Лекция №13:

4.4 Рекурсивни схеми

Синтаксисът на рекурсивните схеми идейно следва дефиницията на рекурсивните програми от езика REC. Разликата е в това, че тези програми бяха дефинирани в естествените числа $\mathbb{N}$ с базисни операции, които означавахме най-общо с op, докато схемите се дефинират за произволна сигнатура Σ. Да я фиксираме отново:

$$\Sigma \;=\; (\mathbf{c}_1, \dots, \mathbf{c}_p;\; \mathbf{f}_1, \dots, \mathbf{f}_s;\; \mathbf{p}_1, \dots,\; \mathbf{p}_t).$$

Терм τ в сигнатурата Σ дефинираме по подобие на старата дефиниция за терм

$$\tau ::= \mathbf{c}_i \;\mid\; X_i \;\mid\; \mathbf{f}_i(\tau, \dots, \tau) \;\mid\; \texttt{if } \mathbf{p}_i(\tau, \dots, \tau) \texttt{ then } \tau \texttt{ else } \tau \;\mid\; F_i(\tau, \dots, \tau)$$

Да отбележим, че в условията $\mathbf{p}_i(\tau, \dots, \tau)$ на термове `if then else` стоят изрази, които (вече) имат булеви стойности.

И тук ще пишем $\tau(X_1, \dots, X_n, F_1, \dots, F_k)$, за да означим, че променливите на τ са измежду $X_1, \dots, X_n, F_1, \dots, F_k$.

Определение 4.7. *Рекурсивна схема* R в сигнатурата Σ ще наричаме синтактичен обект от вида:

$$R \begin{cases} \tau_0(X_1, \dots, X_n, F_1, \dots, F_k) \qquad \texttt{where} \\ F_1(X_1, \dots, X_{m_1}) = \tau_1(X_1, \dots, X_{m_1}, F_1, \dots, F_k) \\ \vdots \\ F_k(X_1, \dots, X_{m_k}) = \tau_k(X_1, \dots, X_{m_k}, F_1, \dots, F_k) \end{cases}$$

в който термовете $\tau_0, \dots, \tau_k$ са термове в сигнатурата Σ.

Сега да вземем произволна структура в тази сигнатура:

$$\mathcal{A} \;=\; (D;\; a_1, \dots, a_p;\; f_1, \dots, f_s;\; p_1, \dots, p_t).$$

В структурата $\mathcal{A}$ рекурсивната схема R се превръща в *рекурсивна програма* $(R, \mathcal{A})$. Приемаме, че семантиката на тази програма е денотационната (или еквивалентно — операционната) семантика *по стойност* на програмата $(R, \mathcal{A})$. За целите, които сме си поставили тук — да покажем, че всяка стандартна схема се транслира в рекурсивна, ще се окаже, че не е от значение дали вземаме семантиката по стойност или по име. Това ще е така, защото всяка стандартна схема ще се превежда в рекурсивна схема от много прост вид — в която няма вложени участия на функционални променливи.

4.5 Транслируемост на стандартните схеми в рекурсивни схеми

4.5.1 Опашкови функции

Да вземем отново произволна стандартна схема S в сигнатурата $\Sigma = (\mathbf{c}_1, \dots, \mathbf{c}_p;\ \mathbf{f}_1, \dots, \mathbf{f}_s;\ \mathbf{p}_1, \dots,\ \mathbf{p}_t)$.

S : $\texttt{input}(X_1, \dots, X_n)$; $\texttt{output}(X_k)$
  $1:\ O_1$
  $\vdots$
  $l:\ O_l$
  $\vdots$
  $q:\ O_q$,

Отново ще предполагаме, че паметта на S е $(X_1, \dots, X_m)$. Да фиксираме и произволна структура $\mathcal{A} = (D;\ a_1, \dots, a_p;\ f_1, \dots, f_s;\ p_1, \dots, p_t)$ в сигнатурата Σ.

За всяко $l = 1, \dots, q$ ще въведем *опашкови функции (tail functions)* g_l, които са изображения от вида:

$$g_l\colon\ D^m \ \multimap\!\!\!\rightarrow\ D.$$

Дефиницията на опашковите функции е в духа на *Определение* (4.1) на семантиката $Sem(S, \mathcal{A})$ — посредством функцията $Step$:

Определение 4.8. За всяко $l = 1, \dots, q$ полагаме

$$g_l(x_1, ..., x_m) \simeq y \ \stackrel{\text{деф}}{\Longleftrightarrow}\ \exists t \exists y_1 ... \exists y_m\ Step^t(\underline{l}, x_1, ..., x_m) \simeq (q, y_1, ..., y_m)\ \&\ y = y_k. \tag{4.2}$$

Смисълът на тези функции, в общи линии, се вижда от определението им — те са това, което ще пресметне програмата, когато я стартираме не от първия, а от l-тия оператор. Какво е тяхното предназначение ще стане ясно по-нататък.

