Лекция 7: Теорема за нормален вид на Клини. S_n^m-теорема

3.2.2 Две важни приложения на диагоналния метод на Кантор

Ще формулираме две твърдения, свързани с универсални функции, които се доказват с диагоналния метод на Кантор. Въпреки че доказателствата им са елементарни, смисълът на тези твърдения е важен компютърната наука.

Да означим с $\mathcal{R}_n$ класа на всички n-местни рекурсивни функции. Оказва се, че за разлика от частично рекурсивните (изчислимите) функции, класът $\mathcal{R}_n$ няма универсална функция.

Твърдение 3.5. За всяко $n \geq 1$ класът $\mathcal{R}_n$ на n-местните рекурсивни функции няма универсална функция.

Доказателство. Първо ще разгледаме случая $n = 1$, с който ще илюстрираме отново класическата диагонална конструкция на Кантор. Да допуснем, че $U(a, x)$ е универсална за $\mathcal{R}_1$. Тогава U е изчислима. Освен това тя е тотална, защото за всяко фиксирано a функцията $\lambda x.U(a, x)$ е от класа $\mathcal{R}_1$, и следователно $!U(a, x)$ за всяко x. Получихме, че $\forall a \forall x !U(a, x)$, т.е. U наистина е тотална функция. Ние приехме, че тя е изчислима, и значи U е рекурсивна функция.

Нека за всяко фиксирано a с f_a да означим функцията

$$f_a = \lambda x.U(a, x).$$

От условие 1) на дефиницията за УФ имаме, че редицата

$$f_0, f_1, \ \ldots\ , f_a, \ \ldots \tag{3.7}$$

съдържа всички едноместни рекурсивни функции. Ще построим нова функция $d(x)$, която хем ще е рекурсивна, хем няма да се среща в редицата (3.7), което ще е търсеното противоречие. $d(x)$ е свързана с *диагонала* на таблицата от стойностите на функциите от горната редица, така както я обяснихме в раздел 2.3.2. Да вземем отново

$$d(x) \ = \ f_x(x) + 1$$

за всяко $x \in \mathbb{N}$. Тази функция е рекурсивна, защото можем да я препишем като $d(x) = U(x, x) + 1$ (тук използваме, че по допускане U е рекурсивна). Но щом е рекурсивна, тя е изброена в горната редица (3.7) поне веднъж, т.е. съществува поне едно a, такова че $d = f_a$. Но тогава би трябвало $d(x) = f_a(x)$ за всяко x, което при $x = a$ очевидно се нарушава, защото съгласно избора на d имаме $d(a) = f_a(a) + 1$.

По подобен начин разсъждаваме за произволно n. Допускаме, че класът $\mathcal{R}_n$ има УФ $U_n(a, x_1, \ldots, x_n)$ и полагаме

$$d(x_1, \ldots, x_n) \ = \ U_n(x_1, x_1, x_2, \ldots, x_n) + 1.$$

Разбира се, функцията U_n също е рекурсивна. Тогава и d ще е рекурсивна, което означава, че $d = \lambda \bar{x}.U(a, \bar{x})$ за някое a. В частност, в точката $\underbrace{(a, \ldots, a)}_{n \text{ пъти}}$ ще имаме $d(a, \ldots, a) = U(a, a, \ldots, a)$, което противоречи на дефиницията на d, според която

$$d(a, \ldots, a) \ \stackrel{\text{деф}}{=} \ U(a, a, \ldots, a) + 1.$$

□

Това твърдение ни казва, че е безсмислено да се опитваме да създаваме език за програмиране, чиито програми завършват при всеки вход, и който е такъв, че всяка тотална изчислима функция е програмируема на този език. (Без това второ условие въпросният език може да е съвсем тривиален — примерно в него да няма цикли). Наистина, ако такъв език съществуваше, всички рекурсивни функции (и само те) щяха да се пресмятат с програмите от този език. Но тогава щяхме да можем да построим УФ за тези функции, следвайки идеята от доказателството на теоремата за универсалната функция за нашия език за МНР. Но това, както видяхме в *Твърдение* 3.5, е невъзможно.

Да означим с $\mathcal{PR}_n$ класа на n-местните примитивно рекурсивни функции. Този клас има универсална функция и ние ще я построим понататък в курса. Твърдението, което сега ще докажем, казва, обаче, че никоя УФ за $\mathcal{PR}_n$ не може да бъде примитивно рекурсивна.

Твърдение 3.6. Нека U е универсална функция за класа $\mathcal{PR}_n$ на n-местните примитивно рекурсивни функции, $n \geq 1$. Тогава U не е примитивно рекурсивна.

