ЛОГИЧЕСКИЕ МЕТОДЫ В ПРОГРАММИРОВАНИИ
(вычислительные системы)
1987 год Выпуск 120

УДК 519.68

Σ^+-ПРОГРАММЫ И ИХ СЕМАНТИКИ

С.С.Гончаров, Д.И.Свириденко

Введение

В [1-15] были изложены математические основы новой логической концепции описания вычислений - семантического программирования. Следующий естественный шаг - разработка языка семантического программирования. Однако написание транслятора для этого языка требует, помимо указания денотационной семантики, описания и того, как должны использоваться семантические программы, или, другими словами, процедурной семантики языка. В свою очередь, чтобы точно определить денотационную и процедурную семантики, необходимо предварительно построить и изучить формальную модель семантической программы. В этом и заключается цель настоящей работы.

Семантическая программа - это специального вида логико-математическое описание задачи, допускающее эффективное "исполнение" в виде проверки его истинности. Подобное описание (или, как еще говорят, спецификация) в семантическом программировании задается парой (S,q), где S - логико-математическое представление (называемое в дальнейшем схемой) в языке исчисления предикатов I-го порядка той проблемной области, к которой относится решаемая задача, а q - цель задачи (запрос) вида $?\bar{y}.\varphi(\bar{y})$, интерпретируемая как указание "найти все те $\bar{y}$, для которых в модели для S истинно утверждение φ".Заметим, что если набор $\bar{y}$ пуст, то обработка запроса сводится просто к проверке истинности формулы φ.

Характерной особенностью семантических программ является модульность их схем, что естественно предполагает наличие в языке развитого механизма параметризации. В том варианте языка семантического программирования*), для которого в данной работе будет

*) В одном из следующих выпусков "Вычислительные системы" будет помещено концептуальное описание этого варианта - языка СИГМА.

строиться модель программ, различают три вида модулей - базисные (B-модули), дефинициальные (D-модули) и аксиоматические (A-моду - ли). B-модуль - это описание того набора исходных (встроенных или, как мы их будем называть, базисных) операций и отношений, в терми- нах которых в конечном счете будет исполняться данная семантиче - ская программа. Таким образом, B-модуль - это как-бы своеобраз - ный "вычислитель". С логико-математической точки зрения B-модуль является многосортной конструктивной (в общем случае даже позитив- ной) моделью. Заметим, что B-модуль может быть параметрическим, где в качестве параметров выступают индивидные, функциональные и предикатные переменные. Параметрами могут быть и модульные пере- менные. Такая параметризация позволяет связывать B-модули в слож- ные структуры, которые можно рассматривать как своеобразные базы данных.

Далее, D-модули - это параметрическое описание функций и от- ношений в виде системы рекурсивных определений, где в качестве определяющих конструкций выступают Σ^+-формулы [5,I5]. И, наконец, A-модули - это аксиоматическое описание функций и отношений. Та- ким образом, в данном варианте языка предпринята попытка синтеза двух хорошо известных в математической логике механизмов формаль- ного описания моделей - элементарной определимости одной модели в другой и аксиоматического описания модели. Поскольку A-модули яв- ляются Σ^+-определимыми (см. [4,I0], а также §4 настоящей работы), то, следовательно, на A-модуль можно смотреть как на частный случай D-модуля. Поэтому естественно в качестве объекта модели - рования рассматривать параметрический D-модуль. В роли его фор - мальной модели в настоящей работе выступает понятие Σ^+-програм - мы - конечной совокупности рекурсивных определений отношений и функций вида 'R опр φ' (R - определяемое отношение (функция), φ - определяющая это отношение Σ^+-формула) и запрос вида $?\bar{y}.\phi(\bar{y})$, где ϕ - Σ^+-формула.

В основе определения денотационной семантики Σ^+-программ, ко- торая согласно методологическим установкам концепции должна быть первичной по отношению к процедурной, лежит аналог теоремы Ганди, позволяющей решать рекурсивные Σ^+-определения [5,7,I0,I5]. По- скольку подобные решения ищутся эффективно, то соответствующий ал- горитм поиска решений можно положить в основу процедурной семан - тики Σ^+-программ, которую мы назовем условно компиляционной. При

компиляционном исполнении Σ^+-программ вначале ищется решение системы рекурсивных определений, составляющих содержание ее схемы, а затем, на основе полученных явных Σ^+-определений предикатов и функций схемы обрабатывается запрос. Легко понять, что такая схема исполнения наиболее целесообразна в тех ситуациях, когда решается широкий класс задач, в спецификациях которых присутствует одна и та же схема.

Другой вид процедурной семантики, которую можно назвать интерпретационной, с самого начала отталкивается от запроса и использует определения схемы программы по мере необходимости. Между этими разновидностями семантики много общего. Так, например, если схема программы не рекурсивна, то обе процедурные семантики для нее просто совпадают. Это обстоятельство и дает возможность определить единую схему процедурной семантики Σ^+-программ.

Основным понятием при описании процедурной семантики будет понятие стратегии, являющееся своеобразной математической абстракцией процесса пошагового исполнения Σ^+-программ, что позволяет объективно выделять и изучать различные свойства денотатов используемых семантических конструкций, имея в виду прежде всего такие их характеристики, как детерминированность и принципиальная недетерминированность, означающая принципиальный параллелизм исполнения Σ-программ. Мы делали ударение на "принципиальный", поскольку хотим указать на различие таких явлений, как неизбежность (принципиальность) и возможность параллельного исполнения. Отметим, что наличие неизбежного параллелизма - это один из центральных моментов концепции семантического программирования, последствия которого, касающиеся, скажем, сложности вычислений, еще практически не изучены. Присутствие в языке принципиально параллельных конструкций существенно увеличивает его выразительную силу. В этом смысле ориентация на детерминированные, последовательные стратегии исполнения, что неизбежно при использовании большей части современных вычислительных машин, есть сознательное ограничение выразительной мощи языка (зачастую существенно усложняющее его денотационную семантику). Данное замечание носит, конечно, методологический характер, и его можно на данном этапе принимать во внимание или нет, но вот превратить это замечание в точный математический факт - серьезная проблема. Отметим только, что разработка формализмов, которые дают лишь только возможность классификации денотатов на детерминированные и принципиально недетермини-

рованные, само по себе уже является математическим результатом.

Стратегии исполнения Σ^+-программ, являясь формальными и в то же время эффективными конструкциями, сами допускают их представление в виде Σ^+-определимых функций, действующих на "синтаксических" списках, т.е. списках, построенных из термов, Σ^+-формул, Σ^+-схем и т.п. Поскольку речь идет об исполнении программ, то естественно желание иметь дело с "быстрыми" стратегиями. Выбор же или построение таковых, с одной стороны, определяется решаемой задачей (классом задач), а с другой - набором простейших, базисных "быстрых" стратегий и способами конструирования на них более сложных. Последнее можно мыслить себе как отдельный В-модуль СТРАТЕГ, что позволяет нужный набор базисных стратегий и конструкторов оформлять в виде специального В-модуля - фрагмента модуля СТРАТЕГ.

Процедурная семантика Σ^+-программ в настоящей работе будет определена в виде некой универсальной схемы их исполнения. Однако в эту схему будет "встроена" возможность ее локальной конкретизации, цель которой - оптимизация исполнения конкретной программы (класса программ). Таким образом, в концепции семантического программирования оптимизация программ может осуществляться двумя способами - оптимизационными преобразованиями самой программы (эта проблема требует своего самостоятельного решения) и оптимизационной надстройкой интерпретатора посредством конкретизации универсальной схемы исполнения наиболее подходящими для данной программы (или целого класса программ) "быстрыми" функциями и отношениями (проблема, требующая также дополнительных исследований). И здесь хотелось бы сделать еще одно методологическое замечание. В-модуль СТРАТЕГ (как и вообще любой В-модуль) представляет собой частичную многоосновную структуру. Но эта структура не должна рассматриваться как нечто постоянное и завершенное: она является модифицируемой, т.е. по мере накопления опыта решения задач и/или по теоретическим соображениям может со временем меняться набор сортов, операций и отношений этого модуля. В настоящей работе мы будем считать, что начальный набор сортов В-модуля СТРАТЕГ включает в себя сорта бул, сорт, сигнатура, терм, формула, схема, программа, нат и наслед-кон-список, т.е. сорта булевых значений, сортов, термов, формул, Σ^+-схем, Σ^+-программ, натуральных чисел и наследственно конечных списков (как объединяющей конструкции, в

которой все другие, перечисленные здесь, представляются стандартным образом). Для построения подмоделей сорта сорт и сигнатура можно воспользоваться опытом работы [I8]. Для сортов бул, терм, формула необходимо предусмотреть такие операции, как подстановка (термов и подформул), унификация (односторонняя и двусторонняя), дизъюнкции, конъюнкции, отрицания, навешивание кванторов, приведение к нормальным формулам и всевозможные другие эквивалентные преобразования формул, распознавание того, является ли формула алфавитным вариантом другой и т.п.