Забележка. Да обърнем внимание, че опашковите функции действат върху цялата памет $(x_1, \dots, x_m)$, т.е. са функции на m аргумента, за разлика от $Sem(S, \mathcal{A})$, която е функция само на входните променливи $(x_1, \dots, x_n) \in D^n$.

Непосредствено от определението получаваме и следната връзка между първата опашкова функция g_1 и $Sem(S, \mathcal{A})$:

$$Sem(S, \mathcal{A})(x_1, \dots, x_n) \ \simeq\ g_1(x_1, \dots, x_n, \underbrace{x_n, ..., x_n}_{m-n \text{ пъти}}). \tag{4.3}$$

Понеже и в дефиницията (4.2) участват квантори за съществуване, трябва отново да докажем коректност на на тази дефиниция, но тъй като тя се показва точно както при доказателството на *Твърдение* 4.1, ще я пропуснем.

За упражнение да намерим опашковите функции на стандартната програма $(S, \mathcal{A}_1)$, определена с блок-схемата 4.2, записана в езика на стандартните схеми.

Задача 4.1. Намерете явния вид на всяка от опашковите функции $g_1, \ldots, g_6$ на следната стандартна програма S:

input(X); output(Y)

1 : $Y := \mathbf{1}$

2 : if $X = 0$ then goto 6 else goto 3

3 : $Y := X.Y$

4 : $X := X \dot{-} 1$

5 : goto 2

6 : stop

Решение. Лесно се съобразява, че първите две опашкови функции са

$$g_1(x, y) = x! \quad \text{и} \quad g_2(x, y) = x!.y.$$

За g_3 се вижда, че връща 0 при $x = 0$, а при $x > 0$ е същата като $g_2(x, y)$, или:

$$g_3(x, y) = \begin{cases} 0, & \text{ако } x = 0 \\ x!.y, & \text{ако } x > 0. \end{cases}$$

Тъй като четвъртият оператор е присвояването $X := X \dot{-} 1$, то би трябвало

$$g_4(x, y) \simeq g_5(x \dot{-} 1, y).$$

Но коя е функцията g_5? Ами тя е същата като g_2, защото операторът, от който се стартира пресмятането на g_5, е goto 2. Сега вече можем да намерим и явния вид на g_4:

$$g_4(x, y) \simeq g_5(x \dot{-} 1, y) \simeq g_2(x \dot{-} 1, y) = (x \dot{-} 1)!.y.$$

Последната функция g_6 очевидно винаги връща y, т.е. $g_6(x, y) = y$. □

Всъщност с разсъжденията по-горе не само намерихме опашковите функции $g_1, \ldots, g_6$, но и забелязахме нещо по-общо, което ще е ключово за теоремата която предстои да формулираме. Това е наблюдението, че в зависимост от вида на оператора O_l, опашковата функция g_l удовлетворява определено равенство. Тези равенства, събрани заедно, ни дават следната *система*, която се удовлетворява от функциите $g_1, \ldots, g_6$:

$$\left|\begin{array}{lcl} g_1(x,y) & = & g_2(x,1) \\ g_2(x,y) & = & \texttt{if } x == 0 \ \texttt{ then } g_6(x,y) \ \texttt{ else } g_3(x,y) \\ g_3(x,y) & = & g_4(x, x.y) \\ g_4(x,y) & = & g_5(x \dot{-} 1, y) \\ g_5(x,y) & = & g_2(x,y) \\ g_6(x,y) & = & y \end{array}\right.$$

Ако искаме да сме една крачка по-близо до рекурсивната програма, еквивалентна на стандартната програма, от която тръгнахме, можем да кажем, че векторът от опашковите функции $(g_1, \ldots, g_6)$ е решение на системата от уравнения, съответстваща на ето тези дефиниции на функционалните променливи $F_1, \ldots, F_6$:

$$\begin{array}{lcl} F_1(X,Y) & = & F_2(X,1) \\ F_2(X,Y) & = & \texttt{if } X == 0 \ \texttt{ then } F_6(X,Y) \ \texttt{ else } F_3(X,Y) \\ F_3(X,Y) & = & F_4(X, X.Y) \\ F_4(X,Y) & = & F_5(X \dot{-} 1, Y) \\ F_5(X,Y) & = & F_2(X,Y) \\ F_6(X,Y) & = & Y \end{array}$$

