (x,\,1,\,h(x,0,\underbrace{g(x)}_{f(x,0)})), \quad \text{т.е.} \quad (x,0,g(x)) \;\mapsto\; (x,1,f(x,1)).$$

Аналогично

$$(x,1,f(x,1)) \;\mapsto\; (x,2,\underbrace{h(x,1,f(x,1))}_{f(x,2)}), \text{ т.е. } (x,1,f(x,1)) \;\mapsto\; (x,2,f(x,2)).$$

Тези експерименти ни дават основание да предположим, че тръгвайки от $(x,0,g(x))$, след y итерации ще стигнем до наредената тройка $(x,\;y,\;f(x,y))$, от която ще можем да извлечем $f(x,y)$. За да направим нещата строги, ще трябва да разглеждаме горното преобразование не върху тройки от естествени числа, а върху техните *кодове*.

Да дефинираме функцията H по следния начин:

$$H(\Pi_3(x,y,z)) \overset{\text{деф}}{=} \Pi_3(x,y+1,h(x,y,z)). \tag{3.18}$$

С индукция по y ще покажем, че

$$H^y(\Pi_3(x,0,g(x))) \;=\; \Pi_3(x,y,f(x,y)). \tag{3.19}$$

Наистина, при $y=0$ имаме, че

$$H^0(\Pi_3(x,0,g(x))) \overset{\text{деф}}{=} \Pi_3(x,0,g(x)) \overset{(3.17)}{=} \Pi_3(x,0,f(x,0)).$$

Ако приемем, че за някое y е вярно (3.19), то за $y+1$ получаваме:

$$\begin{aligned} H^{y+1}(\Pi_3(x,0,g(x))) \;&=\; H(H^y(\Pi_3(x,0,g(x)))) \overset{\text{и.х.}}{=} H(\Pi_3(x,y,f(x,y))) \\ &\overset{\text{деф}}{=} \Pi_3(x,y{+}1,h(x,y,f(x,y))) \overset{(3.17)}{=} \Pi_3(x,y{+}1,f(x,y{+}1)). \end{aligned}$$

Така проверихме верността на равенството (3.19). От него получаваме следното изразяване за f:

$$f(x,y) \;=\; J_3^3(H^y(\Pi_3(x,0,g(x)))).$$

Виждаме, че примитивната рекурсия вече е заместена с итерация. Остана тази итерация да изразим чрез операцията ρ и освен това да видим, че функцията H е в класа $\mathcal{SL}$. Наистина, гледайки дефиницията (3.18), за H можем да запишем:

$$H(z) = \Pi_3(J^3_1(z), J^3_2(z)+1, h(J^3_1(z), J^3_2(z), J^3_3(z))) = \Pi_3(J^3_1(z), J^3_2(z)+1, \hat{h}(z))$$
$$= \Pi(\Pi(J^3_1(z), \mathcal{S}(J^3_2(z))), \hat{h}(z)) = \mathsf{\Pi}(\mathsf{\Pi}(J^3_1, \mathcal{S} \circ J^3_2), \hat{h})(z).$$

Следователно $H = \mathsf{\Pi}(\mathsf{\Pi}(J^3_1, \mathcal{S} \circ J^3_2), \hat{h}) \in \mathcal{SL}$, тъй като по индукционната хипотеза $\hat{h} \in \mathcal{SL}$, а и всяка от функциите J^3_1, J^3_2 и $\mathcal{S}$ също са в този клас. Финално, за представящата $\hat{f}$ на f ще имаме:

$$\begin{aligned}\hat{f}(z) &= f(L(z), R(z)) = J^3_3(H^{R(z)}(\Pi_3(L(z), 0, g(L(z)))))\\ &= J^3_3(\rho(H)(\Pi(R(z), \Pi_3(L(z), \mathcal{O}(z), g(L(z))))))\\ &= J^3_3(\rho(H)(\mathsf{\Pi}(R, \mathsf{\Pi}(\mathsf{\Pi}(L, \mathcal{O}), g \circ L)))))(z).\end{aligned}$$

Сега вече уверено можем да твърдим, че $\hat{f} \in \mathcal{SL}$.

