СХЕМЫ ПРОГРАММ

Программы и схемы программ

Схемы программ – это математические модели программ, описывающие строение программы, или точнее строение множества программ, где конкретные операции и функции заменены абстрактными функциональными и предикатными символами. Следующий пример программ вычисления факториала *n*! и переворачивания слов поясняет различие между программами и их схемой S_1.

begin integer x,y;
ввод(x);
y:=1;
L:if x=0 then
goto L1;
y:=x*y;
x:=x-1;
goto L;
L1:вывод(y);
end
	begin integer x,y;
	ввод(x);
	y:=ε;
	L:if x=0 then
	goto L1;
	y:=CONSCAR(x,y);
	x:=COR(x);
	goto L;
	L1:вывод(y);
	end
		begin
		ввод(x);
		y:=a;
		L:if p(x) then
		goto L1;
		y:=g(x,y);
		x:=h(x);
		goto L;
		L1:вывод(y);
		end

Функция CONSCAR (суперпозиция функций CONS и CAR из языка Лисп) приписывает первую букву первого слова ко второму слову (т. е. CONSCAR (аб, в)=ав), а функция CAR стирает первую букву слова (т. е. CAR (аб)=б).

Базис класса стандартных схем программ

Стандартные схемы программ (ССП) характеризуются базисом и структурой схемы. Базис класса фиксирует символы, из которых строятся схемы, указывает их роль (переменные, функциональные символы и др.), задает вид выражений и операторов схем.

Полный *базис B класса стандартных схем* состоит из 4-х непересекающихся, счетных множеств символов и множества операторов – слов, построенных из этих символов.

Множества символов полного базиса:

1) $X = \{x, x_1, x_2, \dots, y, y_1, y_2, \dots, z, z_1, z_2, \dots\}$ - множество символов, называемых *переменными*;

2) $F = \{f^{(0)}, f^{(1)}, f^{(2)}, \dots, g^{(0)}, g^{(1)}, g^{(2)}, \dots, h^{(0)}, h^{(1)}, h^{(2)}, \dots\}$ - множество *функциональных символов*; верхний символ задает *местность символа*; нульместные символы называют константами и обозначают начальными буквами латинского алфавита *a*, *b*, *c*,...;

3) $P = \{p^{(0)}, p^{(1)}, p^{(2)}, \dots, q^{(0)}, q^{(1)}, q^{(2)}, \dots\}$ - множество *предикатных символов*; $p^{(0)}, q^{(0)}$ - нульместные символы называют логическими константами;

4) $\{\mathbf{start}, \mathbf{stop}, \dots, :=, \text{и т. д.}\}$ - множество специальных символов.

Термами (*функциональными выражениями*) называются слова, построенные из переменных, функциональных и специальных символов по следующим правилам:

1) односимвольные слова, состоящие из переменных или констант, являются термами;

2) слово τ вида $f^{(n)}(\tau_1, \tau_2, \dots, \tau_n)$, где $\tau_1, \tau_2, \dots, \tau_n$ - термы, является термом;

3) те и только те слова, о которых говорится в ССП являются термами.

Примеры термов: $x, f^{(0)}, a, f^{(1)}(x), g^{(2)}\left(x, h^{(2)}(y, a)\right)$.

Тестами (логическими выражениями) называются логические константы и слова вида $p^{(n)}(\tau_1, \tau_2, \dots, \tau_n)$.

Примеры тестов: $p^{(0)}, p^{(1)}(x), g^{(3)}(x, y, z), p^{(2)}\left(f^{(2)}(x, y), z\right)$.

Допускается в функциональных и логических выражениях опускать индексы местности, если это не приводит к двусмысленности или противоречию.