Доказателство. Да фиксираме $n \geq 1$. Ще следваме схемата от доказателството на предното твърдение. Допускаме, че U_n е примитивно рекурсивна УФ за класа $\mathcal{PR}_n$. Отново разглеждаме "диагоналната" функция

$$d(x_1, \ldots, x_n) = U(x_1, x_1, x_2, \ldots, x_n) + 1.$$

Тя също е примитивно рекурсивна и значи $d = \lambda \bar{x}.U(a, \bar{x})$ за някое a. В частност, $d(a, \ldots, a) = U(a, a, \ldots, a)$, което влиза в противоречие с избора на d. □

Ако се питате защо със същото диагонално разсъждение не може да се докаже, че никоя УФ за класа на изчислимите функции не е изчислима, краткият отговор е: защото изчислимите функции са частични.

Наистина, разсъждавайки за $n = 1$ за по-просто, да вземем същата

$$d(x) \simeq \Phi_1(x, x) + 1.$$

Тази функция е изчислима, съгласно теоремата за универсалната функция. Следователно $d = \lambda x.\Phi_1(a, x)$ за поне едно a. Оттук при $x = a$ ще имаме

$$d(a) \simeq \Phi_1(a, a),$$

и едновременно с това от дефиницията на d:

$$d(a) \simeq \Phi_1(a, a) + 1.$$

Тези равенства, обаче, не си противоречат, защото са *условни*. Те могат да са в сила едновременно, когато $\neg!\Phi_1(a, a)$. Нещо повече, понеже знаем, че Φ_1 *е* изчислима, можем със сигурност да твърдим, че $\neg!\Phi_1(a, a)$, и това е вярно за всеки индекс a на функцията d.

3.2.3 Потенциално рекурсивни функции

Ще казваме, е функцията f е *потенциално рекурсивна*, ако тя може да бъде продължена до рекурсивна функция, с други думи, ако съществува рекурсивна функция g, такава че $f \subseteq g$.

Едно естествено достатъчно условие за потенциалната рекурсивност на f се задава чрез условието за рекурсивност на нейната дефиниционна област $Dom(f)$. Но тъй като $Dom(f)$ е множество, а ние формално имаме дефиниция за рекурсивност само на предикат, можем да считаме, че множеството $Dom(f)$е *рекурсивно*, ако е рекурсивен предикатът d_f, дефиниран като:

$$d_f(\bar{x}) \overset{\text{деф}}{\iff} \bar{x} \in Dom(f).$$

Да отбележим, че такива множества — с рекурсивен предикат за принадлежност към тях — обикновено се наричат *разрешими*. Тях ще изучаваме много подробно в следващата глава.

Задача 3.1. Докажете, че всяка частично рекурсивна функция, която има рекурсивна дефиниционна област, е потенциално рекурсивна.

Доказателство. Нека частично рекурсивната функция f има рекурсивна дефиниционна област. Да дефинираме g по следния начин:

$$g(\bar{x}) \simeq \begin{cases} f(\bar{x}), & \text{ако } \bar{x} \in Dom(f) \\ 0, & \text{ако } \bar{x} \notin Dom(f). \end{cases}$$

Ясно е, че $f \subseteq g$, освен това g очевидно е тотална. Тя е и частично рекурсивна, което се вижда например от представянето

$$g(\bar{x}) \simeq f(\bar{x}).\overline{sg}(\chi_{d_f}(\bar{x})).$$

Следователно g е рекурсивна. Разбира се, тази функция не е единственото рекурсивно продължение на f. Може да получите безброй много други, ако положите например $g(\bar{x}) = c$ за всяко $\bar{x} \notin Dom(f)$. □

В сила е и следното обръщане на горната задача:

Задача 3.2. Докажете, че ако f е потенциално рекурсивна функция и множеството $Dom(f)$ е рекурсивно, то f е частично рекурсивна.

Доказателство. Нека f се продължава от рекурсивната функция g. Тогава f можем да представим по следния начин (приемайки, че тя има n аргумента):

$$f(\bar{x}) \simeq \begin{cases} g(\bar{x}), & \text{ако } \bar{x} \in Dom(f) \\ \emptyset^{(n)}(\bar{x}), & \text{ако } \bar{x} \notin Dom(f). \end{cases}$$

Тъй като функциите g и $\emptyset^{(n)}$ са частично рекурсивни, а множеството $Dom(f)$ е рекурсивно, то съгласно *Твърдение* 1.11, f ще е частично рекурсивна. □

Разбира се, можем да говорим и за потенциална *примитивно* рекурсивност на една функция: казваме, че функцията f е *потенциално примитивно рекурсивна*, ако f има примитивно рекурсивно продължение.