По подобен начин разсъждаваме и в случая, когато f е на $n = 1$ или на $n \geq 3$ променливи, което оставяме като добро упражнение за читателя ☺. □

3.9.2 Кодиране на SL-изразите

Имаме 4 типа SL-изрази, затова решаваме да ги кодираме според остатъците им при деление на 4, по-точно — изразите от първия вид да имат остатък 0 при деление на 4, от втория вид — остатък 1 и т.н.

С индукция по дефиницията 3.7 на SL-израз полагаме:

1) $\alpha(\boldsymbol{\mathcal{S}}) = 0,\ \alpha(\boldsymbol{\mathcal{O}}) = 4,\ \alpha(\mathbf{I}) = 8,\ \alpha(\mathbf{L}) = 12,\ \alpha(\mathbf{R}) = 4k,\ k \geq 4;$

2) $\alpha(E_1 \circ E_2) = 4\Pi(\alpha(E_1), \alpha(E_2)) + 1;$

3) $\alpha(\mathbf{\Pi}(E_1, E_2)) = 4\Pi(\alpha(E_1), \alpha(E_2)) + 2;$

4) $\alpha(\boldsymbol{\rho}(E)) = 4\alpha(E) + 3.$

Всяко от числата $\alpha(E)$ ще наричаме *код на SL-израза E*. От определението се вижда, че базисният SL-израз $\mathbf{R}$ има безброй много кодове, откъдето следва, че всички изрази, в които той участва, също ще имат безброй много кодове. Това, че изображението α е многозначно, не създава проблем, защото нас ще ни интересува по-скоро обратното изображение α^{-1}, чрез което ще номерираме всички SL-изрази, полагайки $E_a = \alpha^{-1}(a)$ (съвсем по аналогия с $P_a = \gamma^{-1}(a)$).

За да можем да направим това, обаче, трябва да сме сигурни, че на всяко естествено число a съответства точно един SL-израз. Да се убедим:

Твърдение 3.13. Всяко естествено число a е код на точно един SL-израз.

Доказателство. Ще разсъждаваме с пълна индукция по a. Базата на индукцията са числата, кратни на 4. За тях е ясно, че кодират по точно един (базисен) SL-израз.

Нека $a = 4k + 1$. В края на раздел 1.8.1 съобразихме, че за всяко естествено k,$L(k) \leq k$ и $R(k) \leq k$. Тогава със сигурност $L(k) < a$ и $R(k) < a$ и по индуктивната хипотеза ще съществуват единствени SL-изрази E_1 и E_2, такива че $\alpha(E_1) = L(k)$ и $\alpha(E_2) = R(k)$. В такъв случай

$$\alpha(E_1 \circ E_2) \;=\; 4\Pi(\alpha(E_1), \alpha(E_2)) + 1 \;=\; 4\Pi(L(k), R(k)) + 1 \;=\; 4k + 1 = a$$

и очевидно $E_1 \circ E_2$ е единственият SL-израз с това свойство. По същия начин разсъждаваме и когато $a = 4k + 2$. Ако $a = 4k + 3$, имаме $k < a$ и по индуктивната хипотеза ще има единствен SL-израз E, чийто код е k. Така за кода на израза $\boldsymbol{\rho}(E)$ получаваме:

$$\alpha(\boldsymbol{\rho}(E)) \;=\; 4\alpha(E) + 3 \;=\; 4k + 3 = a.$$

□

Сега вече, като сме сигурни, че на всяко $a \in \mathbb{N}$ съответства точно един SL-израз, можем да положим

$$E_a \stackrel{\text{деф}}{=} SL\text{-израза с код } a.$$

Нека още

$$f_a \stackrel{\text{деф}}{=} \text{функцията, която се определя от } SL\text{-израза } E_a.$$

Тогава очевидно редицата

$$E_0, E_1, \ldots, E_a, \ldots$$

изброява всички SL-изрази, а редицата

$$f_0, f_1, \ldots, f_a, \ldots$$

изброява всички SL-определими функции, или все едно — всички едноместни примитивно рекурсивни функции.