Тези декларации формират тялото на рекурсивната програма, която искаме да е еквивалентна на стандартната програма от *Задача* 4.1. А коя е главата ѝ? Спомняйки си за равенството (4.3), което дава връзката между първата опашкова функция g_1 и семантиката на една стандартна програма, стигаме до извода, че главата на тази програма трябва да е $F_1(X,X)$. Така получаваме следната рекурсивна програма R:

$$\begin{array}{llcl} R: & F_1(X,X) & & \texttt{where} \\ & F_1(X,Y) & = & F_2(X,1) \\ & F_2(X,Y) & = & \texttt{if } X == 0 \ \texttt{ then } F_6(X,Y) \ \texttt{ else } F_3(X,Y) \\ & F_3(X,Y) & = & F_4(X, X.Y) \\ & F_4(X,Y) & = & F_5(X \dot{-} 1, Y) \\ & F_5(X,Y) & = & F_2(X,Y) \\ & F_6(X,Y) & = & Y \end{array}$$

Разбира се, тази програма може да се оптимизира, като много от функционалните променливи отпаднат. Всъщност остава само променливата F_2 и новата програма, която получаваме, е следната:

$$\begin{array}{llcl} R^*: & F_2(X,1) & & \texttt{where} \\ & F_2(X,Y) & = & \texttt{if } X == 0 \ \texttt{ then } \ Y \ \texttt{ else } \ F_2(X - 1, X.Y) \end{array}$$

Да отбележим, че следвайки горните разсъждения, можем да отидем и още по-далече, като "повдигнем" цялата конструкция на синтактично ниво. Алгоритъмът, който преобразува всяка стандартна *схема* в еквивалентна на нея рекурсивна *схема*, ще опишем подробно при доказателството на теоремата на Маккарти. Завършвайки, тук само ще отбележим, че стандартната схема от *Пример* 4.1, с която започнахме

S : input(X); output(Y)

1 : $Y := \mathbf{c}$

2 : if $\mathbf{p}(X)$ then goto 6 else goto 3

3 : $Y := \mathbf{g}(X, Y)$

4 : $X := \mathbf{f}(X)$

5 : goto 2

6 : stop

този алгоритъм ще преработи в следната рекурсивна схема R:

$$
\begin{aligned}
R\colon\ & F_1(X, X) \quad \texttt{where} \\
& F_1(X, Y) \;=\; F_2(X, c) \\
& F_2(X, Y) \;=\; \texttt{if } \mathbf{p}(X) \texttt{ then } F_6(X, Y) \texttt{ else } F_3(X, Y) \\
& F_3(X, Y) \;=\; F_4(X, \mathbf{g}(X, Y)) \\
& F_4(X, Y) \;=\; F_5(\mathbf{f}(X), Y) \\
& F_5(X, Y) \;=\; F_2(X, Y) \\
& F_6(X, Y) \;=\; Y
\end{aligned}
$$

4.5.2 Теорема на Маккарти

Сега ще покажем, че класът на стандартните схеми се транслира в класа на рекурсивните схеми — резултат, известен като теорема на Маккрати.

Теорема 4.1. (Теорема на Маккарти.) Нека Σ е произволна сигнатура. За всяка стандартна схема S в сигнатурата Σ съществува еквивалентна на нея рекурсивна схема R в същата сигнатура, с други думи, такава, че:

$$Sem(S, \mathcal{A}) \;=\; Sem(R, \mathcal{A})$$

за всяка структура $\mathcal{A}$ в Σ.

Доказателство. Ще опишем алгоритъма, който на произволна стандартна схема съпоставя еквивалентна на нея рекурсивна схема. По същество ще следваме стъпките при преобразуването на стандартната схема от примера в предишния раздел.

Стандартната схема S, от която тръгваме, има следния общ вид:

$$
\begin{aligned}
S:\ & \texttt{input}(X_1,\dots,X_n);\ \texttt{output}(X_k)\\
& 1:\ O_1\\
& \vdots\\
& l:\ O_l\\
& \vdots\\
& q:\ O_q
\end{aligned}
$$

Нека паметта на S е $(X_1,\dots,X_m)$. Рекурсивната схема R, която ще конструираме, ще има q на брой функционални променливи $F_1,\dots,F_q$, като всички са на един и същ брой аргументи — точно m. Дефинициите на $F_1,\dots,F_q$ ще отразяват връзките между опашковите функции, които наблюдавахме, когато решавахме *Задача* 4.1. Идеята е във всяка структура $\mathcal{A}$ функцията, която се пресмята от F_l да е точно l-тата опашкова функция.