Всички частично рекурсивни функции, които сме разгледали дотук (като например тези от последната задача от раздел 1.7.5), всъщност са потенциално примитивно рекурсивни. Такива са, да кажем, следните две функции g и h, които се продължават от примитивно рекурсивната функция $qt(x,y)$:

$$g(x,y) \simeq \begin{cases} [\frac{y}{x}], & \text{ако } x>0 \\ \neg!, & \text{иначе,} \end{cases} \quad \text{и} \quad h(x,y) \simeq \begin{cases} \frac{y}{x}, & \text{ако } x>0 \ \& \ x \text{ дели } y \\ \neg!, & \text{иначе,} \end{cases}$$

Да отбележим, че дефиниционните области на тези две функции също са примитивно рекурсивни.

Не всяка частично рекурсивна функция има рекурсивно продължение. Сега ще конструираме една такава функция, като се възползваме от току-що доказаната теорема за универсалната функция.

Твърдение 3.7. Съществува частично рекурсивна функция, която не може да бъде продължена до рекурсивна.

Доказателство. Ще конструираме едноместна функция с това свойство, като тръгнем от "най-сложната" функция — универсалната функция $\Phi_1(a,x)$. Да вземем отново функцията

$$d(x) \overset{\text{деф}}{\simeq} \Phi_1(x,x)+1.$$

По-горе се убедихме, че d е изчислима. Ще покажем, че тя не може да се продължи до рекурсивна функция. Наистина, да допуснем, че съществува рекурсивна g, такава че $d \subseteq g$. Функцията g е рекурсивна, което ще рече и изчислима, и значи тя има индекс. Нека a е някакъв индекс на g. Понеже $g=\varphi_a$, то φ_a е тотална и в частност, $\varphi_a(a)$ е дефинирана. Но тогава е дефинирана и $d(a)$, тъй като

$$d(a) \simeq \Phi_1(a,a)+1 \simeq \varphi_a(a)+1.$$

От $d \subseteq g$ и $!d(a)$ получаваме, че $d(a)=g(a)$, което обаче е невъзможно, тъй като съгласно избора на d:

$$d(a)=\varphi_a(a)+1=g(a)+1 \neq g(a).$$

Забележка. Ако си мислите, че d не може да бъде продължена до рекурсивна функция, защото може да има много големи стойности, приложете горните разсъждения към 0-1-функцията $\overline{sg}(\Phi_1(x,x))$. Пак ще стигнете до противоречие. □

3.3 Теорема за нормален вид на Клини

Следващото твърдение, известно като *теорема за нормален вид на Клини*, дава едно стандартизирано представяне на всяка изчислима функция, което се оказва полезно в много случаи.

Теорема 3.3. (Теорема за нормален вид на Клини) Нека $n \geq 1$. Съществува примитивно рекурсивна функция $T_n(a, \bar{x}, z)$, такава че

$$\varphi_a^{(n)}(\bar{x}) \simeq L(\mu z[T_n(a, \bar{x}, z) = 0]) \tag{3.8}$$

за всички естествени a и $\bar{x}$.

Забележка. Функцията T_n от горното представяне обикновено се дефинира като $0,1$-функция и се тълкува като предикат, често наричан *предикат на Клини*.

Доказателство. При фиксирано $n \geq 1$ да означим с $A_n(a, \bar{x}, y, t)$ следния предикат:

$$A_n(a, \bar{x}, y, t) \overset{\text{деф}}{\iff} P_a \text{ спира върху вход } \bar{x} \text{ за } \leq t \text{ такта с резултат } y.$$

Този предикат е примитивно рекурсивен, защото можем да го представим по следния начин:

$$A_n(a, \bar{x}, y, t) \iff (Q_n(a, \bar{x}, t))_0 > lh(a) \ \& \ (Q_n(a, \bar{x}, t))_1 = y.$$

Сега дефинираме предиката $T_n(a, \bar{x}, z)$ като:

$$T_n(a, \bar{x}, z) \overset{\text{деф}}{\iff} A_n(a, \bar{x}, L(z), R(z)).$$

T_n формално е предикат, но ние ще го отъждествяваме с характеристичната му функция

$$\chi_{T_n}(a, \bar{x}, z) = \begin{cases} 0, & \text{ако } T_n(a, \bar{x}, z) \\ 1, & \text{ако } \neg T_n(a, \bar{x}, z). \end{cases}$$

Да се убедим, че този предикат удовлетворява равенството (3.8). За целта при фиксирано a да означим с g_a функцията, която е в дясната част на това равенство:

$$g_a(\bar{x}) \simeq L(\mu z[T_n(a, \bar{x}, z) = 0]).$$

Трябва да покажем, че $g_a = \varphi_a^{(n)}$. За целта отново ще се възползваме от *Задача* 1.1, според която е достатъчно да съобразим, че

$$g_a \subseteq \varphi_a^{(n)} \quad \text{и} \quad Dom(\varphi_a^{(n)}) \subseteq Dom(g_a).$$

1) $g_a \subseteq \varphi_a^{(n)}$: Нека $g_a(\bar{x}) \simeq y$. Тогава в частност $g_a(\bar{x})$ е дефинирано, и значи съществува $z : T_n(a, \bar{x}, z) = 0$ и $L(z) = y$. От дефиницията на T_n ще имаме, че е вярно $A_n(a, \bar{x}, y, R(z))$, което означава, че $\varphi_a^{(n)}(\bar{x}) \simeq y$.