3.9.3 Построяване на компилатор за езика SL

Искаме да конструираме *изчислимо* изображение $\kappa : SL \longrightarrow$ МНР, което транслира всеки SL-израз в еквивалентна на него програма за МНР. За да твърдим строго, че κ е изчислима, тя трябва да е *числова* функция, с други думи — да преработва *кодовете* на SL-изразите в съответните *кодове* на програми за МНР.

Искаме SL-изразът E_a и преводът му $P_{\kappa(a)}$ да бъдат еквивалентни, което ще рече — да пресмятат една и съща функция. Формално това означава да е изпълнено равенството

$$f_a = \varphi_{\kappa(a)}. \tag{3.20}$$

Нашата идея е да дефинираме κ така, че тя да удовлетворява горното условие върху кодовете на *базисните* SL-изрази, а после да разширим дефиницията ѝ върху останалите кодове, като се грижим, разбира се, да поддържаме условието (3.20).

Върху базисните SL-изрази можем да дефинираме κ по най-различни начини. Ако това е истински компилатор, желателно е да превеждаме във възможно най-простите програми. Например изразът **S** да се транслира в програмата с единствен оператор $X_1 := X_1 + 1$, изразът **O** — в програмата $X_1 := 0$ и прочее.

В случая не се интересуваме от ефективност, затова просто фиксираме някакви МНР програми $P_{a_0}, \ldots, P_{a_4}$, които пресмятат функциите $\mathcal{S}, \mathcal{O}, I, L$ и R, съответно. Да положим

$$\kappa(0) = a_0,\ \kappa(4) = a_1,\ \kappa(8) = a_2,\ \kappa(12) = a_3 \text{ и } \kappa(4k) = a_4 \text{ за всяко } k \geq 4.$$

Нека $a = 4k + 1$. Тогава $E_a = E_{L(k)} \circ E_{R(k)}$ и съответно $f_a = f_{L(k)} \circ f_{R(k)}$. Да приемем за момент (в доказателството на следващото *Твърдение* 3.14 това ще бъде оформено строго като индуктивна хипотеза), че за изразите $E_{L(k)}$ и $E_{R(k)}$, чийто кодове $L(k)$ и $R(k)$ са *по-малки* от k, е в сила условието (3.20). Искаме да дефинираме κ за $a = 4k + 1$ така, че условието (3.20) да продължава да е в сила.

При $a = 4k + 1$ знаем, че $\kappa(a)$ е код на SL-израз, който е композиция на два други. И тук решаваща роля има *ефективността* на операцията (или оператора) композиция, която проверихме в *Задача* 3.15. Нещо повече, там показахме, че съществува и *примитивно рекурсивна* индексна функция, която нарекохме *comp*. Тази функция беше такава, че за всички естествени a и b:

$$\varphi_a \circ \varphi_b = \varphi_{comp(a,b)}.$$

Както вече казахме, ще предполагаме, че върху $L(k)$ и $R(k)$ компилаторът κ се държи коректно, т.е. $f_{L(k)} = \varphi_{\kappa(L(k))}$ и $f_{R(k)} = \varphi_{\kappa(R(k))}$. Тогава за f_a можем да запишем:

$$f_a = f_{L(k)} \circ f_{R(k)} \overset{\text{и.х.}}{=} \varphi_{\kappa(L(k))} \circ \varphi_{\kappa(R(k))} = \varphi_{comp(\kappa(L(k)),\kappa(R(k)))}.$$

Получихме, че $f_a = \varphi_{comp(\kappa(L(k)),\kappa(R(k)))}$, което ни навежда на мисълта да дефинираме κ за $a = 4k+1$ така:

$$\kappa(4k+1) \overset{\text{деф}}{=} comp(\kappa(L(k)), \kappa(R(k))).$$

Тогава за $a = 4k+1$ условието (3.20) ще бъде осигурено автоматично:

$$\varphi_{\kappa(a)} = \varphi_{comp(\kappa(L(k)),\kappa(R(k)))} = f_a.$$

За да определим κ и в останалите случаи, ще ни е нужно да знаем, че операциите Π и ρ също са ефективни и значи имат примитивно рекурсивни индексни функции.