Да си спомним за връзката (4.3) между $Sem(S,\mathcal{A})$ и първата опашкова функция:

$$Sem(S,\mathcal{A})(x_1,\dots,x_n) \ \simeq\ g_1(x_1,\dots,x_n,\underbrace{x_n,...,x_n}_{m-n \text{ пъти}}).$$

Като имаме предвид това и факта, че първата опашкова функция съответства на F_1, полагаме главата на бъдещата схема R да е следният терм τ_0:

$$\tau_0(X_1,\dots,X_n,F_1) \ =\ F_1(X_1,\dots,X_n,\underbrace{X_n,...,X_n}_{m-n \text{ пъти}}). \tag{4.4}$$

Тялото на рекурсивната схема R е декларация от вида:

$$
\begin{aligned}
&F_1(X_1,\dots,X_m) = \tau_1(X_1,\dots,X_m,F_1,\dots,F_q)\\
&\vdots\\
&F_q(X_1,\dots,X_m) = \tau_q(X_1,\dots,X_m,F_1,\dots,F_q)
\end{aligned}
\tag{4.5}
$$

Всеки от горните термове τ_l определяме в зависимост от вида на l-тия оператор на схемата S. Разглеждаме поотделно случаите за O_l.

– ако O_l е $X_i := X_j$, то

$$\tau_l(X_1,\dots,X_m,F_1,\dots F_q) \ \stackrel{\text{деф}}{=}\ F_{l+1}(X_1,\dots,X_{i-1},X_j,X_{i+1},\dots,X_m);$$

– ако O_l е $X_i := \mathbf{c}_j$, то

$$\tau_l(X_1,\dots,X_m,F_1,\dots F_q) \ \stackrel{\text{деф}}{=}\ F_{l+1}(X_1,\dots,X_{i-1},\mathbf{c}_j,X_{i+1},\dots,X_m);$$

– ако O_l е $X_i := \mathbf{f}_j(X_{i_1}, \dots, X_{i_k})$, то

$$\tau_l(X_1, \dots, X_m, F_1, \dots F_q) \stackrel{\text{деф}}{=} F_{l+1}(X_1, \dots, X_{i-1}, \mathbf{f}_j(X_{i_1}, \dots, X_{i_k}), X_{i+1}, \dots, X_m);$$

– ако O_l е `goto` l', то

$$\tau_l(X_1, \dots, X_m, F_1, \dots F_q) \stackrel{\text{деф}}{=} F_{l'}(X_1, \dots, X_m);$$

– ако O_l е `if` $\mathbf{p}_j(X_{i_1}, \dots, X_{i_n})$ `then goto` l' `else goto` l'', то

$$\tau_l(X_1, \dots, X_m, F_1, \dots F_q) \stackrel{\text{деф}}{=} \texttt{if } \mathbf{p}_j(X_{i_1}, ..., X_{i_n}) \texttt{ then } F_{l'}(X_1, ..., X_m) \texttt{ else } F_{l''}(X_1, ..., X_m);$$

– ако O_l е `stop`, то

$$\tau_l(X_1, \dots, X_m, F_1, \dots F_q) \stackrel{\text{деф}}{=} X_k.$$

Идеята зад всяко от тези полагания е съвсем прозрачна. Например ако O_l е оператор за присвояване, отиваме на *следващия* оператор, при това с памет, изменена според това, което "казва" O_l. Ако O_l е оператор за преход или `stop`, смисълът на термовете τ_l е още по-ясен. Ясно е също, че новата схема R ще бъде в същата сигнатура като тази на S.

На лекции разказах неформално доказателството на теоремата на Маккарти. По-долу е формалното доказателство, в случай, че някой има желание да го прочете :).

Пристъпваме към доказателството на факта, че рекурсивната схема R, която конструирахме, е еквивалентна на S. За целта фиксираме произволна структура в сигнатурата Σ:

$$\mathcal{A} \ = \ (D;\ a_1, \dots, a_p;\ f_1, \dots, f_s;\ p_1, \dots, p_t).$$

Трябва да покажем, че

$$Sem(S, \mathcal{A}) \ = \ Sem(R, \mathcal{A}).$$

Да означим с $(R, \mathcal{A})$ рекурсивната *програма*, която се получава от схемата R в структурата $\mathcal{A}$. Нека още $g_1, \dots, g_q$ са опашковите функции на стандартната програма $(S, \mathcal{A})$. Ще покажем, че те са най-малко решение на системата, определена от тялото (4.5) на програмата $(R, \mathcal{A})$.