Структура работы следующая. В §I дается определение Σ^+-программы. В §2 строится денотационная семантика Σ^+-программ. В §3 описывается процедурная семантика Σ^+-программ, показываются ее корректность и полнота относительно денотационной, а в §4 изучается Σ^+-определимость параметрических логических программ в виде Σ^+-программ. И, наконец, в заключении указывается на возможность различных расширений понятия Σ^+-программа.

§I. Σ^+-программы

Пусть $\langle I, \sqsubseteq \rangle$ – некоторое частичное упорядоченное множество, элементы которого будем называть сортами; если $s_1, s_2 \in I$ и $s_1 \sqsubseteq s_2$, то будем говорить, что s_1 – подсорт сорта s_2. Последовательность $K_1, \ldots, K_n$, где $K_i \subseteq I$, $i = 1, \ldots, n$, называется типом над I. Между типами естественно определяется отношение предпорядка $\preccurlyeq$:

$$K_1, \ldots, K_n \preccurlyeq L_1, \ldots, L_m \Leftrightarrow (n = m) \wedge \forall i \le n \, \forall s \in K_i \, \exists s' \in L_i (s \sqsubseteq s').$$

Если имеет место $T_1 \preccurlyeq T_2$, то будем говорить, что тип T_1 есть подтип типа T_2.

Пусть σ_0 – сигнатура (в дальнейшем называемая базовой) с множеством сортов I. Предполагается, что каждому сигнатурному символу приписан тип над I. В дальнейшем, если a есть сигнатурный символ типа T (или другая конструкция, которой приписан тип), то будем писать $a: T$. Если a – n-местный функциональный символ типа $K_1, \ldots, K_n, K$, то будем также писать $a: K_1, \ldots, K_n \to K$. Кроме того, если $K = \{s\}$, $s \in I$, то вместо $a: K$ будем писать $a: s$.

Обозначим через σ сигнатуру с множеством сортов $I \cup \{list\}$, являющуюся обогащением сигнатуры σ_0 списочными функциями и предикатами (вместе с именем указывается и тип):

nil: list; head: list → I ∪ { list} ;

tail: list → list; cons: list, I ∪ { list} → list;

tc: list → list; ∈: I∪{ list}, list ;

⊆: list,list; At: I; List: list и т.п.

Помимо сигнатур σ_0 и σ, введем в рассмотрение сигнатуру σ^*, которая является обогащением σ множеством $\mathbb{P} = \{P, \ldots, Q, \ldots\}$ предикатных переменных и множеством $\mathbb{F} = \{F, \ldots G, \ldots\}$ функциональных переменных ($\mathbb{P} \cap \mathbb{F} \neq 0$), предполагая, что каждому типу над $I \cup \{\text{list}\}$ соответствует бесконечное подмножество предикатных и функциональных переменных. Множеством сортов сигнатуры σ^* будет множество $I(\sigma^*) = I \cup \{ \text{list}\}$. Далее будем считать, что для каждого типа над $I(\sigma^*)$ вида K имеется бесконечное множество (индивидных) переменных X (K) этого типа.

Множество объектных термов (или σ-термов) $T(\sigma)$ определяется обычным образом:

I) переменные и константы типов K, $K \subseteq I \cup \{ \text{list} \}$, являются σ-термами этих же типов;

2) если $f \in \sigma$, тип $(f) = K_1, \ldots, K_n \to K$, а $t_1(:K_1), \ldots, t_n(:K_n)$ суть σ-термы, то $f(t_1, \ldots, t_n)(:K)$ - σ-терм;

3) других σ-термов нет.

Определим множество формул сигнатуры σ^*.

I) Если t_1, t_2 - σ-термы одного и того же типа T, то $t_1 =_T t_2$ - простая формула.

ЗАМЕЧАНИЕ. Предполагается, что с каждым сортом и, следовательно, с каждым типом T над $I \cup \{\text{list}\}$ связано отношение равенства $=_T$, согласованное при своей интерпретации с отношениями "быть подсортом" и "быть подтипом".

2) Если $P \in \sigma^*$ - n-местный предикатный символ типа $K_1, \ldots, K_n$ над $I \cup \{\text{list}\}$, а $t_1 : K_1, \ldots, t_n : K_n$ - σ-термы, то $P(t_1, \ldots, t_n)$ - формула.

3) Если $F \in \sigma^* \setminus \sigma$ - n-местная функциональная переменная типа $K_1, \ldots, K_n \to K$ над $I \cup \{\text{list}\}$ и $t_1 : K_1, \ldots, t_n : K_n, t : K$ - σ-термы, то $F(t_1, \ldots, t_n) = t$ - формула.

4) Если φ и ψ - формулы, то $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \supset \psi)$, $\neg\varphi$, $\forall x \varphi$ и $\exists x \varphi$ - также формулы; здесь x - переменная типа K над $I \cup \{\text{list}\}$.

5) Если φ - формула, t: list - σ-терм, x: $I \cup \{\text{list}\}$ - переменная, то $\forall x \in t . \varphi$ и $\exists x \in t . \varphi$ - формулы.

6) Если φ - формула, t: list - σ-терм, x: list - переменная, то $\forall x \subseteq t . \varphi$, $\exists x \subseteq t . \varphi$ - формулы.

7) Других формул нет.

В п п.5 и 6 кванторы носят название ограниченных. Далее, если φ - формула, а $P(\bar{t})$ или $F(\bar{t}) = t$ $(P, F \in \sigma^* \setminus \sigma)$ - ее подформулы, то P и F называются предикатным и функциональным параметрами формулы φ соответственно. Запись $\varphi(\bar{x}; P; F)$ будет означать формулу, чьи предметные, предикатные и функциональные параметры содержатся в наборах $\bar{x}$, $\bar{P}$ и $\bar{F}$. Обозначим построенный выше язык $L_0(\sigma^*)$.

Будем говорить, что формула φ находится в негативной нормальной форме (н.н.ф.), если в φ нет вхождений импликации и отрицание $\neg$ встречается только перед атомарными формулами. Легко понять, что каждая формула может быть эквивалентным образом представлена в н.н.ф. Используя этот факт, будем говорить, что вхождение атомарной подформулы φ в формулу ψ, находящуюся в н.н.ф., позитивно, если в данном вхождении перед φ не стоит знак отрицания $\neg$. Подформула φ позитивна в формуле ψ, если все вхождения φ в н.н.ф. (ψ) позитивны.

ОПРЕДЕЛЕНИЕ I. а) Формула φ называется Δ_0^+-формулой, если все ее атомарные формулы вида $P(\bar{t})$ и $F(\bar{t}) = q$, где $P, F \in \sigma^* \setminus \sigma$, позитивны, а все кванторы являются ограниченными.

б) Формула φ называется Σ^+-формулой, если она является элементом наименьшего класса формул, содержащего все Δ_0^+-формулы и замкнутого относительно дизъюнкции и конъюнкции формул, а также относительно навешивания неограниченного квантора существования.

в) Σ^+-формула φ называется Δ^+-формулой, если существует Σ^+-формула ψ, эквивалентная $\neg\varphi$.

Положим $\Delta_0 \leftrightarrows \Delta_0^+ \restriction \sigma$, $\Sigma = \Sigma^+ \restriction \sigma$ и $\Delta \leftrightarrows \Delta^+ \restriction \sigma$.

Определим теперь семантику формул. Пусть $\mathcal{M}$ - модель сигнатуры σ_0 и $HW(\mathcal{M})$ - ее стандартная списочная надстройка, состоящая из всех наследственно конечных списков над $\mathcal{M}$. Заметим, что для $s \equiv s'$ должно выполняться $M_s \subseteq M_{s'}$, где M_s - множество элементов модели сорта s. В дальнейшем будем предполагать, что $\mathcal{M}$ - конструктивная модель. Вообще говоря, мы можем даже считать, что для некоторых сортов и типов над $I \cup \{list\}$ соответствующие им предикаты равенства или другие предикаты могут быть рекурсивно-перечислимыми. В этом случае мы будем требовать, чтобы в Σ^+-формулы эти предикаты входили только позитивно. Итак, в общем случае $\mathcal{M}$ может быть и позитивной моделью. Следовательно, и пара $(\mathcal{M}, HW(\mathcal{M}))$ будет также позитивной моделью теории GES_1^+ (см. [I5]).