Задача 3.35. Докажете, че съществуват примитивно рекурсивни функции pi и ro, такива че за всяко a и b:
а) $\varphi_{pi(a,b)} = \Pi(\varphi_a, \varphi_b)$;
б) $\varphi_{ro(a)} = \rho(\varphi_a)$.

Решение. а) Разглеждаме функцията

$$F(a,b,x) \overset{\text{деф}}{\simeq} \Pi(\varphi_a, \varphi_b)(x) \simeq \Pi(\varphi_a(x), \varphi_b(x)).$$

Тя е изчислима, защото можем да я препишем като

$$F(a,b,x) \simeq \Pi(\Phi_1(a,x), \Phi_1(b,x)).$$

Тогава по S_n^m-теоремата ще съществува примитивно рекурсивна функция pi, такава че

$$\varphi_{pi(a,b)}(x) \simeq F(a,b,x) \simeq \Pi(\varphi_a, \varphi_b)(x)$$

за всяко a, b, x. Това означава, че

$$\varphi_{pi(a,b)} = \Pi(\varphi_a, \varphi_b)$$

за всяко a и b.

б) Разсъждаваме аналогично, като разглеждаме функцията

$$G(a,x) \overset{\text{деф}}{\simeq} \rho(\varphi_a)(x) \simeq \varphi_a^{L(x)}(R(x)) \simeq \underbrace{\varphi_a(\ldots\varphi_a}_{L(x)\ \text{пъти}}(R(x))\ldots).$$

За да покажем, че G е изчислима, отново минаваме през универсалната функция Φ_1. Ще имаме

$$G(a,x) \simeq \underbrace{\Phi_1(a,\dots\Phi_1}_{L(x)\text{ пъти}}(a,R(x))\dots) \simeq \Phi_1^*(L(x),a,R(x)).$$

Като си спомним за *Задача* 1.14, можем да твърдим, че G е изчислима. Прилагаме и към нея S_n^m-теоремата и получаваме, че за някоя примитивно рекурсивна функция — да я наречем ro — ще е изпълнено

$$\varphi_{ro(a)}(x) \simeq G(a,x) \simeq \rho(\varphi_a)(x)$$

за всяко a и x. Оттук за всяко a ще имаме

$$\varphi_{ro(a)} = \rho(\varphi_a).$$

□

Като имаме опита от случая $a = 4k+1$, ни е съвсем ясно как да дефинираме $\kappa(a)$ при $a = 4k+2$ и $a = 4k+3$:

$$\kappa(4k+2) \overset{\text{деф}}{=} pi(\kappa(L(k)),\kappa(R(k))) \quad \text{и} \quad \kappa(4k+3) \overset{\text{деф}}{=} ro(\kappa(k)).$$

Като използваме свойствата на pi и ro от *Задача* 3.35, както и предположението, че върху по-малки индекси κ удовлетворява условието (3.20), ще имаме

$$f_{4k+2} = \Pi(f_{L(k)}, f_{R(k)}) \overset{\text{и.х.}}{=} \Pi(\varphi_{\kappa(L(k))}, \varphi_{\kappa(R(k))}) = \varphi_{pi(\kappa(L(k)),\kappa(R(k)))} = \varphi_{\kappa(4k+2)}$$

и аналогично

$$f_{4k+3} = \rho(f_k) \overset{\text{и.х.}}{=} \rho(\varphi_{\kappa(k)}) = \varphi_{ro(\kappa(k))} = \varphi_{\kappa(4k+3)}.$$

Така получихме, че $\kappa(a)$ удовлетворява условието (3.20) и при $a = 4k+2$ и $a = 4k+3$.

От всичко, казано дотук, със сигурност можем да твърдим, че

Твърдение 3.14. Функцията κ е примитивно рекурсивна и удовлетворява условието

$$f_a = \varphi_{\kappa(a)}$$

за всяко естествено a.

Доказателство. Това, че κ удовлетворява условието (3.20), осигурихме докато я конструирахме. Пробвайте да го покажете "начисто", като разсъждавате с пълна индукция относно a.