Но преди това да се убедим, че от този факт ще следва теоремата. По определение

$$Sem(R, \mathcal{A}) \ = \ D_V(R, \mathcal{A})$$

Като вземем предвид главата (4.4) на схемата R, за денотационната семантика $D_V(R, \mathcal{A})$ ще имаме:

$$D_V(R, \mathcal{A})(x_1, \ldots, x_n) \stackrel{\text{деф}}{\simeq} \tau_0(x_1, \ldots, x_n, g_1) \stackrel{(4.4)}{=} g_1(x_1, \ldots, x_n, \underbrace{x_n, ..., x_n}_{m-n \text{ пъти}}).$$

От друга страна, вече отбелязахме, че $Sem(S, \mathcal{A})$ е свързана с първата опашкова функция g_1 по следния начин:

$$Sem(S, \mathcal{A})(x_1, \ldots, x_n) \simeq g_1(x_1, \ldots, x_n, \underbrace{x_n, ..., x_n}_{m-n \text{ пъти}}).$$

От горните две равенства получаваме, че за всички $(x_1, \ldots, x_n) \in D^n$:

$$Sem(S, \mathcal{A})(x_1, \ldots, x_n) \simeq D_V(R, \mathcal{A})(x_1, \ldots, x_n),$$

и следователно $Sem(S, \mathcal{A}) = D_V(R, \mathcal{A}) \stackrel{\text{деф}}{=} Sem(R, \mathcal{A})$.

Сега се насочваме към доказателството на факта, че опашковите функции $g_1, \ldots, g_q$ са най-малко решение на системата, съответна на тялото на програмата $(R, \mathcal{A})$. Да означим с $f_1, \ldots, f_q$ неизвестните функции в тази система. Тогава тя ще изглежда така:

$$\left| \begin{array}{l} f_1 = \Gamma_{\tau_1}(f_1, \ldots, f_q) \\ \vdots \\ f_q = \Gamma_{\tau_q}(f_1, \ldots, f_q), \end{array} \right. \tag{4.6}$$

където всеки от термалните оператори $\Gamma_{\tau_1}, \ldots, \Gamma_{\tau_q}$ се определя с равенството

$$\Gamma_{\tau_i}(\bar{f})(\bar{x}) \stackrel{\text{деф}}{\simeq} \tau_i(\bar{x}, \bar{f}).$$

Да проверим, че опашковите функции $g_1, \ldots, g_q$ са решение на горната система означава да проверим, че за всяко $l \in \{1, \ldots, q\}$ и всяко $(x_1, \ldots, x_m) \in D^m$ е изпълнено:

$$g_l(x_1, \ldots, x_m) \simeq \tau_l(x_1, \ldots, x_m, g_1, \ldots, g_q).$$

Всички те са очевидни от съответните дефиниции на опашкова функция и на терма τ_l.

По-интересно е да видим защо опашковите функции $g_1, \ldots, g_q$ са *най-малкото решение* на системата (4.6). За целта нека $h_1, \ldots, h_q$ са друго решение на (4.6). Това означава, че за всяко $(x_1, \ldots, x_m) \in D^m$ са за тях е изпълнено:

$$\left| \begin{array}{l} h_1(x_1, \ldots, x_m) \simeq \tau_1(x_1, \ldots, x_m, h_1, \ldots, h_q) \\ \vdots \\ h_q(x_1, \ldots, x_m) \simeq \tau_q(x_1, \ldots, x_m, h_1, \ldots, h_q) \end{array} \right. \tag{4.7}$$

Трябва да покажем, че

$$\text{за всяко } l \in \{1, \dots, q\}: \quad g_l \subseteq h_l,$$

което по дефиниция означава

$$\forall l \in \{1, \dots, q\}\ \forall x_1 \dots \forall x_m \forall y\ (g_l(x_1, \dots, x_m) \simeq y \implies h_l(x_1, \dots, x_m) \simeq y).$$

Да препишем още веднъж това условие, като минем през дефиницията (4.2) на опашкова функция:

$$\forall l \forall \bar{x} \forall y\ (\exists t \exists \bar{y}\ \mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y})\ \&\ y = y_k \implies h_l(x_1, \dots, x_m) \simeq y). \quad (4.8)$$

Тук си спомняме за един логически закон, който ви е известен от ЛП ⌣̈:

$$\forall \bar{x}(\exists \bar{y} \varphi(\bar{x}, \bar{y}) \implies \psi(\bar{x}, \bar{y})) \iff \forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \implies \psi(\bar{x}, \bar{y})).$$