В наших формулах присутствуют предикатные и функциональные параметры. Следовательно, помимо базисной модели и ее надстройки, мы должны ввести в рассмотрение класс предикатов $\tilde{\mathbf{P}}$ и класс, вообще говоря, частичных функций $\tilde{\mathbf{F}}$. Поскольку функциональные переменные входят в формулы только в виде равенств $F(\bar{t}) = q$, то в качестве $\tilde{\mathbf{F}}$ нам удобнее будет рассматривать предикаты, являющиеся графиками частичных функций. Итак, моделью для нас будет четверка $\tilde{\mathcal{M}} = \langle \mathcal{M}, HW(\mathcal{M}), \tilde{\mathbf{P}}, \tilde{\mathbf{F}} \rangle$.

Интерпретацией нашего языка $L_0(\sigma^*)$ в $\tilde{\mathcal{M}}$ будет тройка отображений $\langle \nu, \mu, \eta \rangle$:

$$\nu : \mathbf{X}(I \cup \{list\}) \to |\mathcal{M}| \cup |HW(\mathcal{M})|,$$

$$\mu : \mathbf{P} \to \tilde{\mathbf{P}},$$

$$\eta : \mathbf{F} \to \tilde{\mathbf{F}},$$

для которых должны выполняться естественные требования на их согласованность с местностью и типизацией сигнатурных символов.

Одно неформальное соображение. При "вычислении" истинности формул языка $L_0(\sigma^*)$ на $\mu(P)$ (или $\eta(F)$), где $P, F \in \sigma^* \setminus \sigma$, лучше всего смотреть как на оракулы, которые в качестве ответов выдают признак истинности. В дальнейшем в качестве элементов классов $\tilde{\mathbf{P}}$ и $\tilde{\mathbf{F}}$ мы будем, как правило, рассматривать Σ-определимые множества. Напомним, что множество $A \subseteq |\mathcal{M}|^n \cup |HW(\mathcal{M})|^n$ называется Σ-определимым, если существует Σ-формула $\varphi(x_1, \ldots, x_n)$ такая, что $\bar{a} \in A \Leftrightarrow (\mathcal{M}, HW(\mathcal{M})) \models \varphi(\bar{a})$. Кроме того, если A – график функции и $\varphi(\bar{x}, y)$ – определяющая это множество формула, то должно выполняться $(\mathcal{M}, HW(\mathcal{M})) \models \varphi(\bar{x}, y_1) \wedge \varphi(\bar{x}, y_2) \supset y_1 = y_2$.

Пусть $Q, G \in \sigma^* \setminus \sigma$ – предикатная и соответственно функциональная переменные, $\varphi(\bar{y}; \bar{P}; \bar{F})$ и $\psi(\bar{z}; \bar{P}; \bar{F})$ – Σ^*-формулы. Выражение вида

$$Q(\bar{x}) \ \underline{опр} \ \varphi(\bar{y}; \bar{P}; \bar{F}) \qquad (1)$$

и

$$G(\bar{x}) = y \ \underline{опр} \ \psi(\bar{z}; \bar{P}; \bar{F}), \qquad (2)$$

где в (1) $\bar{x}$ – поднабор $\bar{y}$, а $\bar{x}, y$ – поднабор $\bar{z}$ в (2), будем называть Σ^*-определениями. Заметим, что в (1) и (2) переменные Q и G могут входить в наборы $\bar{P}$ и $\bar{F}$. В этом случае мы имеем дело с рекурсивными определениями. Далее, $Q(\bar{x})$ и $G(\bar{x}) = y$ в (1) и (2) будут называться заголовками, символы Q и G – именами, а формулы φ

и ψ – содержанием определений. Если α – определение, то через $H(\alpha)$, $N(\alpha)$ и $B(\alpha)$ будем обозначать его заголовок, имя и содержание соответственно. Запись $\alpha(\bar{x},N(\alpha);\bar{v},\bar{P}';\bar{F}')$ будет обозначать определение, у которого в случае (1): $\bar{v}=\bar{y}\setminus\bar{x}$, $\bar{P}'=\bar{P}\setminus\{N(\alpha)\}$, $\bar{F}'=\bar{F}$ и, следовательно, само определением имеет вид:

$$N(\alpha)(\bar{x}) \ \underline{\text{опр}} \ \varphi(\bar{x},\bar{v};\ N(\alpha),\ \bar{P}';\ \bar{F});$$

а в случае (2): $\bar{v}=\bar{z}\setminus(\bar{x}\cup\{y\})$, $\bar{P}'=\bar{P}$, $\bar{F}'=\bar{F}\setminus\{N(\alpha)\}$ и само определение имеет вид:

$$N(\alpha)(\bar{x})=y \ \underline{\text{опр}} \ \psi(\bar{x},\bar{y},\bar{v};\ \bar{P};\ N(\alpha),\ \bar{F}').$$

Элементы наборов $\bar{v}$, $\bar{P}'$ и $\bar{F}'$ называются соответственно предметными, предикатными и функциональными параметрами определения α.

Назовем Σ^+-схемой (сигнатуры σ) конечный набор Σ^+-определений. В дальнейшем без ограничения общности будем считать, что в рассматриваемых Σ^+-схемах нет двух определений с одинаковыми именами и $\bar{v}_\alpha \cap \bar{v}_\beta = \emptyset$, если $\alpha \neq \beta$. Если

$$S=\{\alpha_1(\bar{x}_1,N(\alpha_1);\bar{v}_1,\bar{P}_1,\bar{F}_1),\ldots,\alpha_n(\bar{x}_n,N(\alpha_n);\ \bar{v}_n,\bar{P}_n,\bar{F}_n)$$

есть Σ^+-схема, то элементы наборов

$$\bar{v}=\bigcup_{i=1}^{n}\bar{v}_i, \quad \bar{P}=\bigcup_{i=1}^{n}\bar{P}_i\setminus\{N(\alpha)\mid\alpha\in S\} \quad \text{и} \quad \bar{F}=\bigcup_{i=1}^{n}\bar{F}_i\setminus\{N(\alpha)\mid\alpha\in S\}$$

называются предметными, предикатными и функциональными параметрами схемы S и этот факт будем обозначать $S(N(\alpha_1),\ldots,N(\alpha_n);\ \bar{v},\bar{P},\bar{F})$. Иногда Σ^+-схемы мы будем записывать так:

$$\begin{array}{lll} N(\alpha_1)(\bar{x}_1) & \underline{\text{опр}} & B(\alpha_1)(\bar{x}_1,N(\alpha_1);\ \bar{v}_1,\bar{P}_1,\bar{F}_1); \\ \ldots & \ldots & \ldots \\ N(\alpha_n)(\bar{x}_n) & \underline{\text{опр}} & B(\alpha_n)(\bar{x}_n,N(\alpha_n);\ \bar{v}_n,\bar{P}_n,\bar{F}_n). \end{array}$$

Пусть $\varphi(\bar{x})$ – Σ^+-формула. Выражение вида $?\bar{y}.\text{вход}\ \bar{z}.\varphi(\bar{x})$; где $\bar{y}\cup\bar{z}\subseteq\bar{x}$, $\bar{y}\cap\bar{z}=\emptyset$ и $(\bar{x}\setminus(\bar{y}\cup\bar{z}))\cap\bar{v}=\emptyset$, будем называть запросом к S. Переменные из $\bar{z}$ называются входными, а переменные из $\bar{x}\setminus(\bar{y}\cup\bar{z})$ – параметрами запроса.

ОПРЕДЕЛЕНИЕ 2. Пара (S,q), где q – запрос к S, а S – Σ^+-схема, называется Σ^+-программой.

ПРИМЕР. Ниже приводится Σ^+-программа для задачи упорядочения элементов списка $\langle a,b,c\rangle$.

Схема:

$УВ(\alpha,\beta)$ опр $[$если $\beta = nil$ то $\alpha = nil$ иначе

$(\forall x \in \alpha)(x \in \beta) \wedge (\forall x \in \beta)(x \in \alpha) \wedge УП(\alpha)]$;

$УП(\alpha)$ опр $[ЛИН(\alpha) \wedge (tail(\alpha) = nil \vee (\forall \gamma \subseteq \alpha)$

$(tail(\gamma) \neq nil \supset \leq (head(tail(\gamma)), head(\gamma))]$;

$ЛИН(\alpha)$ опр $[List(\alpha) \wedge \alpha \neq nil \wedge (\forall x \in \alpha)\ P(x)]$;

Запрос: $?\alpha.\ УВ(\alpha, \langle a,b,c \rangle)$;.

Здесь предикат $УВ(\alpha,\beta)$ выражает свойство "α есть упорядоченная версия списка β ", $УП(\alpha)$ – "быть упорядоченным списком по отношению $\leq$ "; $ЛИН(\alpha)$ – "быть линейным списком, элементы которого удовлетворяют свойству P ". Таким образом, предикатными параметрами у этой программы будут $\leq$ и P . Запрос можно интерпретировать как указание на генерацию всех упорядоченных версий списка $\langle a,b,c \rangle$, где a,b,c – некоторые константы. Запись 'если φ то ψ иначе θ ' является другой формой записи формулы $(\varphi \wedge \psi) \vee \vee (\neg\varphi \wedge \theta)$.