2) $Dom(\varphi_a^{(n)}) \subseteq Dom(g_a)$: Да вземем произволно $\bar{x} \in Dom(\varphi_a^{(n)})$, т.е. такова, че $!\varphi_a^{(n)}(\bar{x})$. Тогава за някои y и t ще е изпълнено

$$A_n(a, \bar{x}, y, t),$$

а оттук и $T_n(a, \bar{x}, \Pi(y, t))$ ще е вярно. Получихме, че съществува z, за което $T_n(a, \bar{x}, z) = 0$, и следователно $!g_a(\bar{x})$.

С това показахме, че $g_a = \varphi_a^{(n)}$, с други думи, доказахме теоремата. □

Тъй като всяка изчислима функция е частично рекурсивна, от тази теорема получаваме, че и за всяка *частично рекурсивна* функция $f \in \mathcal{F}_n$ ще е вярно условието (3.8), с други думи, всяка частично рекурсивна функция $f \in \mathcal{F}_n$ ще се представя във вида

$$f = \lambda \bar{x}.L(\mu z[T_n(a, \bar{x}, z) = 0])$$

за някое a. Това означава, в частност, че всяка такава функция може да се получи от базисните функции с *една единствена* минимизация (нещо, което отбелязахме и по-рано като *Следствие* 3.1 от теоремата за еквивалентност). Тази единствена минимизация се прилага към примитивно рекурсивната функция T_n, а след нея се прилага още една примитивно рекурсивна функция — функцията L.

Възниква въпросът дали е възможно да махнем външната примитивно рекурсивна функция, т.е. дали *всяка* частично рекурсивна f може да се представи като

$$f(\bar{x}) \simeq \mu z[g(\bar{x}, z) = 0],$$

където g е някаква примитивно рекурсивна (или дори само рекурсивна) функция. Оказва се, че това не е възможно. За да го съобразим, най-напред да забележим, че дефиниционното множество на универсалната функция Φ_1 е доста сложно:

Задача 3.3. Докажете, че *не* е рекурсивен предикатът $p(x)$, дефиниран като:

$$p(x) \overset{\text{деф}}{\iff} !\Phi_1(x, x)$$

за всяко $x \in \mathbb{N}^n$.

Доказателство. Да допуснем, че горният предикат е рекурсивен и да разгледаме отново диагоналната функция d:

$$d(x) \overset{\text{деф}}{\simeq} \Phi_1(x, x) + 1.$$

От доказателството на *Твърдение* 3.7 знаем, че тази функция не може да се продължи до рекурсивна. От нашето допускане, че предикатът p е рекурсивен, следва, че домейнът на d също ще бъде рекурсивен, защото за всяко естествено x:

$$!d(x) \iff !\Phi_1(x,x) \overset{\text{деф}}{\iff} p(x).$$

Но f е частично рекурсивна, а нейният домейн е разрешим. Тогава съгласно *Задача* 3.1 функцията d ще има рекурсивно продължение, което, както се убедихме, не е възможно. □

Вече сме готови да конструираме търсения от нас контрапример.

Задача 3.4. Докажете, че съществува частично рекурсивна функция f, такава че за никоя рекурсивна функция g не е вярно, че

$$f(x) \simeq \mu z[g(x,z) = 0] \qquad \text{за всяко } x \in \mathbb{N}. \tag{3.9}$$

Доказателство. Да вземем функцията

$$f(x) \simeq \begin{cases} x, & \text{ако } !\Phi_1(x,x) \\ \neg!, & \text{иначе.} \end{cases}$$

f е частично рекурсивна — това се вижда например от представянето $f(x) \simeq x.C_1^1(\Phi_1(x,x))$, или следва директно от *Твърдение* 1.11.

Сега да допуснем, че за някоя рекурсивна функция g е в сила равенството (3.9). Тогава за всяко естествено x ще е вярно, че:

$$!f(x) \iff g(x,x) = 0.$$

Да проверим тази еквивалентност, като използваме представянето (3.9) на f. Наистина, ако $!f(x)$, то $f(x) \simeq x$ и съгласно (3.9), $g(x,x) = 0$. Ако $\neg!f(x)$, тогава отново от (3.9) следва, че $\forall z g(x,z) > 0$. В частност, $g(x,x) > 0$.

Окончателно, получихме, че за всяко x:

$$!\Phi_1(x,x) \overset{\text{деф } f}{\iff} !f(x) \iff g(x,x) = 0.$$

Но това означава, че предикатът

$$p(x) \overset{\text{деф}}{\iff} !\Phi_1(x,x)$$

е рекурсивен — противоречие със *Задача* 3.3.