Прилагаме го към формулата (4.8) и получаваме

$$\forall l \forall \bar{x} \forall y \forall t \forall \bar{y}\ (\mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y})\ \&\ y = y_k \implies h_l(\bar{x}) \simeq y).$$

Разбира се, оттук можем да изключим променливата y_k. Ще получим

$$\forall l \forall \bar{x} \forall t \forall \bar{y}\ (\mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y}) \implies h_l(\bar{x}) \simeq y_k). \quad (4.9)$$

Понеже всички квантори в (4.9) са еднотипни, можем да ги разместваме на воля. Да ги препишем така:

$$\forall t\, \underbrace{\forall l \forall \bar{x} \forall \bar{y}\ (\mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y}) \implies h_l(x_1, \dots, x_m) \simeq y_k)}_{P(t)} \quad (4.10)$$

Да означим с P следното свойство над $\mathbb{N}$:

$$P(t) \overset{\text{деф}}{\iff} \forall l \forall \bar{x} \forall \bar{y}\ (\mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y}) \implies h_l(\bar{x}) \simeq y_k).$$

Ще докажем, че $\forall t P(t)$ като използваме обикновена индукция относно $t \in \mathbb{N}$, или другояче казано — индукция по броя стъпки, за които спира програмата $\mathit{Sem}(S, \mathcal{A})$, извикана от l-тия си оператор O_l върху вход $(x_1, \dots, x_m)$.

База $t = 0$. Нека за произволни $l \in \{1, \dots, q\}$, $\bar{x} \in D^m$ и $\bar{y} \in D^m$

$$\mathit{Step}^0(l, \bar{x}) \simeq (q, \bar{y}).$$

Но $\mathit{Step}^0(l, \bar{x}) \overset{\text{деф}}{=} (l, \bar{x})$, следователно $(l, \bar{x}) = (q, \bar{y})$, което означава, че $l = q$ и $\bar{x} = \bar{y}$. Значи O_l е операторът `stop`. Но тогава по дефиниция $\tau_q = X_k$, и q-тото уравнение на системата (4.6) ще е

$$F_q(X_1, \dots, X_m) \ = \ X_k.$$

Това означава, че за h_q ще е изпълнено:

$$h_q(x_1, \dots, x_m) = x_k.$$

Но по-горе видяхме, че всъщност $\bar{x} = \bar{y}$, в частност, $x_k = y_k$, откъдето получаваме търсеното $h_l(\bar{x}) = y_k$.

Нека сега за произволно t е в сила $P(t)$, т.е. вярно е, че

$$\forall l \forall \bar{x} \forall \bar{y}\ (\mathit{Step}^t(l, \bar{x}) \simeq (q, \bar{y}) \implies h_l(\bar{x}) \simeq y_k).$$

Искаме да покажем и $P(t{+}1)$, което разписано означава, че за фиксирани $l \in \{1, \dots, q\}$, $\bar{x} \in D^m$ и $\bar{y} \in D^m$ трябва да покажем импликацията

$$\mathit{Step}^{t+1}(l, \bar{x}) \simeq (q, \bar{y}) \implies h_l(\bar{x}) \simeq y_k.$$

Наистина, да приемем, че предпоставката $\mathit{Step}^{t+1}(l, \bar{x}) \simeq (q, \bar{y})$ е вярна. Разглеждаме различните възможности за оператора O_l.

Ако той е оператор за присвояване, ще се спрем само на един от трите случая, защото всички те се разглеждат аналогично. Нека например O_l е $X_i := X_j$. Тогава от дефиницията на функцията Step ще имаме, че

$$\mathit{Step}(l, \bar{x}) = (l + 1, x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m),$$

откъдето

$$\mathit{Step}^{t+1}(l, \bar{x}) \overset{\text{деф}}{\simeq} \mathit{Step}^t(\mathit{Step}(l, \bar{x})) \simeq \mathit{Step}^t(l{+}1, x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m) \simeq (q, \bar{y}).$$

Получихме, че

$$\mathit{Step}^t(l + 1, x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m) \simeq (q, \bar{y}),$$

което съгласно индукционното предположение $P(t)$, означава, че

$$h_{l+1}(x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m) \simeq y_k. \tag{4.11}$$

Да си спомним, че функциите $h_1, \dots, h_q$ удовлетворяват равенствата (4.7). В случая ни трябва l-тото от тях:

$$h_l(x_1, \dots, x_m) \simeq \tau_l(x_1, \dots, x_m, h_1, \dots, h_q). \tag{4.12}$$