§2. Денотационная семантика Σ^+-программ

Определение денотационной семантики Σ^+-программ мы начнем с описания семантики Σ^+-схем.

Пусть $S(N(\alpha_1), \ldots, N(\alpha_n); \bar{v}, \bar{P}, \bar{F})$ – Σ^+-схема. Через $\sigma_0(S)$ и $\sigma^*(S)$ будем соответственно обозначать сигнатурные символы из σ_0 и σ^*, встречающиеся в S . В дальнейшем $\sigma_0(S)$ и $\sigma^*(S)$ будем называть базисной и полной сигнатурами схемы S соответственно. Сигнатуру $\sigma_S = \sigma^*(S) \setminus \{N(\alpha) \mid \alpha \in S\}$ будем называть основной сигнатурой или просто сигнатурой схемы S.

Рассмотрим теперь конструктивную (или позитивную, если для содержания определений из S выполнены условия, указанные в §1) модель $\mathcal{M}$ сигнатуры $\sigma_0(S)$ вместе с ее списочной надстройкой $HW(\mathcal{M})$, а также семейства $\tilde{P}$ и $\tilde{F}$ предикатов и графиков функций. Четверку $\mathcal{M}_0 = \langle \mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F} \rangle$ будем называть предмоделью схемы S . Определим теперь оператор Γ_S, соотносящий при каждой интерпретации $\xi = \langle \nu, \mu, \eta \rangle$ параметров схемы $\bar{v}, \bar{P}$ и $\bar{F}$ произвольному набору предикатов $\langle Q_1^{m_1}, \ldots, Q_n^{m_n} \rangle$ из $\langle \mathcal{M}, HW(\mathcal{M}) \rangle$, где m_i – местность $N(\alpha_i)$, набор предикатов $\Gamma_S(\xi)(Q_1, \ldots, Q_n)$, определяе-

мый следующим образом:

$$\Gamma_S(\xi)(Q_1,\ldots,Q_n) \leftrightarrows \langle\{\bar{a}_1 \mid \mathcal{N}_0 \models B(\alpha_1)(\bar{a}_1, Q_1; \nu(\bar{v}_1), Q_2, \ldots, Q_n, \mu(\bar{P}), \eta(\bar{F}))\}, \ldots, \{\bar{a}_n \mid \mathcal{N}_0 \models B(\alpha_n)(\bar{a}_n, Q_n; \nu(\bar{v}_n), Q_1, \ldots, Q_{n-1}, \mu(\bar{P}), \eta(\bar{F}))\}\rangle .$$

Полагаем

$$\Gamma_S^0(\xi)(Q_1,\ldots,Q_n) \leftrightarrows \langle Q_1,\ldots,Q_n\rangle$$

и

$$\Gamma_S^{n+1}(\xi)(Q_1,\ldots,Q_n) \leftrightarrows \Gamma_S(\xi)(\Gamma_S^n(\xi)(Q_1,\ldots,Q_n)) .$$

Набор предикатов $\langle Q_1,\ldots,Q_n\rangle$ называется неподвижной точкой оператора $\Gamma_S(\xi)$, если $\Gamma_S(\xi)(Q_1,\ldots,Q_n) = \langle Q_1,\ldots,Q_n\rangle$. Неподвижная точка $\langle Q_1,\ldots,Q_n\rangle$ оператора $\Gamma_S(\xi)$ называется наименьшей и обозначается $\mathrm{LFP}(\Gamma_S(\xi))$, если для любой другой неподвижной точки $\langle Q_1',\ldots,Q_n'\rangle$ этого оператора имеет место $Q_i \subseteq Q_i'$; $i = 1,\ldots,n$.

ТЕОРЕМА I. Для каждых предмодели $\mathcal{N}_0$ схемы S и интерпретации ξ оператор $\Gamma_S(\xi)$ обладает $\mathrm{LFP}(\Gamma_S(\xi))$ и

$$\mathrm{LFP}(\Gamma_S(\xi)) = \bigcup_{n=1}^{\infty} \Gamma_S^n(\xi)(\emptyset,\ldots,\emptyset) .$$

ДОКАЗАТЕЛЬСТВО теоремы вытекает из принципа обобщенной Σ^+-непрерывности для Σ^+-формул (см. [I5]), а также из того, что $\Gamma_S(\xi)$ является монотонным оператором и что в качестве надстройки рассматривается стандартная надстройка $\mathrm{HW}(\mathcal{M})$.

В [4,I5] была сформулирована теорема, из которой следует Σ^+-определимость $\mathrm{LFP}(\Gamma_S(\xi))$ в случае, когда S состоит из одного рекурсивного определения. Этот результат обобщается и на произвольный случай. Ниже будет описана процедура, позволяющая строить наборы Σ^+-формул, определяющие в $\mathcal{N}_0$ для каждой интерпретации ξ наименьшую неподвижную точку оператора Γ_S, и напоминающая по своему содержанию метод "исключения неизвестных" решения уравнений. В качестве примера рассмотрим Σ^+-схему S, состоящую из двух определений вида (I):

α_1: $Q_1(\bar{x}_1)$ опр $\varphi_1(\bar{x}_1, Q_1, Q_2; \bar{v}_1, \bar{P}_1, \bar{F}_1)$;

α_2: $Q_2(\bar{x}_2)$ опр $\varphi_2(\bar{x}_2, Q_1, Q_2; \bar{v}_2, \bar{P}_2, \bar{F}_2)$;

Через S_1 (S_2) обозначим схему, состоящую из определения α_1 (α_2). Пусть $\psi_2(\bar{x}_2, Q_1; \bar{v}, \bar{P}, \bar{F})$ – Σ^+-формула, определяющая $LFP(\Gamma_{S_2}(\xi))$. Обозначим через $S_1[\psi_2]$ Σ^+-схему, полученную из S_1 подстановкой формулы ψ_2 вместо всех вхождений предиката Q_2. Пусть $\psi_1(\bar{x}_1; \bar{v}, \bar{P}, \bar{F})$ – Σ^+-формула, определяющая $LFP(\Gamma_{S_1[\psi_2]}(\xi))$.

ТЕОРЕМА 2. Для каждых предмодели $\mathcal{M}_0$ схемы S и интерпретации ξ

$$LFP(\Gamma_S(\xi)) = \langle\{ \bar{a}_1 | (\mathcal{M}_0, \xi) \models \psi_1(\bar{a}_1; \bar{v}, \bar{P}, \bar{F}) \},$$

$$\{ \bar{a}_2 | (\mathcal{M}_0, \xi) \models [\psi_2(\bar{a}_2, Q_1; \bar{v}_2, \bar{P}_2, \bar{F}_2)]^{Q_1}_{\psi_1} \} \rangle,$$

где $[\psi_2]^{Q_1}_{\psi_1}$ – результат подстановки формулы ψ_1 в ψ_2 вместо вхождений предиката Q_1.

ДОКАЗАТЕЛЬСТВО теоремы осуществляется проверкой того, что правая часть равенства есть неподвижная точка оператора $\Gamma_S(\xi)$, и последующими вычислениями, подтверждающими, что она есть подмножество левой.

Заметим, что порядок "исключения неизвестных" неважен для определения наименьшей неподвижной точки оператора $\Gamma_S(\xi)$. Но это обстоятельство может оказаться важным при ее конкретном вычислении.

Теперь мы в состоянии определить семантику Σ^+-схемы $S(N(\alpha_1), \ldots, N(\alpha_n); \bar{v}, \bar{P}, \bar{F})$. Пусть $\mathcal{M}_0$ – предмодель схемы S и $\xi = \langle \nu, \mu, \eta \rangle$ – некоторая интерпретация ее параметров $\bar{v}, \bar{P}$ и $\bar{F}$. Доопределим ξ до $\bar{\xi}$, поставив в соответствие кортежу $\langle N(\alpha_1), \ldots, N(\alpha_n) \rangle$ набор $LFP(\Gamma_S(\xi))$.

Пару $(\mathcal{M}_0, \bar{\xi})$ будем называть моделью Σ^+-схемы S.

Рассмотрим теперь Σ^+-программу (S, q), где $q = ?\bar{x}.$вход $\bar{y}.\varphi(\bar{x}, \bar{y}, \bar{z}, \ldots)$.

ОПРЕДЕЛЕНИЕ 3. Денотационной семантикой программы (S, q) в модели $(\mathcal{M}_0, \bar{\xi})$ называется множество

$$Den^{\bar{b}, \bar{c}, \ldots}_q = \{ \bar{a} | (\mathcal{M}_0, \bar{\xi}) \models \varphi(\bar{a}, \bar{b}, \bar{c}, \ldots) \}.$$

Сейчас мы сформулируем результаты, поясняющие роль Σ^+-программ при иерархическом построении спецификаций задач.