За да се убедим, че κ е примитивно рекурсивна, трябва да съберем в едно частите от нейната дефиниция за всеки от случаите за аргумента ѝ a. Така получаваме следната рекурсивна дефиниция:

$$\kappa(a) = \begin{cases} a_0, & \text{ако } a = 0 \\ a_1, & \text{ако } a = 4 \\ a_2, & \text{ако } a = 8 \\ a_3, & \text{ако } a = 12 \\ a_4, & \text{ако } a \equiv 0 \ (mod\ 4) \ \& \ \lfloor \frac{a}{4} \rfloor \geq 4 \\ comp(\kappa(L(\lfloor \frac{a}{4} \rfloor)), \kappa(R(\lfloor \frac{a}{4} \rfloor))), & \text{ако } a \equiv 1 \ (mod\ 4) \\ pi(\kappa(L(\lfloor \frac{a}{4} \rfloor)), \kappa(R(\lfloor \frac{a}{4} \rfloor))), & \text{ако } a \equiv 2 \ (mod\ 4) \\ ro(\kappa(\lfloor \frac{a}{4} \rfloor)), & \text{ако } a \equiv 3 \ (mod\ 4). \end{cases}$$

От нея се вижда, че κ се дефинира с пълна рекурсия, като в дефиницията ѝ участват функции и предикати, които са примитивно рекурсивни. Следователно и κ е примитивно рекурсивна. □

3.9.4 Универсална функция за примитивно рекурсивните функции

Целта ни е да покажем, че за всяко $n \geq 1$, класът $\mathcal{PR}_n$ на n-местните примитивно рекурсивни функции има универсална функция. Това ще получим като директно следствие от факта, че примитивно рекурсивните функциите са точно функциите, които са програмируеми на езика SL, и имаме транслатор от този език към езика за МНР (а МНР-изчислимите функции вече имат универсална функция).

Теорема 3.11. За всяко $n \geq 1$ класът на n-местните примитивно рекурсивни функции има универсална функция.

Доказателство. Най-напред ще конструираме универсална функция за едноместните примитивно рекурсивни функции. Да положим

$$U(a, x) \stackrel{\text{деф}}{=} f_a(x),$$

където f_a е функцията, която се определя от SL-израза E_a. Тъй като всички SL-изрази пробягват редицата

$$E_0, E_1, \ldots, E_a, \ldots$$

то всички SL-определими функции (и само те) са в редицата

$$f_0, f_1, \ldots, f_a, \ldots$$

Теорема 3.10 ни дава, че това са точно едноместните примитивно рекурсивни функции. Тогава условията 1) и 2) от дефиницията за УФ са изпълнени автоматично за функцията $U(a,x)$. Остана да покажем, че тя е изчислима. Наистина, от равенството (3.20) имаме, че

$$U(a,x) \overset{\text{деф}}{=} f_a(x) = \varphi_{\kappa(a)}(x) = \Phi_1(\kappa(a),x),$$

където Φ_1 е универсалната за едноместните изчислими функции. Но Φ_1 е изчислима, $\kappa(a)$ — също (тя дори е примитивно рекурсивна), което означава, че и U е изчислима. Всъщност U е рекурсивна функция, защото е тотална.

Да фиксираме сега произволно $n \geq 1$ и да положим

$$U_n(a,x_1,\ldots,x_n) \overset{\text{деф}}{=} U(a,\Pi_n(x_1,\ldots,x_n)).$$

Да се убедим, че U_n е универсална за класа $\mathcal{PR}_n$. Тя очевидно е рекурсивна, тъй че трябва да проверим само условията 1) и 2) от дефиницията за УФ.