Множество операторов включает пять типов:

1) *начальный оператор* - слово вида $\mathbf{start}(x_1, x_2, \ldots, x_k)$, где $k \geq 0$, а $x_1, x_2, \ldots, x_k$ - переменные, называемые результатом этого оператора;

2) *заключительный оператор* - слово вида $\mathbf{stop}(\tau_1, \tau_2, \ldots, \tau_n)$, где $n \geq 0$, а $\tau_1, \tau_2, \ldots, \tau_n$ - термы; вхождения переменных в термы τ называются *аргументами* этого *оператора*;

3) *оператор присваивания* - слово вида $x := \tau$, где x – переменная (*результат оператора*), а τ – терм; вхождения переменных в термы называются *аргументами* этого *оператора*;

4) *условный оператор* (тест) – логическое выражение; вхождения переменных в логическое выражение называются *аргументами* этого *оператора*;

5) *оператор петли* - односимвольное слово loop.

Среди операторов присваивания выделим случаи: когда τ – переменная, то оператор называется *пересылкой* $(x := y)$ и когда τ – константа, то оператор называется *засылкой* $(x := a)$.

Подклассы используют ограниченные базисы. Так, например, подкласс Y_1 имеет базис:

$\{x_1, x_2\}$, $\{a, f^{(1)}\}$, $\{p^{(1)}\}$, $\{\mathbf{start}, \mathbf{stop}, (,), :=, ,\}$ и множество операторов $\{\mathbf{start}(x_1, x_2);\ x_1 := f(x_1);\ x_2 := f(x_2);\ x_1 := a;\ x_2 := a;\ p(x_1);\ p(x_2);\ \mathbf{stop}(x_1, x_2);\}$, т. е. схемы из этого подкласса используют две

переменные, константу *a*, один одноместный функциональный символ, один предикатный символ и операторы указанного вида.

Графовая форма стандартной схемы.

Представим стандартную схему программ как размеченный граф, вершинам которого приписаны операторы из некоторого базиса *B*.

Стандартной схемой в базисе *B* называется конечный (размеченный ориентированный) граф без свободных дуг и с вершинами следующих пяти видов:

1) *Начальная вершина* (ровно одна) помечена начальным оператором. Из нее выходит ровно одна дуга. Нет дуг, ведущих к начальной вершине.

2) *Заключительная вершина* (может быть несколько). Помечена заключительным оператором. Из нее не выходит ни одной дуги.

3) *Вершина-преобразователь.* Помечена оператором присваивания. Из нее выходит ровно одна дуга.

4) *Вершина-распознаватель.* Помечена условным оператором (называемым условием данной вершины). Из нее выходит ровно две дуги, помеченные 1 (левая) и 0 (правая).

5) *Вершина-петля.* Помечена оператором петли. Из нее не выходит ни одной дуги.

Конечное множество переменных схемы *S* составляют ее память X_S.

Из определения следует, что один и тот же оператор может помечать несколько вершин схемы. Вершины именуются (метки вершины) целым неотрицательным числом (0, 1, 2,...). Начальная вершина всегда помечается меткой 0.

Схема *S* называется правильной, если на каждой дуге заданы все переменные.

Пример правильной ССП S_1 в графовой форме приведен на рис. 5.1, а.

Вершины изображены прямоугольниками, а вершина-распознаватель - овалом. Операторы записаны внутри вершины.