Пусть σ' – многосортная конечная сигнатура с множеством сортов $I(\sigma')$, а $\mathcal{M}$ – модель этой сигнатуры. Следующее определение является "рекурсивно-параметрическим" обобщением определения Σ-определимости из [2].

ОПРЕДЕЛЕНИЕ 4. Модель $\mathcal{N}$ Σ^+-определима в модели $(\mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F})$ сигнатуры σ', если существуют Σ^+-схема $S(N(\alpha_1), \ldots, N(\alpha_n); \bar{\nu}, \bar{P}, \bar{F})$ и интерпретация $\xi = \langle \nu, \mu, \eta \rangle$ параметров схемы такие, что $S(N(\alpha_1), \ldots, N(\alpha_n); \nu(\bar{\nu}), \mu(\bar{P}), \eta(\bar{F}))$ определяет для каждого сорта $s \in I(\sigma')$ множество $X_s \subseteq |\mathcal{M}| \cup |HW(\mathcal{M})|$ и отношение эквивалентности Θ_s на X_s, а для каждого сигнатурного предикатного символа P (функционального символа F) из σ' предикат Q_P (график функции G_F) так, что $\Theta = (\Theta_s)_{s \in I(\sigma')}$ есть отношение конгруэнтности на $\mathcal{K} = \langle (X_s)_{s \in I(\sigma')}, \{Q_P\}_{P \in \sigma'}, \{G_F\}_{F \in \sigma'} \rangle$ и $\mathcal{K}/\Theta$ изоморфна $\mathcal{N}$.

Следуя [2], будем говорить, что $\mathcal{N}$ Σ^+-транслируема в $\mathcal{M}$, и писать $\mathcal{N} \leq_{T^+} \mathcal{M}$, если $\mathcal{N}$ Σ^+-определима в $(\mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F})$.

ПРЕДЛОЖЕНИЕ 1. Если каждое $Q \in \mu(\tilde{P}) \cup \eta(\tilde{F})$ является Σ^+-определимым множеством, $\mathcal{N} \leq_{T^+} \mathcal{M}$ и $\mathcal{M}$ – позитивная модель, то $\mathcal{N}$ – также позитивная модель.

ТЕОРЕМА 3. (ср.с [2]). 1) Если $\mathcal{N} \leq_{T^+} \mathcal{M}$, то

$$\langle \mathcal{N}, HW(\mathcal{N}) \rangle \leq_{T^+} \mathcal{M};$$

2) отношение $\leq_{T^+}$ является отношением предпорядка.

В дальнейшем для класса $\mathbb{K}$ моделей сигнатуры σ' через $HW^*(\mathbb{K})$ будем обозначать класс моделей вида $\langle \mathcal{N}, HW(\mathcal{N}), \tilde{P}, \tilde{F} \rangle$, где $\mathcal{N} \in \mathbb{K}$.

ОПРЕДЕЛЕНИЕ 5. Класс моделей $\mathbb{K}_0$ сигнатуры σ' Σ^+-определим в $HW^*(\mathbb{K}_1)$, где $\mathbb{K}_1$ – класс моделей сигнатуры σ_0, если существует Σ^+-схема $S(N(\alpha_1), \ldots, N(\alpha_n); \bar{\nu}, \bar{P}, \bar{F})$ такая, что для каждой модели $\mathcal{N} \in \mathbb{K}_0$ найдутся модель $\langle \mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F} \rangle$ из $HW^*(\mathbb{K})$ и интерпретация $\xi = \langle \nu, \mu, \eta \rangle$ параметров схемы S такие, что S Σ^+-определяет $\mathcal{N}$ в $\langle \mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F} \rangle$.

Как и ранее, если класс $\mathbb{K}_0$ Σ^+-определим в $HW^*(\mathbb{K}_1)$, то будем писать $\mathbb{K}_0 \leq_{T^+} \mathbb{K}_1$ и говорить, что $\mathbb{K}_0$ Σ^+-транслируем в $\mathbb{K}_1$. Особенно здесь интересен случай, когда $\mathbb{K}_1 = \{\mathcal{M}\}$. В этом случае будем писать $\mathbb{K}_0 \leq_{T^+} \mathcal{M}$.

СЛЕДСТВИЕ I. 1) Если $\mathbb{K}_0 \leq_{T^+} \mathbb{K}_1$, то

$$\{ (\mathcal{M}, HW(\mathcal{M})) \mid \mathcal{M} \in \mathbb{K}_0 \} \leq_{T^+} \mathbb{K}_1 .$$

2) Если $\mathbb{K}_0 \leq_{T^+} \mathbb{K}_1$ и $\mathbb{K}_1 \leq_{T^+} \mathbb{K}_2$, то $\mathbb{K}_0 \leq_{T^+} \mathbb{K}_2$.

§3. Процедурная семантика Σ^+-программ

Чтобы говорить о процедурной семантике Σ^+-программ, введем в рассмотрение формальные объекты, которые будут называться стратегиями (исполнения Σ^+-программ). Среди всех стратегий будет выделена некоторая базисная стратегия STAND, которая и будет подразумеваться, если при построении Σ-программы ничего не говорится о стратегии ее исполнения.

Будем считать, что базисная сигнатура σ_0 содержит сорта Σ^+-программ, формул, термов и т.п., а также сорт наследственно конечных списков. В дальнейшем мы будем отождествлять такие "синтаксические" объекты, как программы, формулы, термы и т.п. с соответствующими наследственно конечными списками. Это делается с целью определения на данных объектах "быстрых" функций их обработки. В свою очередь, необходимость выделения таких функций объясняется следующими содержаниями. В Σ^+-программах (особенно в Δ_0-программах), определяющих функции, достаточно быстро (во всяком случае с теоретической точки зрения) осуществляется проверка истинности условий, но вот поиск значений для таких функций очень неэффективен из-за возникающего при этом ненаправленного перебора элементов в модели без учета преследуемой цели. Идея выбора стратегии исполнения программы наиболее адекватной данной цели как раз и есть средство борьбы с данным недостатком. Но поскольку стратегия в семантическом программировании вновь задается как Σ-программа, то и возникает задача выделения быстро исполнимых функций. Данный класс (назовем его ПРИМ) строится следующим образом. Предположим, что в нашем распоряжении уже имеется некий набор "базисных" быст-

рых функций (некоторые из них будут указаны ниже)[*]. ПРИМ получается замыканием этого набора функций посредством следующих Σ^+-определимых конструкторов.

3.1. Рекурсия по спискам. Если h и f - быстро вычислимые функции, то функция g, определенная следующим образом:

$$\begin{cases} g(nil,\bar{x}) = f(\bar{x}), \\ g(cons(\bar{a},y),\bar{x}) = h(\bar{a},y,\bar{x},g(\bar{a},\bar{x})), \end{cases}$$

также является быстро вычислимой. Соответствующий конструктор обозначим РЕК (f,g).

3.2. Подстановка. Если $h(x_1,\ldots,x_n)$ и $f_1,\ldots,f_n$ - быстро вычислимые функции, то $g(\bar{y}) = h(f_1(\bar{y}),\ldots,f_n(\bar{y}))$ также является быстро вычислимой функцией; соответствующий конструктор обозначается ПОДСТ $(h,f_1,\ldots,f_n)$.

Легко понять, что таким образом мы получаем класс списочных "примитивно рекурсивных" функций. Но для практических нужд потребуется только небольшой фрагмент класса ПРИМ. Рассмотрим соответствующие функциональные конструкторы.

3.3. Пусть f - базисная операция, определенная на некотором базисном множестве. Распространением EXT(f) функции f называется функция, определенная на списочной надстройке над данным базисным множеством посредством следующего определения: EXT(f)$(\bar{a},x)$ = = f$(\bar{a},x)$, если $\bar{a},x$ - набор элементов базисного множества; EXT(f)$(\bar{a},nil)$ = nil и

$$EXT(f)(\bar{a},cons(b,x)) = cons(EXT(f)(\bar{a},b),f(\bar{a},x)),$$

если b - список.

3.4. Если φ - Δ_0-формула, то конструктор Δ_0-выделения CON(φ) (см. также [I6]) по φ строит функцию $\hat{\varphi}$, определяемую следующим образом:

$$\hat{\varphi}(nil) = nil,$$

$$\hat{\varphi}(cons(x,a)) = \begin{cases} \hat{\varphi}(x), & \text{если } \neg\varphi(a), \\ cons(\hat{\varphi}(x),a), & \text{если } \varphi(a). \end{cases}$$

---

[*] Эти функции в языке СИГМА действительно являются базисными, т.е. встроенными, исходными операциями.

3.5. Конструктор направленного поиска UNTIL(f, φ) = h определяется так: если f - функция, а φ - Δ_0-формула, то функция $h(\bar{x}) = \alpha$ - это такая функция, что α является первым элементом в последовательности $\alpha_0 = nil, \ldots, \alpha_{n+1} = f(\alpha_n, x), \ldots$, для которого $\varphi(\alpha)$.