Да си спомним, че представящата представяща $\hat{f}$ на всяка функция $f \in \mathcal{PR}_n$ удовлетворява условието:

$$\hat{f}(z) \overset{\text{деф}}{\simeq} f(J_1^n(z),\ldots,J_n^n(z)).$$

Ясно е, че $\hat{f}$ също ще е от класа $\mathcal{PR}_1$. Но тогава съгласно *Теорема* 3.10 ще съществува a, за което $\hat{f} = f_a$. Следователно

$$U_n(a,\bar{x}) \overset{\text{деф}}{=} U(a,\Pi_n(\bar{x})) \overset{\text{деф } U}{=} f_a(\Pi_n(\bar{x})) = \hat{f}(\Pi_n(\bar{x})) \overset{\text{деф}}{=} f(\bar{x}),$$

с други думи, $\lambda\bar{x}.U_n(a,\bar{x}) = f$, т.е. условието 1) от дефиницията за УФ е в сила. Обратно, за всяко фиксирано a имаме, че функцията

$$\lambda\bar{x}.U_n(a,\bar{x}) = \lambda\bar{x}.U(a,\Pi_n(\bar{x})) = \lambda\bar{x}.f_a(\Pi_n(\bar{x})) = f_a \circ \Pi_n$$

очевидно е примитивно рекурсивна, т.е. налице е и второто условие от дефиницията за УФ. □

Забележка. Да обърнем внимание, че съгласно *Твърдение* 3.6, никоя от универсалните функции U_n не може да е примитивно рекурсивна. Всяка от тях е пример за функция, която е рекурсивна, но не е примитивно рекурсивна. Така вече имаме строго доказателство, че класът на примитивно рекурсивните функции се включва строго класа на рекурсивните.

3.9.5 Построяване на интерпретатор за езика SL

Сега се насочваме към конструиране на интерпретатор $\mathbb{I}(a, x)$ за езика SL. Идеята е по дадени a и x, $\mathbb{I}$ да връща стойността на функцията, определена от SL-израза E_a в точката x. По-горе тази функция означихме с f_a. Следователно дефиницията на $\mathbb{I}$ трябва да е такава:

$$\mathbb{I}(a, x) \ = \ f_a(x).$$

Върху кодовете на базисните изрази е ясно как трябва да дефинираме $\mathbb{I}$:

$$\mathbb{I}(0, x) \stackrel{\text{деф}}{=} \mathcal{S}(x) = x + 1; \quad \mathbb{I}(4, x) \stackrel{\text{деф}}{=} \mathcal{O}(x) = 0; \quad \mathbb{I}(8, x) \stackrel{\text{деф}}{=} I(x) = x;$$

$$\mathbb{I}(12, x) \stackrel{\text{деф}}{=} L(x); \quad \mathbb{I}(4k, x) \stackrel{\text{деф}}{=} R(x) \text{ за всяко } \ k \geq 4.$$

Идеята ни е, по подобие с конструкцията на κ в предишния раздел, да дефинираме рекурсивно $\mathbb{I}(a, x)$ за a такива, че E_a не са базисни. Тук, разбира се, отново ще предполагаме, че за $b < a$ вече сме осигурили $\mathbb{I}(b, x) = f_b(x)$ (и на това равенство по-долу ще се позоваваме като на индуктивна хипотеза).

Когато $a = 4k + 1$, имаме по определение $f_a = f_{L(k)} \circ f_{R(k)}$, и значи в този случай:

$$f_a(x) = f_{L(k)}(f_{R(k)}(x)) \stackrel{\text{и.х.}}{=} f_{L(k)}(\mathbb{I}(R(k), x)) \stackrel{\text{и.х.}}{=} \mathbb{I}(L(k), \mathbb{I}(R(k), x)).$$

Това означава, че за да осигурим $\mathbb{I}(a, x) = f_a(x)$, за $\mathbb{I}(a, x)$ трябва да положим

$$\mathbb{I}(a, x) = \mathbb{I}(L(k), \mathbb{I}(R(k), x)),$$

или разписано само чрез a:

$$\mathbb{I}(a, x) = \mathbb{I}(L(\lfloor \frac{a}{4} \rfloor), \mathbb{I}(R(\lfloor \frac{a}{4} \rfloor), x)).$$

Аналогично, за $a = 4k + 2$ по определение $f_a = \mathsf{\Pi}(f_{L(k)}, f_{R(k)})$ и следователно

$$f_a(x) = \mathsf{\Pi}(f_{L(k)}, f_{R(k)})(x) \stackrel{\text{и.х.}}{=} \Pi(\mathbb{I}(L(k), x), \mathbb{I}(R(k), x)).$$