3.4 S_n^m-теорема

Тази теорема е известна още като *теорема за параметризацията* или *Лема за транслация*. По-нататък ще стане ясно откъде идват тези имена. За да си изясним "на първо четене" смисъла на S_n^m-теоремата, ще започнем с едно частно наблюдение, което после ще обобщим. Преди това да напомним едно означение, което въведохме по-рано:

$\varphi_a^{(n)} \overset{\text{деф}}{=} n$-местната функция, която се пресмята от програмата P_a.

При $n = 1$ горният индекс обикновено се пропуска, т.е.

$\varphi_a \overset{\text{деф}}{=}$ едноместната функция, която се пресмята от програмата P_a.

Ако $f = \varphi_a^{(n)}$, то a нарекохме *индекс* на f.

Сега да вземем произволна изчислима функция на два аргумента $f(a, x)$. Ако за момент си представим, че първият ѝ аргумент a е фиксиран, получаваме едноместната функция

$$f_a = \lambda x.f(a, x).$$

Тази функция, разбира се, също е изчислима, и значи за нея съществува индекс b, такъв че $f_a = \varphi_b$. Излезе, че

$$\forall a \exists b \; f_a = \varphi_b,$$

което означава (с използване на аксиомата за избора), че съществува тотална функция, да кажем h, такава че за всяко a:

$$f_a = \varphi_{h(a)}.$$

Това, което S_n^m-теоремата добавя към това наблюдение е, че функцията h може да се избере *рекурсивна* (и дори примитивно рекурсивна). От доказателството ще се види още, че функцията h се *конструира* (т.е. доказателството на теоремата е конструктивно).

За случая на двуместна изчислима функция $f(a, x)$ S_n^m-теоремата казва, че съществува примитивно рекурсивна функция h, такава че за всяко a и x:

$$\varphi_{h(a)}(x) \simeq f(a, x).$$

Този резултат би могъл да се обобщи по два начина. Първият е директен — вместо $f(a, x)$ да разглеждаме $f(a_1, \ldots, a_m, x_1, \ldots, x_n)$, т.е. параметрите на конструкцията да бъдат $a_1, \ldots, a_m$.

Вторият начин е да направим т. нар. *равномерно* обобщение. То се състои в следното: щом $f(a, x)$ е изчислима, значи можем да си я мислим във

вида $\varphi_e^{(2)}(a,x)$ за някое e, с други думи, да разглеждаме f, зададена чрез своя индекс e. Оказва се, че "в играта" можем да включим и този индекс e. С други думи, вярно е не просто, че

$$\forall e \exists h\ \varphi_{h(a)} = \lambda x.\varphi_e^{(2)}(a,x),$$

а нещо доста по-силно — че функцията h може да се избере отнапред и да е *обща* за всички индекси e, или все едно, горните два квантора да се разменят:

$$\exists h \forall e\ \varphi_{h(e,a)} = \lambda x.\varphi_e^{(2)}(a,x).$$

3.4.1 S_n^m-теорема

Ето и най-общата формулировка на S_n^m-теоремата, която включва и двата начина за обобщаване, които обсъждахме по-горе:

Теорема 3.4. **(S_n^m-теорема)** Нека $m \geq 1$ и $n \geq 1$. Съществува примитивно рекурсивна функция S_n^m, такава че:

$$\varphi^{(n)}_{S_n^m(e,a_1,\ldots,a_m)}(x_1,\ldots,x_n) \simeq \varphi_e^{(m+n)}(a_1,\ldots,a_m,x_1,\ldots,x_n)$$

за всички естествени $e, a_1, \ldots, a_m, x_1, \ldots, x_n$.

Доказателство. За начало ще разгледаме случая $m = n = 1$. Трябва да построим примитивно рекурсивна функция S_1^1, такава че за всяко e, a и x:

$$\varphi_{S_1^1(e,a)}(x) \simeq \varphi_e^{(2)}(a,x).$$

При фиксирани e и a, нека Q е МНР програма, която пресмята функцията $\lambda x.\varphi_e^{(2)}(a,x)$. Как работи Q? При вход $(x,0,0,\ldots)$ тя симулира P_e върху нейния вход $(a,x,0,0,\ldots)$

За целта най-напред Q трябва да прехвърли x във втория регистър, а в първия да зареди a. Това става с редицата от инструкции

$$X_2 := X_1,\ X_1 := 0,\ \underbrace{X_1 := X_1 + 1,\ \ldots\ , X_1 := X_1 + 1}_{a \text{ пъти}}$$

След тези инициализации Q трябва да започне да изпълнява инструкциите на P_e, които, обаче, преди това *трябва да бъдат преадресирани*. Това се налага заради тези $a+2$ на брой оператора, които слагаме в началото на програмата. Заради тях всеки оператор за преход $J(m,n,q)$ на P_e трябва да се замени с $J(m,n,q+a+2)$.