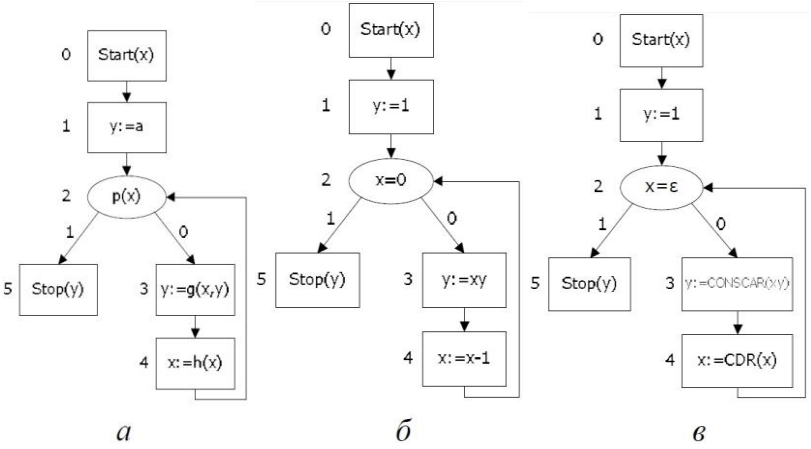

Рис. 5.1. Стандартная схема программы

Интерпретация стандартных схем программ

ССП не является записью алгоритма, поэтому позволяет исследовать только структурные свойства программ, но не семантику вычислений. При построении «семантической» теории схем программ вводится понятие интерпретация ССП. Определим это понятие.

Пусть в некотором базисе *B* определен класс ССП. *Интерпретацией базиса B в области интерпретации D* называется функция *I*, которая сопоставляет:

1) каждой переменной *x* из базиса *B* - некоторый элемент $d = I(x)$ из области интерпретации *D*;

2) каждой константе *a* из базиса *B* - некоторый элемент $d = I(a)$ из области интерпретации *D*;

3) каждому функциональному символу $f^{(n)}$ - всюду определенную функцию $F = I\left(f^{(n)}\right)$;

4) каждой логической константе $p^{(0)}$ - один символ множества {0, 1};

5) каждому предикатному символу $p^{(n)}$ - всюду определенный предикат $P = I\left(p^{(n)}\right)$.

Пара (*S, I*) называется *интерпретированной стандартной схемой* (ИСС), или *стандартной программой* (СП).

Определим понятие *выполнения программы*.

Состоянием памяти программы (*S, I*) называют функцию: *W:* $X_s \to D$, которая каждой переменной *x* из памяти схемы *S* сопоставляет элемент *W*(*x*) из области интерпретации *D*.

Значение терма τ при интерпретации *I* и состоянии памяти *W* (обозначим $\tau_I(W)$) определяется следующим образом:

1) если $\tau = x$, *x* - переменная, то $\tau_I(W) = I(x)$;

2) если $\tau = a$, *a* - константа, то $\tau_I(W) = I(a)$;

3) если $\tau = f^{(n)}(\tau_1, \tau_2, \dots, \tau_n)$, то $\tau_I(W) = I\left(f^{(n)}\left(\tau_{1I}(W), \tau_{2I}(W), \dots, \tau_{nI}(W)\right)\right)$.

Аналогично определяется значение теста π при интерпретации *I* и состоянии памяти *W* или $\pi_I(W)$:

4) если $\pi = p^{(n)}(\tau_1, \tau_2, \dots, \tau_n)$, то $\pi_I(W) = I\left(p^{(n)}\left(\tau_{1I}(W), \tau_{2I}(W), \dots, \tau_{nI}(W)\right)\right)$, $n \geq 0$.

Конфигурацией программы называют пару $U = (L, W)$, где *L* - метка вершины схемы *S*, а *W* - состояние ее памяти. Выполнение программы описывается конечной или бесконечной последовательностей конфигураций, которую называют протоколом выполнения программы (ПВП).

Протокол $(U_0, U_1, \dots, U_i, U_{i+1}, \dots)$ выполнения программы (S, I) определяем следующим образом (ниже k_i означает метку вершины, а W_i - состояние памяти в *i*-ой конфигурации протокола, $U_i = (k_i, W_i)$):

$U_0 = (0, W_0)$ – начальное состояние памяти схемы *S* при интерпретации *I*.