Тогава за $\mathbb{I}(a, x)$ при $a = 4k + 2$ полагаме

$$\mathbb{I}(a, x) = \Pi(\mathbb{I}(L(\lfloor \frac{a}{4} \rfloor), x), \mathbb{I}(R(\lfloor \frac{a}{4} \rfloor), x)).$$

Остана случая $a = 4k+3$. В този случай $E_a = \boldsymbol{\rho}(E_k)$ и съответно $f_{4k+3} = \rho(f_k)$. Следователно

$$f_a(x) = \rho(f_k)(x) = f_k^{L(x)}(R(x)) = \underbrace{f_k(\dots f_k}_{L(x) \text{ пъти}}(R(x)\dots) \stackrel{\text{и.х.}}{=} \underbrace{\mathbb{I}(k, \dots, \mathbb{I}(k}_{L(x) \text{ пъти}}, R(x))\dots).$$

Тогава за $\mathbb{I}(a,x)$ в този случай полагаме

$$\mathbb{I}(a,x) = \underbrace{\mathbb{I}(\lfloor\frac{a}{4}\rfloor,\ldots,\mathbb{I}(\lfloor\frac{a}{4}\rfloor}_{L(x)\text{ пъти}},R(x))\ldots).$$

Така получаваме, че $\mathbb{I}$ трябва да удовлетворява следната рекурсивна схема:

$$\mathbb{I}(a,x) = \begin{cases} x+1, & \text{ако } a=0 \\ 0, & \text{ако } a=4 \\ x, & \text{ако } a=8 \\ L(x), & \text{ако } a=12 \\ R(x), & \text{ако } a\equiv 0\ (mod\ 4)\ \&\ \lfloor\frac{a}{4}\rfloor \geq 4 \\ \mathbb{I}(L(\lfloor\frac{a}{4}\rfloor),\mathbb{I}(R(\lfloor\frac{a}{4}\rfloor),x)), & \text{ако } a\equiv 1\ (mod\ 4) \\ \Pi(\mathbb{I}(L(\lfloor\frac{a}{4}\rfloor),x),\mathbb{I}(R(\lfloor\frac{a}{4}\rfloor),x)), & \text{ако } a\equiv 2\ (mod\ 4) \\ \underbrace{\mathbb{I}(\lfloor\frac{a}{4}\rfloor,\ldots,\mathbb{I}(\lfloor\frac{a}{4}\rfloor}_{L(x)\text{ пъти}},R(x))\ldots), & \text{ако } a\equiv 3\ (mod\ 4). \end{cases} \tag{3.21}$$

Дали можем да твърдим, че в такъв случай $\mathbb{I}$ е точно функцията, която ни трябва? Това беше основният ни замисъл, докато я конструирахме, но да го проверим и формално.

Твърдение 3.15. Нека $\mathbb{I}$ удовлетворява (3.21). Тогава $\mathbb{I}(a,x) = f_a(x)$ за всяко a и x.

Доказателство. С пълна индукция относно a показваме, че

$$\forall x\ \mathbb{I}(a,x) = f_a(x).$$

Ще пропуснем доказателството, защото всички необходими разсъждения вече проведохме по-горе. □

От горното твърдение получаваме, в частност, че има единствена функция, която удовлетворява (3.21) и това е $\mathbb{I}$. Остана да покажем, че тя е изчислима.

Твърдение 3.16. Функцията $\mathbb{I}$ е рекурсивна.

Доказателство. Ще използваме теоремата за определимост по рекурсия, което означава, че ще търсим $\mathbb{I}$ във вида $\varphi_e^{(2)}$ за някое e. По-точно, целта ни е да видим, че една такава функция $\varphi_e^{(2)}$ удовлетворява рекурсивната схема (3.21), с която дефинирахме $\mathbb{I}$, т. е. за нея е вярно, че