Нека за определеност $P_e\colon I_0,\ldots,I_k$. За всяко $l=0,\ldots,k$ нека

$$I_l' = \begin{cases} I_l, & \text{ако } I_l \text{ е оператор за присвояване} \\ J(m,n,q+a+2), & \text{ако } I_l \text{ е } J(m,n,q). \end{cases}$$

Значи Q трябва да изглежда така:

$$T(2,1),\ Z(1),\ \underbrace{S(1),\ \ldots\ ,S(1),}_{a \text{ пъти}}\ \underbrace{I_0',\quad I_1',\quad \ldots\quad ,I_k'}_{\text{преадресираните инстр. на } P_e}$$

За да си даваме сметка, че нашата програма Q зависи от двата параметъра e и a, нека я означаваме с $Q_{e,a}$. Сега задачата ни се свежда до това да покажем, че кодът $\gamma(Q_{e,a})$ на програмата $Q_{e,a}$ зависи "гладко" (в случая — примитивно рекурсивно) от e и a. Тогава, ако положим

$$S_1^1(e,a) \ :=\ \gamma(Q_{e,a}),$$

то $S_1^1(e,a)$ ще е примитивно рекурсивна, и освен това едноместната функция, която програмата с код $S_1^1(e,a)$ пресмята, ще е точно тази, която се пресмята от $Q_{e,a}$, т.е. точно $\lambda x.\varphi_e^{(2)}(a,x)$. Така ще имаме, че за всяко x

$$\varphi_{S_1^1(e,a)}(x) \simeq \varphi_e^{(2)}(a,x).$$

И тъй като e и a са произволни, то горното условно равенство ще е изпълнено за всяко e, a и x, което е точно S_n^m-теоремата при $m=n=1$.

Сега вече имаме мотивация да се заемем с доказателството на примитивната рекурсивност на функцията $\lambda e,a.\gamma(Q_{e,a})$.

Да означим с tr (от ***transform***) функцията, която преобразува кода на оператор за преход $J(m,n,q)$ по следния начин:

$$tr(\beta(J(m,n,q)),b) \ =\ \beta(J(m,n,q+b)).$$

С други думи, $tr(z,b)$ измества с b единици адреса на оператора за преход с код z. Функцията $tr(z,b)$ е примитивно рекурсивна, защото (като си спомним детайлите за код на оператор за преход (2.1)), можем да я запишем като

$$tr(z,b) \ =\ 4.\Pi_3(J_1^3([\frac{z}{4}]),J_2^3([\frac{z}{4}]),J_3^3([\frac{z}{4}])+b)+3.$$

Нека $prim$ е функцията, която преобразува кода на всеки оператор I на P_e в кода на I'. Тази функция също е примитивно рекурсивна, защото

$$prim(z) = \begin{cases} z, & \text{ако } rem(4,z) < 3 \\ tr(z, a+2), & \text{ако } rem(4,z) = 3. \end{cases}$$

Да напомним, че номерът k на последния оператор на $P_e : I_0, \ldots, I_k$ е точно $lh(e)$. Освен това, ако I_l е l-тият оператор на P_e, то неговият код, съгласно определението (2.2), ще е $mem(e,l)$, и следователно кодът на I_l' ще е $prim(mem(e,l))$.

Да препишем отново редицата от инструкции на програмата $Q_{e,a}$:

$$T(2,1),\ Z(1),\ \underbrace{S(1),\ \ldots\ ,S(1),}_{a \text{ пъти}}\ \underbrace{I_0',\quad I_1',\quad \ldots\quad ,I_k'}_{\text{преадресираните инстр. на } P_e}$$

Да означим с $f(e,a,l)$ функцията, която за всяко $l \leq a+2+k$ дава кода на l-тия оператор на тази програма, по-точно нека

$$f(e,a,l) = \begin{cases} \text{кода на } l\text{-ия оператор на } Q_{e,a}, & \text{ако } l \leq a+2+k \\ 0, & \text{ако } l > a+2+k. \end{cases}$$

Да съобразим, че и тази функция е примитивно рекурсивна. За целта да забележим, че за $l \in \{a+2, \ldots, a+2+k\}$ имаме, че l-тият оператор на $Q_{e,a}$ е точно I'_{l-a-2}, и този оператор, съгласно това, което отбелязахме по-горе, е с код $prim(mem(e, l-a-2))$. Така за $f(e,a,l)$ получаваме следната дефиниция с разглеждане на случаи:

$$f(e,a,l) = \begin{cases} \beta(T(2,1)), & \text{ако } l = 0 \\ \beta(Z(1)), & \text{ако } l = 1 \\ \beta(S(1)), & \text{ако } 2 \leq l < a+2 \\ prim(mem(e, l-a-2)), & \text{ако } a+2 \leq l \leq a+2+lh(e) \\ 0, & \text{ако } l > a+2+lh(e). \end{cases}$$

Оттук се вижда, че f е примитивно рекурсивна.