Но колко е τ_l , когато операторът O_l е $X_i := X_j$? Поглеждаме към съответните дефиниции по-горе и виждаме, че в този случай то е

$$\tau_l(X_1, \dots, X_m, F_1, \dots F_q) \overset{\text{деф}}{=} F_{l+1}(X_1, \dots, X_{i-1}, X_j, X_{i+1}, \dots, X_m).$$

Следователно равенството (4.12) можем да препишем така:

$$h_l(x_1, \dots, x_m) \simeq h_{l+1}(x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m).$$

Но от (4.11) имаме, че $h_{l+1}(x_1, ..., x_{i-1}, x_j, x_{i+1}, ..., x_m) \simeq y_k$, откъдето получаваме търсеното

$$h_l(x_1, \ldots, x_m) \simeq y_k.$$

Да разгледаме и случая, в който O_l е оператор за преход. Нека той е `goto` l'. Тогава по дефиниция

$$Step(l, \bar{x}) = (l', \bar{x}),$$

откъдето

$$Step^{t+1}(l, \bar{x}) \stackrel{\text{деф}}{\simeq} Step^t(Step(l, \bar{x})) \simeq Step^t(l', \bar{x}) \simeq (q, \bar{y}).$$

Получихме, че

$$Step^t(l', \bar{x}) \ \simeq \ (q, \bar{y}),$$

което съгласно индукционното предположение $P(t)$, означава, че

$$h_{l'}(\bar{x}) \simeq y_k. \tag{4.13}$$

Отново от това, че $h_1, \ldots, h_q$ удовлетворяват равенствата (4.7) ще имаме

$$h_l(x_1, \ldots, x_m) \simeq \tau_l(x_1, \ldots, x_m, h_1, \ldots, h_q). \tag{4.14}$$

Остана да видим как сме дефинирали τ_l в този случай:

$$\tau_l(X_1, \ldots, X_m, F_1, \ldots F_q) \ \stackrel{\text{деф}}{=} \ F_{l'}(X_1, \ldots, X_m).$$

В такъв случай

$$\tau_l(x_1, \ldots, x_m, h_1, \ldots, h_q) \simeq h_{l'}(x_1, \ldots, x_m).$$

Сега отново комбинираме това равенство с горните (4.13) и (4.14), за да получим, че

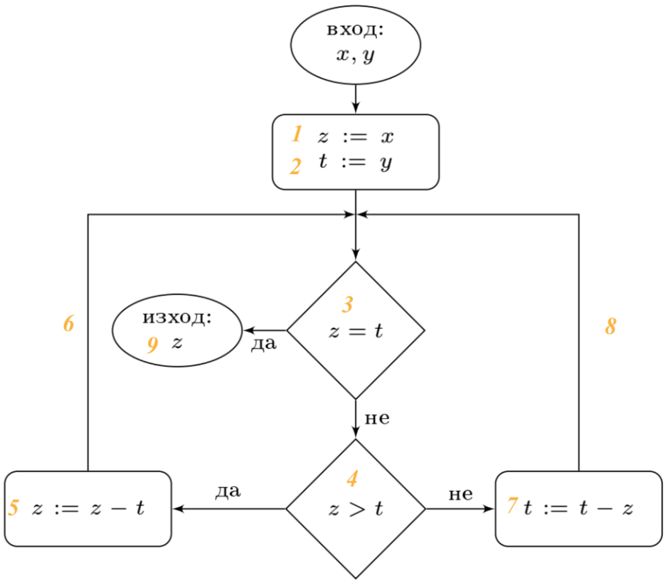
$$h_l(x_1, \ldots, x_m) \simeq y_k.$$

$\square$

От доказателството, което проведохме, се вижда, че преводът на всяка стандартна схема в еквивалентната на нея рекурсивна е алгоритмичен. С други думи, не просто е вярно, че за всяка стандартна схема S съществува рекурсивна схема R, еквивалентна на нея, но имаме *общ метод* по схемата S *да конструираме* резултантната схема R. Този метод се нарича *метод на опашковите функции* и се използва за намиране на ефективни рекурсивни програми, тъй като рекурсията в R е изцяло *линейна*.