$$\varphi_e^{(2)}(a,x) = \underbrace{\begin{cases} x+1, & \text{ако } a = 0 \\ 0, & \text{ако } a = 4 \\ x, & \text{ако } a = 8 \\ L(x), & \text{ако } a = 12 \\ R(x), & \text{ако } a \equiv 0 \ (mod\ 4) \ \&\ \lfloor \frac{a}{4} \rfloor \geq 4 \\ \varphi_e^{(2)}(L(\lfloor \frac{a}{4} \rfloor), \varphi_e^{(2)}(R(\lfloor \frac{a}{4} \rfloor), x)), & \text{ако } a \equiv 1 \ (mod\ 4) \\ \Pi(\varphi_e^{(2)}(L(\lfloor \frac{a}{4} \rfloor), x), \varphi_e^{(2)}(R(\lfloor \frac{a}{4} \rfloor), x)), & \text{ако } a \equiv 2 \ (mod\ 4) \\ \underbrace{\varphi_e^{(2)}(\lfloor \frac{a}{4} \rfloor, \ldots, \varphi_e^{(2)}(\lfloor \frac{a}{4} \rfloor}_{L(x) \text{ пъти}}, R(x)) \ldots), & \text{ако } a \equiv 3 \ (mod\ 4). \end{cases}}_{F(e,a,x)}$$

(3.22)

Да означим функцията вдясно с F. Искаме да покажем, че тя е изчислима. Минаваме стандартно през универсалната функция Φ_2:

$$F(e,a,x) \simeq \underbrace{\begin{cases} x+1, & \text{ако } a = 0 \\ 0, & \text{ако } a = 4 \\ x, & \text{ако } a = 8 \\ L(x), & \text{ако } a = 12 \\ R(x), & \text{ако } a \equiv 0 \ (mod\ 4) \ \&\ \lfloor \frac{a}{4} \rfloor \geq 4 \\ \Phi_2(e, L(\lfloor \frac{a}{4} \rfloor), \Phi_2(e, R(\lfloor \frac{a}{4} \rfloor), x)), & \text{ако } a \equiv 1 \ (mod\ 4) \\ \Pi(\Phi_2(e, L(\lfloor \frac{a}{4} \rfloor), x), \Phi_2(e, R(\lfloor \frac{a}{4} \rfloor), x)), & \text{ако } a \equiv 2 \ (mod\ 4) \\ \underbrace{\varphi_e^{(2)}(\lfloor \frac{a}{4} \rfloor, \ldots, \varphi_e^{(2)}(\lfloor \frac{a}{4} \rfloor}_{L(x) \text{ пъти}}, R(x)) \ldots), & \text{ако } a \equiv 3 \ (mod\ 4). \end{cases}}_{g(e,a,x)}$$

Да означим с g функцията от последния ред на дефиницията на F. Искаме да покажем, че тя е изчислима. Затова да разгледаме по-общата функция

$$G(n,e,a,x) \overset{\text{деф}}{\simeq} \underbrace{\varphi_e^{(2)}(a, \ldots \varphi_e^{(2)}}_{n \text{ пъти}}(a,x) \ldots).$$

За G имаме следната примитивно рекурсивна схема:

$$\begin{cases} G(0,e,a,x) = x \\ G(n+1,e,a,x) \simeq \varphi_e^{(2)}(a, G(n,e,a,x)) \simeq \Phi_2(e,a,G(n,e,a,x)). \end{cases}$$

откъдето се вижда, че G е изчислима. Но $g(e,a,x) \simeq G(L(x), e, \lfloor \frac{a}{4} \rfloor, R(x))$, следователно и g е изчислима. Сега вече можем да твърдим, че и F е

изчислима. Но тогава по теоремата за определимост по рекурсия ще съществува индекс e_0, такъв че

$$\varphi^{(2)}_{e_0}(a, x) \simeq F(e_0, a, x)$$

за всяко a и x. Това означава, че $\varphi^{(2)}_{e_0}$ удовлетворява равенството (3.22), а оттук и (3.21). Но единствената функция с това свойство е $\mathbb{I}$. Следователно $\mathbb{I} = \varphi^{(2)}_{e_0}$, което значи, че $\mathbb{I}$ е изчислима. Освен това тя очевидно е и тотална, и значи общо $\mathbb{I}$ е рекурсивна. □

Задача за ЕК. Докажете, че $\mathbb{I}$ е рекурсивна, като използвате първата теорема за рекурсия.