Сега вече за кода $\gamma(Q_{e,a})$ на програмата $Q_{e,a}$ ще имаме, съгласно дефиницията (2.2) на код на програма:

$$\gamma(Q_{e,a}) = \tau(\langle f(e,a,0),\ \ldots\ , f(e,a,a+2+lh(e))\rangle).$$

Но този израз е точно историята H_f на f, съгласно определение (1.11). Сега вече $\gamma(Q_{e,a})$ придобива вида

$$\gamma(Q_{e,a}) \ = \ H_f(e,a,a+2+lh(e)).$$

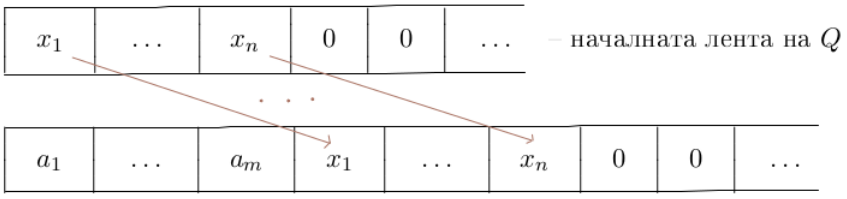
От *Твърдение* 1.25 знаем, че ако f е примитивно рекурсивна, то и нейната история H_f също ще е, откъдето финално получаваме, че функцията $\lambda e, a.\gamma(Q_{e,a})$ е примитивно рекурсивна.

С това приключва доказателството на S_n^m-теоремата за случая $m=n=1$.

За произволни m и n разсъжденията са много подобни: при фиксирани $e, a_1, \ldots, a_m$ разглеждаме програмата $Q = Q_{e,a_1,\ldots,a_m}$, която пресмята функцията

$$\lambda x_1, \ldots, x_n.\varphi_e^{(m+n)}(a_1, \ldots, a_m, x_1, \ldots, x_n).$$

Този път началната конфигурация на Q е $(x_1, \ldots, x_n, 0, 0, \ldots)$ и тя работи така, както P_e работи върху $(a_1, \ldots, a_m, x_1, \ldots, x_n, 0, 0, \ldots)$, или сега картинката е:



началната лента на P_e

Значи Q трябва най-напред да прехвърли съдържанието на първите си n регистъра в регистри с номера $m+1, \ldots, m+n$. Добре е да започне от последния към първия регистър (съобразете защо):

$$X_{m+n} := X_n, \ \ldots \ , X_{m+1} := X_1$$

Следва блок от инструкции, с които първите m регистъра се зареждат с $a_1, \ldots, a_m$:

$$X_1{:=}0, \underbrace{X_1{:=}X_1{+}1, \ldots, X_1{:=}X_1{+}1}_{a_1 \text{ пъти}}, \ldots, X_m{:=}0, \underbrace{X_m{:=}X_m{+}1, \ldots, X_m{:=}X_m{+}1}_{a_m \text{ пъти}}$$

Това са общо $a_1{+}\cdots{+}a_m{+}m$ на брой инструкции, които заедно с горните n правят общо $a_1+\cdots+a_m+m+n$ начални инструкции. Оттук нататък доказателството върви по същия начин, както при случая $m = n = 1$, с тази разлика, че отместването в адресите на операторите за преход трябва да е не $a+2$, а $a_1 + \cdots + a_m + m + n$. □

Задача 3.5. (Задача за ЕК) Докажете, че S_n^m-функцията в S_n^m-теоремата може да се избере така, че да не зависи от n.

3.4.2 Слаба S_n^m-теорема

За повечето от приложенията на S_n^m-теоремата ще ни е достатъчна следната "неравномерна" нейна версия, с която всъщност започнахме тази тема. Тя се получава, когато не се интересуваме от индекса на функцията, която ще параметризираме.