ПРЕДЛОЖЕНИЕ 2. Класс ПРИМ замкнут относительно конструкторов EXT , CON и UNTIL.

В связи с данным утверждением естественно возникает следующий вопрос: можно ли получить все функции из ПРИМ, используя только конструкторы EXT , CON и UNTIL? Ответ авторам неизвестен,но, скорее всего, нельзя.

В дальнейшем, при определении стратегий в виде Σ^+-программ, функции из класса ПРИМ могут рассматриваться как базисные. Кроме того, в набор базисных функций включаются для всех позитивных сигнатурных отношений $P \in \sigma$ процедуры $Prod_P$ последовательного порождения всех наборов элементов модели, удовлетворяющих предикату P (если P - рекурсивный предикат, то наряду с процедурой порождения $Prod_P$ рассматривается как базисная и процедура порождения $Prod_{\neg P}$ для $\neg P$), и процедура $Eval_P$ проверки истинности для предиката P на элементах соответствующего основного множества.Мы будем также считать, что в нашем распоряжении имеются процедуры порождения $Prod_{|\mathcal{M}|^n}$,$n \geq 1$, элементов множеств $|\mathcal{M}|^n$, элементов настройки - $Prod_{HW}$,элементов всей списочной модели - $Prod_{\mathcal{M},HW}$ и т.п. Для практических целей желательно иметь более богатый набор порождающих процедур; например, желательно иметь для каждого $s \in I$ процедуру $Prod_s$ генерации элементов M_s и $HW(M_s)$. Если Prod - некоторая процедура генерации элементов, то через Prod(n) будем обозначать элемент, сгенерированный этой процедурой на n-м шаге. Наконец, будем предполагать, что для каждого базисного предиката P (и для его дополнения, если P разрешим) мы располагаем процедурой $Solv_P$, выдающей для каждого набора термов $t_1, \ldots, t_n$ булевозначную функцию $Bool_P[t_1, \ldots, t_n](\bar{x})$ (здесь $\bar{x}$ - набор переменных, входящих в $t_1, \ldots, t_n$), для которой справедливо

$$Bool_P[t_1, \ldots, t_n](\bar{b}) = 1 \Leftrightarrow \langle t_1[\bar{x}:=\bar{b}], \ldots, t_n[\bar{x}:=\bar{b}] \rangle \in P .$$

ОПРЕДЕЛЕНИЕ 6. Стратегия исполнения Σ^+-программ (или просто стратегия) - это вычислимое отношение STR , соотносящее каждой

Σ^+-программе π наборы $\langle \xi_1, \ldots, \xi_n \rangle$ такие, что каждое ξ_i, $i \le n$, - это пара (π', str'), где π' - Σ^+-программа, а str' - имя стратегии STR'.

В практической деятельности нас, конечно же, в первую очередь интересуют эффективные стратегии. С этой целью, например, мы можем ограничиться отношениями, задаваемыми только быстро исполнимыми алгоритмами. С теоретической же точки зрения допустимы любые вычислимые отношения.

Будем говорить, что Σ^+-программа π является концевой для стратегии STR, если $\langle \pi, (\pi, str) \rangle \in STR$. Будем говорить, что Σ^+-программа π псевдоконцевая для стратегии STR, если она либо концевая, либо имеет вид $(s, ?\bar{y}.$ вход $\bar{x}.\varphi(\bar{y}, \bar{x}, \bar{z}))$, где φ - атомарная формула сигнатуры $\sigma^* \setminus \sigma$.

Ниже мы определим так называемую стандартную стратегию $STAND$. Концевыми для $STAND$ будут следующие виды программ:

а) $(s, ?\bar{y}.$ вход $\bar{x}.$ FALSE $)$. В данном случае множество ответов на запрос пусто;

б) $(s, ?\bar{y}.$ вход $\bar{x}.$ TRUE$)$. Для этой программы запрос интерпретируется как указание генерации всех элементов той области, к которой принадлежат переменные $\bar{y}$ согласно своим спецификациям;

в) $(s, ?\bar{y}.$ вход $\bar{x}.\bar{y} = \bar{t}(\bar{x}, \bar{z}))$. Здесь предполагается, что переменные из $\bar{y}$ не встречаются в наборах $\bar{x}$ и $\bar{z}$. В записи же термов из $\bar{t}$ могут входить номера порождающих процедур в следующем виде: $Prod^{j_1, \ldots, j_k}$, где $1 \le j_1 \le \ldots \le j_k \le n$ и n - длина кортежа элементов, генерируемых процедурой $Prod$. Данное имя может рассматриваться как такой кортеж констант $\langle a_1, \ldots, a_k \rangle$ длины k (или список длины k - все зависит от констекста использования), что a_l, $1 \le l \le k$, есть j_l-я компонента набора элементов, генерируемого на очередном шаге процедурой $Prod$. Если в $\bar{t}$ таких имен не встречается, то запрос интерпретируется как параметрическая программа (параметрами являются переменные из $\bar{z}$), позволяющая по заданным входным значениям для $\bar{x}$ "вычислить" значение для $\bar{y}$ (как многозначную функцию с аргументами из $\bar{z}$). В случае, когда в записи $\bar{t}$ встречаются имена процедур генерации элементов, то запрос интерпретируется как набор инструкций, указывающий, как по заданному набору значений $\bar{x}$ (и $\bar{z}$) генерировать значения переменных из $\bar{y}$, полагая, что шаги генерации различных процедур генерации синхронизированы;

г) (S, ?. вход $\bar{x}.P^{\varepsilon}(t_1,\ldots,t_n)$), P - базисный предикат, т.е. $P\in\sigma$, $\varepsilon\in\{0,1\}$, $P^0=\neg P$, $P^1=P$ и $\bar{x}$ - поднабор переменных, входящих в $t_1,\ldots,t_n$. В данном случае запрос интерпретируется как указание "проверить истинность предиката P^{ε}", для осуществления чего используется процедура $\mathrm{Bool}_{P^{\varepsilon}}[t_1,\ldots,t_n](\bar{x},\bar{z})$, где $\bar{x}\cup\bar{z}$ - набор всех переменных из $\bar{t}$. Заметим, что в роли P могут выступать и предикаты равенства. Кроме того, в записи $t_1,\ldots,t_n$ могут, как и в случае "в", встречаться имена процедур генерации. В этом случае истинность предиката проверяется для каждого сгенерированного набора значений термов $t_1,\ldots,t_n$.

Приводимое ниже описание стратегии STAND заключается в указании переходов от программ к соответствующему множеству наборов, о чем шла речь в определении 6. На эти переходы можно смотреть, например, как на систему правил переписывания или как на правила вывода некоторого исчисления.

Схема переходов стратегии STAND.

I. Если y не встречается в $\bar{x}\cup\bar{z}$, то

$$(S,?\,y.\ \text{вход}\ \bar{x}.\varphi(\bar{x},\bar{z}))\to\{\langle((S,\ ?.\ \text{вход}\ \bar{x}.\varphi(\bar{x},\bar{z})),\mathrm{stand})\rangle\}.$$

2. Если P - базисный предикат, то

$$(S,?\,y.\ \text{вход}\ \bar{x}.P^{\varepsilon}(\bar{t}(\bar{x},y,\bar{z})))\to$$
$$\to\{\langle((S,?y.\ \text{вход}\ \bar{x}.[y=\mathrm{Prod}\wedge P^{\varepsilon}(\bar{t}(\bar{x},\mathrm{Prod},z))]),\mathrm{stand})\rangle\},$$

где Prod - подходящая процедура генерации элементов модели, выбор которой определяется спецификацией переменной y. Например, для программы $(S,?\,y.\ \text{вход}\ \bar{x}.P^{\varepsilon}(\bar{t},y,\bar{q}))$, где y не входит в $\bar{t}$ и $\bar{q}$, такой процедурой будет $\mathrm{Prod}^{i}_{P^{\varepsilon}}$ при условии, что y - i-я компонента кортежа $\langle\bar{t},y,\bar{q}\rangle$. (Для данного случая возможны самые разнообразные схемы переходов - все зависит от богатства набора процедур генерации, которым мы располагаем.)