Задача 4.2. Като приложите метода на опашковите функции, намерете рекурсивната схема, еквивалентна на следната блок-схема:

Задача 4.2. Като приложите метода на опашковите функции, намерете рекурсивната схема, еквивалентна на следната блок-схема:



Решение. Първо преписваме блок-схемата като стандартна програма:

$\texttt{input}(X,Y);\ \texttt{output}(Z)$

$1:\ Z := X$

$2:\ T := Y$

$3:\ \texttt{if}\ Z = T\ \texttt{then goto}\ 9\ \texttt{else goto}\ 4$

$4:\ \texttt{if}\ Z > T\ \texttt{then goto}\ 5\ \texttt{else goto}\ 7$

$5:\ Z := Z - T$

$6:\ \texttt{goto}\ 3$

$7:\ T := T - Z$

$8:\ \texttt{goto}\ 3$

$9:\ \texttt{stop}$

Следвайки описания по-горе алгоритъм, от S получаваме следната рекурсивна програма R:

$$
\begin{array}{lll}
R\colon F_1(X,Y,Y,Y) & \texttt{where} & \\
\quad F_1(X,Y,Z,T) & = & F_2(X,Y,X,T) \\
\quad F_2(X,Y,Z,T) & = & F_3(X,Y,Z,Y) \\
\quad F_3(X,Y,Z,T) & = & \texttt{if}\ Z = T\ \texttt{then}\ F_9(X,Y,Z,T)\ \texttt{else}\ F_4(X,Y,Z,T) \\
\quad F_4(X,Y,Z,T) & = & \texttt{if}\ Z > T\ \texttt{then}\ F_5(X,Y,Z,T)\ \texttt{else}\ F_7(X,Y,Z,T) \\
\quad F_5(X,Y,Z,T) & = & F_6(X,Y,Z-T,T) \\
\quad F_6(X,Y,Z,T) & = & F_3(X,Y,Z,T) \\
\quad F_7(X,Y,Z,T) & = & F_8(X,Y,Z,T-Z) \\
\quad F_8(X,Y,Z,T) & = & F_3(X,Y,Z,T) \\
\quad F_9(X,Y,Z,T) & = & Z
\end{array}
$$

След няколко тривиални опростявания достигаме до

$$
\begin{aligned}
R\colon\ & F_1(X,Y,Y,Y) \quad \texttt{where} \\
& F_1(X,Y,Z,T) = F_3(X,Y,X,Y) \\
& F_3(X,Y,Z,T) = \texttt{if } Z = T \texttt{ then } Z \\
& \qquad\qquad \texttt{else if } Z > T \texttt{ then } F_3(X,Y,Z-T,T) \\
& \qquad\qquad\qquad \texttt{else } F_3(X,Y,Z,T-Z)
\end{aligned}
$$

Всъщност F_3 не зависи от X и Y, затова я преписваме като

$$
\begin{aligned}
F_3(Z,T) &= \texttt{if } Z = T \texttt{ then } Z \\
&\quad \texttt{else if } Z > T \texttt{ then } F_3(Z-T,T) \texttt{ else } F_3(Z,T-Z)
\end{aligned}
$$

или все едно

$$
\begin{aligned}
F_3(X,Y) &= \texttt{if } X = Y \texttt{ then } X \\
&\quad \texttt{else if } X > Y \texttt{ then } F_3(X-Y,Y) \texttt{ else } F_3(X,Y-X)
\end{aligned}
$$

Тогава първото уравнение става

$$F_1(X,Y,Z,T) = F_3(X,Y)$$

Сега изключваме Z и T и от F_1, елиминираме F_3, защото става излишна, и получаваме добре познатата ни рекурсивна програма

$$
\begin{aligned}
R\colon\ & F(X,Y) \quad \texttt{where} \\
& F(X,Y) = \texttt{if } X = Y \texttt{ then } X \\
& \qquad \texttt{else if } X > Y \texttt{ then } F(X-Y,Y) \texttt{ else } F(X,Y-X)
\end{aligned}
$$

$\square$

Обратната посока на Теоремата на Маккарти не е вярна, т.е. съществува рекурсивна схема, която не е еквивалентна на никоя стандартна схема. Това означава, че класът на рекурсивните схеми не е транслируем в класа на стандартните схеми ($\mathcal{R} \not\leq_t \mathcal{S}$), с други думи, рекурсията е по-мощна от итерацията.

Теорема 4.2. (Пример на Патерсън и Хюит.) Да означим с R следната рекурсивна схема:

$F(X) \quad \texttt{where}$

$F(X) = \texttt{if } \mathbf{p}(X) \texttt{ then } X \texttt{ else } \mathbf{f}(F(\mathbf{g}(X)),\ F(\mathbf{h}(X)))$

За тази схема R не съществува стандартна схема S, такава че R е еквивалентна на S.

Слагам теоремата на Патерсън и Хюит само за сведение. За съжаление, на последната лекция не остана време за нея.