Пусть $U_i = (k_i, W_i)$ - *i*-я конфигурация ПВП, а *O* - оператор схемы *S* в вершине с меткой k_i. Если *O* - заключительный оператор $\mathbf{stop}(\tau_1, \dots, \tau_m)$, то U_i - последняя конфигурация, так что протокол конечен. В этом случае считают, что, программа (S, I) *останавливается*, а последовательность значений $\tau_{1I}(W), \tau_{2I}(W), \dots, \tau_{nI}(W)$ объявляют *результатом* $val(S, I)$ *выполнения программы* (S, I). В противном случае, т. е. когда *O* - не заключительный оператор, в протоколе имеется следующая, (*i*+1)-я конфигурация $U_{i+1} = (k_{i+1}, W_{i+1})$, причем

1) если *O* - начальный оператор, а выходящая из него дуга ведет к вершине с меткой *L*, то $k_{i+1} = L$ и $W_{i+1} = W_i$;

2) если *O* - оператор присваивания $x := \tau$, а выходящая из него дуга ведет к вершине с меткой *L*, то $k_{i+1} = L$, $W_{i+1} = W_i, W_{i+1}(x) = \tau_I(W_i)$;

3) если *O* - условный оператор π и $\pi_I(W_i) = \Delta$, где $\Delta \in \{0, 1\}$, а выходящая из него дуга ведет к вершине с меткой *L*, то $k_{i+1} = L$ и $W_{i+1} = W_i$;

4) если *O* - оператор петли, то $k_{i+1} = L$ и $W_{i+1} = W_i$ так что протокол бесконечен.

Таким образом, программа останавливается тогда и только тогда, когда протокол ее выполнения конечен. В противном случае программа *зацикливается* и результат ее выполнения не определен.

Рассмотрим две интерпретации ССП S_1 (рис. 5.1). Интерпретация (S_1, I_1) задана так:

1) область интерпретации $D_1 \subseteq Nat$ - подмножество множества *Nat* целых неотрицательных чисел;

2) $I_1(x) = 4; I_1(y) = 0; I_1(a) = 1;$

3) $I_1(g) = G$, где *G* - функция умножения чисел, т.е. $G(d_1, d_2) = d_1 \cdot d_2;$

4) $I_1(h) = H$, где *H* - функция вычитания единицы, т.е. $H(d) = d - 1;$

5) $I_1(p) = P_1$, где P_1 - предикат «равно 0», т.е. $P_1(d) = 1$, если $d = 0$.

Программа (S_1, I_1) вычисляет 4! (рис. 5.1, *б*).

Интерпретация (S_1, I_2) задана следующим образом:

1) область интерпретации $D_2 = V^*$, где $V = \{a, b, c\}$, V^* – множество всех возможных слов в алфавите V;

2) $I_2(x) = abc;$

3) $I_2(y) = \varepsilon$, где ε – пустое слово;

4) $I_2(a) = \varepsilon;$

5) $I_2(g) = \text{CONSTAR};$

6) $I_2(h) = \text{CDR};$

7) $I_2(p) = P_2$, где P_2 - предикат «равное ε», т.е. $P_2(\alpha) = 1$, если $\alpha = \varepsilon$.

Программа (S_1, I_2) преобразует слово *abc* в слово *cba* (рис. 5.1, *в*). ПВП (S_1, I_1) и (S_1, I_2) конечен, результат - 24 и - *cba* (табл. 1 и 2).

Таблица 1

Протокол выполнения программы (S_1, I_1)

Конфигурация		U_0	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}	U_{13}
Метка		0	1	2	3	4	2	3	4	2	3	4	2	3	
Значения	*x*	4	4	4	4	3	3	3	2	2	2	1	1	1	0
	y	0	1	1	4	4	4	12	12	12	24	24	24	24	24

Таблица 2

Протокол выполнения программы (S_1, I_2)

Конфигурация		U_0	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}
Метка		0	1	2	3	4	2	3	4	2	3	4	2	5
Значения	*x*	*abc*	*abc*	*abc*	*abc*	*bc*	*bc*	*bc*	*c*	*c*	*c*	*ε*	*ε*	*ε*
	y	*ε*	*ε*	*ε*	*a*	*a*	*a*	*ba*	*ba*	*ba*	*cba*	*cba*	*cba*	*cba*