3. $(S,\ ?y.\ \text{вход}\ \bar{x}.[\forall z\in t.\varphi(y,\bar{x},z,\bar{w})])\to\{\langle((S,\ ?y.\ \text{вход}\ \bar{x}.\ \varphi(y,\bar{x},\mathrm{head}(t),\bar{w})),\mathrm{stand}),((S,\ ?y.\ \text{вход}\ \bar{x}.\ [\forall z\in\mathrm{tail}(t).\varphi(y,\bar{x},z,\bar{w})],\mathrm{stand})\rangle,\langle((\emptyset,\ ?y.\ \text{вход}\ \bar{x}.\ t=\mathrm{nil}),\mathrm{stand})\rangle\}.$

4. $(S,\ ?y.\ \text{вход}\ \bar{x}.[\exists z\in t.\varphi(y,\bar{x},z,\bar{w})])\to\{\langle((S,\ ?y.\ \text{вход}\ \bar{x}.\ [t\neq\mathrm{nil}\wedge\varphi(y,\bar{x},\mathrm{head}(t),\bar{w})]),\mathrm{stand})\rangle,\langle((S.\ ?y.\ \text{вход}\ \bar{x}.[t\neq\mathrm{nil}\wedge\exists z\in\mathrm{tail}(t).\varphi(y,\bar{x},z,\bar{w})]),\mathrm{stand}),\langle((\emptyset,\ ?y.\ \text{вход}\ \bar{x}.\mathrm{FALSE}),\mathrm{stand})\rangle\}$. Здесь $\emptyset$ - пустая схема.

5. $(S, ?y.\ \text{вход}\ \bar{x}.\ \varphi_1 \wedge \varphi_2) \to \{\langle((S, ?y.\ \text{вход}\ \bar{x}.\ \varphi_1), stand), ((S, ?y.\ \text{вход}\ \bar{x}.\ \varphi_2), stand)\rangle\}$.

6. $(S, ?y.\ \text{вход}\ \varphi_1 \vee \varphi_2) \to \{\langle((S, ?y.\ \text{вход}\ \bar{x}.\ \varphi_1), stand)\rangle, \langle((S, ?y.\ \text{вход}\ \bar{x}.\ \varphi_2), stand)\rangle\}$.

7.

а) Если $\alpha \in S$, то $(S, ?y.\ \text{вход}\ \bar{x}.N(\alpha)(\bar{t})) \to \{\langle((S, ?y.\ \text{вход}\ \bar{x}.B(\alpha)(\bar{t})), stand)\rangle\}$.

б) Если $\alpha \in S$, то $(S, ?y.\ \text{вход}\ \bar{x}.N(\alpha)(\bar{t})) \to \{\langle((S, ?y.\ \text{вход}\ \bar{x}.\ulcorner LFP(\Gamma_\alpha)\urcorner(\bar{t})), stand)\rangle\}$, где $\ulcorner LFP(\Gamma_\alpha)\urcorner$ - Σ^+-формула, определяющая соответствующую α компоненту наименьшей неподвижной точки оператора Γ_S.

Можно пользоваться любым из этих переходов ("а" или "б") - с теоретической точки зрения они равносильны. Что же касается практического их применения, то переход 7 "б" занимает промежуточное положение между чисто интерпретационным переходом 7 "а" и компиляционным исполнением Σ^+-программ (см. введение).

Помимо этих переходов, при использовании стратегии Stand (да и других стратегий) могут оказаться полезными переходы, основанные на семантической информации. Например, если известно, что $(\mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F}) \models (\varphi \leftrightarrow \psi)$, то к переходам 1-7 можно добавить

8. $(S, ?y.\text{вход}\ \bar{x}.\varphi) \to \{\langle((S, ?y.\text{вход}\ \bar{x}.\psi), stand)\rangle\}$.

Так, скажем, если $(\mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F}) \models \varphi\ (y, \bar{x}, \bar{w})$, то вполне оправдан следующий переход:

8'. $(S, ?y.\ \text{вход}\ \bar{x}.\ \varphi) \to \{\langle((\emptyset, ?y.\ \text{вход}\ \bar{x}.TRUE), stand)\rangle\}$.

Аналогично, если $(\mathcal{M}, HW(\mathcal{M}), \tilde{P}, \tilde{F}) \models \neg\varphi$, то

8". $(S, ?y.\ \text{вход}\ \bar{x}.\varphi) \to \{\langle((\emptyset, ?y.\text{вход}\ \bar{x}, FALSE), stand)\rangle\}$.

Правило 8 может оказаться полезным, скажем, для оптимизации программ или при параллельном исполнении Σ^+-программ, когда в качестве ψ можно взять формулу, эквивалентную и представляющую собой конъюнкцию независимых (по переменным) подформул или дизъюнкцию подформул.

ОПРЕДЕЛЕНИЕ 7. Исполнением программы π по стратегии STR (или STR-исполнением) называется последовательность $\gamma_1, \ldots, \gamma_n, \ldots$ такая, что $\gamma_0 = \langle(\pi, str)\rangle$ и γ_{i+1} (для любого $i \geq 0$) есть набор пар $\langle$ программа, имя стратегии $\rangle$, получаемый из γ_i следующим образом:

$\gamma^0_{i+1} \leftarrow nil$,

$cons(\gamma^1_i, \xi)$, если $\xi = (\pi', str')$ есть (1+1)-я компонента γ_i и π' – псевдоконцевая программа для STR';

$conc(\gamma^1_i, \langle \eta_1, \ldots, \eta_k \rangle)$, если (1+1)-я компонента γ_i есть пара (π', str') такая, что $\langle \pi', \langle \eta_1, \ldots, \eta_k \rangle \rangle \in STR'$.

Окончательно полагаем: $\gamma_{i+1} = \gamma^n_{i+1}$, где n – наименьшее число такое, что $\gamma^n_{i+1} = \gamma^{n+1}_{i+1}$. STR-исполнение $\gamma_1, \ldots, \gamma_n, \ldots$ программы π конечно, если найдется γ_n, $n \geq 0$, такое, что $\gamma_n = \gamma_{n+1}$.

Конечное STR-исполнение $\gamma_0, \ldots, \gamma_n$ программы π называется псевдоконцевым (концевым), если γ_n состоит только из пар (π', str'), в которых π'-псевдоконцевая (концевая) программа для STR'. Будем говорить, что псевдоконцевое STR-исполнение программы π результативно для модели $(\mathcal{M}_0, \bar{\xi})$ Σ^+-схемы программы π, если существует такое означивание переменных, входящих в формулы запросов программ из γ_n, что одноименные переменные получают при этом одни и те же значения, а формулы запросов при таком означивании становятся истинными в модели $(\mathcal{M}_0, \bar{\xi})$.

Следующая теорема утверждает, что стандартная стратегия является корректной и полной относительно денотационной семантики Σ^+-программ.

ТЕОРЕМА 4. Пусть $\pi = (S, ?y.\text{вход } \bar{x}.\ \varphi(y, \bar{x}, \bar{z}))$ – Σ^+-программа и $(\mathcal{M}_0, \bar{\xi})$ – модель Σ^+-схемы S. Тогда для каждого набора значений $\bar{b}$ и $\bar{c}$ для переменных из $\bar{x}$ и $\bar{z}$ соответственно найдется элемент $a \in |\mathcal{M}_0|$ такой, что $(\mathcal{M}_0, \bar{\xi}) \models \varphi(a, \bar{b}, \bar{c}) \Leftrightarrow$ существует результативное STAND-исполнение $\gamma_0, \ldots, \gamma_n$ программы π для модели $(\mathcal{M}_0, \bar{\xi})$ такое, что при означивании в γ_n переменных из $\bar{x}$ и $\bar{z}$ константами из $\bar{b}$ и $\bar{c}$ соответственно переменная y получит в качестве своего значения константу a.

Доказательство теоремы осуществляется непосредственно использованием определения стратегии STAND.

§4. Σ^+-представимость логических параметрических программ

Рассмотрим логическую программу $L = (R,q)$ (см., например, [I7]), определяющую линейные списки над некоторым множеством:

$$R:\ List(nil) \leftarrow ;\ List(cons(y,x)) \leftarrow P(x), List(y)\ ,$$
$$q:\ \leftarrow List(a)\ . \qquad (3)$$

В данном примере, как и в случае с Σ^+-программами, P выступает в роли предикатного параметра, значением которого могут быть унарные предикаты. В дальнейшем, если $\bar{P}$ – набор предикатных параметров программы $L = (R,q)$ (т.е. набор предикатов, входящих в тела правил программы, но не встречающихся в заголовках), то будем писать $L(\bar{P})$.

Как известно, при традиционной теоретико-модельной семантике логической программы, как и при семантике неподвижной точки (см., например, [I7]), предикатным параметрам будут соотнесены в качестве их денотатов пустые отношения на множестве замкнутых объектных термов. С другой стороны, на предикатный параметр можно смотреть и как на "вызов" логической программы, определяющей содержание данного параметра. Например, запрос в (3) может быть уточнен так: q': $\leftarrow List(a)\ [P:= Nat]$, интерпретируемый как указание на необходимость определить, является ли список a списком натуральных чисел. Здесь предикат Nat может задаваться либо некоторой логической программой (ее текст, скажем, может быть приведен также в запросе), либо встроенным предикатом (как в данном случае). Заметим, что возможны и рекурсивные описания значений предикатных параметров: q'': $\leftarrow List(a)\ [P:= List[P:= Nat]]$.