Теорема 3.5. (Слаба S_n^m-теорема) Нека $f(a_1, \ldots, a_m, x_1, \ldots, x_n)$ е изчислима. Тогава съществува примитивно рекурсивна функция $h(a_1, \ldots, a_m)$, такава че за всяко $\bar{a} \in \mathbb{N}^m, \bar{x} \in \mathbb{N}^n$ е изпълнено:

$$\varphi^{(n)}_{h(a_1,\ldots,a_m)}(x_1, \ldots, x_n) \simeq f(a_1, \ldots, a_m, x_1, \ldots, x_n).$$

Доказателство. Щом $f(\bar{a}, \bar{x})$ е изчислима, то тя има индекс, т.е. съществува e_0, такова че $f = \varphi^{(m+n)}_{e_0}$. Да положим

$$h(\bar{a}) \overset{\text{деф}}{=} S_n^m(e_0, \bar{a}).$$

От общата S_n^m-теорема имаме, че

$$\varphi^{(n)}_{S_n^m(e,\bar{a})}(\bar{x}) \simeq \varphi^{(m+n)}_e(\bar{a}, \bar{x})$$

за всички естествени $e, \bar{a}, \bar{x}$. Оттук при $e = e_0$ получаваме

$$\varphi^{(n)}_{S_n^m(e_0,\bar{a})}(\bar{x}) \simeq \varphi^{(m+n)}_{e_0}(\bar{a}, \bar{x}) \overset{\text{деф}}{\simeq} f(\bar{a}, \bar{x}), \quad \text{или}$$

$$\varphi^{(n)}_{h(\bar{a})}(\bar{x}) \simeq f(\bar{a}, \bar{x}) \quad \text{за всички естествени } \bar{a}, \bar{x}.$$

□

Всъщност слабата S_n^m-теорема е еквивалентна на S_n^m-теоремата, макар привидно да изглежда по-слаба от нея (а и името ѝ да е такова ◡̈). Поради това и на двете формулировки ще се позоваваме като на "S_n^m-теоремата", а контекстът ще показва коя от двете имаме предвид.

Задача 3.6. Да се докаже, че от слабата S_n^m-теорема следва (първоначалната) S_n^m-теорема.

Решение. Да фиксираме $m \geq 1, n \geq 1$. Искаме да покажем, че съществува примитивно рекурсивна функция S_n^m, за която

$$\varphi^{(n)}_{S_n^m(e,\bar{a})}(\bar{x}) \simeq \varphi^{(m+n)}_e(\bar{a}, \bar{x}).$$

Да приложим слабата S_n^m-теорема към функцията

$$f(e, \bar{a}, \bar{x}) \overset{\text{деф}}{\simeq} \Phi_{m+n}(e, \bar{a}, \bar{x}),$$

като параметризираме по първите ѝ $m+1$ аргумента $(e, a_1, \ldots, a_m)$. Ще получим, че за някоя примитивно рекурсивна функция $h(e, \bar{a})$ е изпълнено

$$\varphi^{(n)}_{h(e,\bar{a})}(\bar{x}) \simeq f(e, \bar{a}, \bar{x}), \quad \text{или все едно}$$

$$\varphi^{(n)}_{h(e,\bar{a})}(\bar{x}) \simeq \underbrace{\Phi_{m+n}(e, \bar{a}, \bar{x})}_{\varphi_e^{(m+n)}(\bar{a},\bar{x})}.$$

Остана да вземем $S^m_n \overset{\text{деф}}{=} h$. □

3.4.3 Приложения

По-съществените приложения на S^m_n-теоремата ще наблюдаваме в теоремите, които ще доказваме по-нататък в курса. Сега ще решим няколко задачи, за да видим как работи на практика тази теорема.

Задача 3.7. Докажете, че съществува примитивно рекурсивна функция h, такава че за всяко естествено n програмата с код $h(n)$ пресмята функцията x^n.

Доказателство. Условието програмата $P_{h(n)}$ да пресмята x^n означава точно, че функцията $\varphi_{h(n)}$ е x^n, т.е. за всяко n и x имаме

$$\varphi_{h(n)}(x) = \underbrace{x^n}_{f(n,x)}.$$

Ако си представяме, че h е "дошла" от слабата S^m_n-теорема, то ясно е, че функцията, към която трябва да приложим тази теорема, е функцията вдясно в горното равенство. Да я означим с $f(n, x)$.

Знаем, че $f(n, x) = x^n$ е примитивно рекурсивна, следователно тя е и изчислима. Тогава по S^m_n-теоремата ще съществува примитивно рекурсивна функция h, такава че за всяко n и x

$$\varphi_{h(n)}(x) = f(n, x), \quad \text{или все едно,} \quad \varphi_{h(n)}(x) = x^n.$$

□

Задача 3.8. Докажете, че съществува примитивно рекурсивна функция l, такава че за всички естествени a и b:

$$\varphi_{l(a,b)}(x) = ax + b.$$

Доказателство. Смисълът на функцията $l(a, b)$ е, че по всяко a и b тя връща код на програма, пресмятаща линейната функция $ax + b$.

За да получим l, тръгваме от функцията $f(a,b,x) = ax+b$, която очевидно е изчислима. Прилагаме S_n^m-теоремата към нея, като този път трябва да параметризираме по първите ѝ два аргумента. Така получаваме, че за някоя примитивно рекурсивна функция $l(a,b)$ ще е в сила

$$\varphi_{l(a,b)}(x) = f(a,b,x) = ax + b$$

за всяко a, b и x. □