Таким образом, в данном случае решается задача, является ли a списком списков натуральных чисел. В дальнейшем мы будем рассматривать логические программы, у которых в запросах также могут встречаться предикатные параметры, но отсутствует спецификация таких параметров: q''' : $\leftarrow List(a), P(a)$.

Вернемся к рассмотрению программы (3). Перепишем ее в следующем виде:

$$S(L):\ List(x)\ \underline{опр}\ [x = nil \vee \exists y,z(x = cons\ (y,z) \wedge$$
$$\wedge P(z) \wedge List(y))];$$
$$q(L):\ \ ?.\ List(a)\ .$$

Легко видеть, что мы получили Σ^+-программу $(S(L), q(L))$, схема которой состоит из единственного Σ^+-определения с предикатным параметром P.

В общем случае трансляция параметрических логических программ в Σ^+-программы выглядит так (см. также [4,10]). Пусть $L(\bar{P}) = (R, q)$ – параметрическая логическая программа

$$R:\ A_0^1 \leftarrow A_1^1, \ldots, A_{m_1}^1$$
$$\ldots$$
$$A_0^k \leftarrow A_1^k, \ldots, A_{m_k}^k$$
$$q:\ \leftarrow C_1, \ldots, C_n\ .$$

Соберем вместе все правила из R, у которых совпадают предикатные символы для их заголовков. Пусть, например, для предикатного символа T это будут правила

$$T(\bar{t}_1(\bar{y}_1)) \leftarrow A_1^{i_1}, \ldots, A_{m_{i_1}}^{i_1},$$
$$\ldots \qquad (4)$$
$$T(\bar{t}_l(\bar{y}_l)) \leftarrow A_1^{i_l}, \ldots, A_{m_{i_l}}^{i_l}\ .$$

Здесь $\bar{y}_j$ – переменные, входящие в набор термов $\bar{t}_j$. Для (4) построим следующее Σ^+-определение $\alpha(T)$:

$$T(\bar{x}) \ \underline{\text{опр}}\ \exists y_1 \ldots y_l \Big[\bigvee_{j=1}^{l} (\bar{x}_j = \bar{t}_j(\bar{y}_j) \wedge A_1^{i_j} \wedge \ldots \wedge A_{m_{i_j}}^{i_j}) \Big].$$

Заметим, что переменные из набора $\bar{x}$ не должны встречаться в (3). Построенные таким образом определения соберем в Σ^+-схему $S(L)$:

$$T_1(\bar{x}_1) \ \underline{\text{опр}}\ \varphi_1\ ;$$
$$\ldots$$
$$T_s(\bar{x}_s) \ \underline{\text{опр}}\ \varphi_s\ ,$$

а дизъюнкт $q = \leftarrow C_1, \ldots, C_n$ перепишем в виде

$$q(L) = ?\bar{z}.C_1 \ \ldots \ C_n\ ,$$

где $\bar{z}$ – набор всех свободных переменных, входящих в дизъюнкт. Σ^+-программу $\pi(L) = (S(L), q(L))$ будем называть Σ^+-представлением программы $L(\bar{P})$.

Обозначим через $\sigma(L)$ множество констант и функциональных символов, встречающихся в L. Пусть $I_{\sigma(L)}$ – множество всех замкнутых термов сигнатуры $\sigma(L)$, и рассмотрим модель $\mathcal{M} = (I_{\sigma(L)}, HW(I_{\sigma(L)}), \tilde{\mathbb{P}})$ с обычной интерпретацией сигнатуры $\sigma(L)$, где $\tilde{\mathbb{P}}$ – некоторый класс (Σ-определимых) предикатов. Очевидно, что $\mathcal{M}$ – предмодель для $S(L)$.

Пусть $\xi : \bar{P} \to \tilde{\mathbb{P}}$ – некоторая интерпретация предикатных переменных программы $L(\bar{P}) = (R, q)$. Определим множество

$$R^{\xi} = R \cup \{ P^n(\bar{t}) \mid \bar{t} \in I^n_{\sigma(L)}, P^n \in \bar{P}, \bar{t} \in \xi(P^n) \}.$$

Будем писать $R \vdash_{\xi} A(\bar{t})$, где $A(\bar{t})$ – атомарная формула, если $R^{\xi} \vdash A(\bar{t})$. Напомним, что если G – набор квазитождеств и A – атомарная формула, то $G \vdash A$ означает, что существует конечная последовательность формул $\varphi_0, \ldots, \varphi_n$, $n \geq 0$, такая, что $\varphi_n = A$ и каждая φ_i, $i \leq n$, является либо квазитождеством из G, либо частным случаем некоторого φ_j, $j < i$ (т.е. $\varphi_i = \theta\varphi_j$, где $\theta: X \to I_{\sigma(L)}(X)$ – подстановка; здесь X – множество переменных, $I_{\sigma(L)}(X)$ – множество термов сигнатуры $\sigma(L)$), либо φ_i получена из предыдущих применением правила modus ponens.

Если $\mathcal{M}$ – предмодель для $S(L)$ и $\xi : \bar{P} \to \tilde{\mathbb{P}}$ – интерпретация предикатных параметров логической программы $L(\bar{P})$, то через $(\mathcal{M}, \bar{\xi})$ обозначим модель для Σ^+-схемы $S(L)$ (см. §2). Следующая теорема подтверждает корректность описанной выше схемы трансляции логических программ в Σ^+-программы.

ТЕОРЕМА 5. Пусть $L(\bar{P}) = (R, q)$ – логическая параметрическая программа и $(\mathcal{M}, \bar{\xi})$ – модель для $S(L)$. Тогда

$$R \vdash_{\xi} q(\bar{t}) \Leftrightarrow (\mathcal{M}, \xi) \models q(L)(\bar{t}).$$

Следствием теорем 5 и 4 (§3) является тот факт, что параметрические логические и Σ^+-программы могут исполняться одним и тем же механизмом, скажем, стратегией STAND. Это и дает возможность в рамках одного языка сочетать аксиоматическое описание моделей и их элементарную определимость, о чем шла речь во введении.

Что же касается Σ^+-определимости параметрических логических программ из [I7], то здесь ситуация несколько иная и схема трансляции в общем случае выглядит иначе: вначале транслируем логические программы в Σ-выражение Ершова (схема трансляции похожа на

приводимую в данном параграфе), а затем используем трансляцию Σ-выражений в Σ^+-программы (см. [3] или [I4]). И вновь можно показать, что подобная Σ^+-представимость корректна и полна.

З а к л ю ч е н и е

В заключение укажем перспективные, на наш взгляд, направления дальнейших исследований.

Прежде всего, требует глубокого анализа понятие стратегии исполнения Σ^+-программ. Возможность представления стратегии в виде Σ^+-схемы позволяет в общем случае в семантической программе выделить две компоненты: логическую (денотационную) и управляющую (процедурную). Данное обстоятельство открывает большие возможности по написанию действительно эффективно исполняемых программ. Оно же указывает на целесообразность введения в язык дополнительных конструкций (см.также [I6]).

Исполнение Σ^+-программы состоит в "раскрутке" определений, описывающих предметную область, в сведении определяемых предикатов и функций к базисным, сигнатурным процедурам. Конкретная направленность такой "раскрутки" и составляет содержание той или иной стратегии. Подобные стратегии могут быть либо встроенными (скажем, стратегия STAND) и, следовательно, используемыми по умолчанию, либо явно описанными в виде соответствующей процедурной компоненты. Однако можно предусмотреть возможность и неявной настройки стратегий на специфику решаемой задачи. Такая настройка может быть осуществлена, например, с помощью дополнительных аксиом, описывающих различные свойства модели и, в частности, логические взаимоотношения между определениями. Кстати, данные аксиомы (см., например, переходы 8-8" в §3) могут выполнять роль не только подсказок для стратегий, но и спецификаций, ориентированных на пользователя. Рассмотрим следующий пример. Пусть дана Σ^+-программа:

схема

$P_1(x)$ опр (если $x = t$ то $P_3(x)$ иначе $P_4(x)$);

$P_2(x)$ опр $\varphi(x)$;

запрос ?. $P_2(t) \wedge P_1(t)$.(Подразумевается, что t – замкнутый терм.)

Чтобы исполнить данную программу, необходимо проверить истинность φ . Предположим, что в подразумеваемой в этом примере модели истинна формула $\forall x.(P_1(x) \to P_2(x))$. Очевидно, что знание этого факта (а оно может быть получено как некоторое содержатель-

ное математическое утверждение) существенно упрощает исполнение программы - достаточно проверить истинность предиката $P_3(t)$. Наличие в текстах программ подобных подсказок открывает широкие возможности для оптимизации семантических программ.

Большие надежды на ускорение исполнения Σ^+-программ возлагаются, как уже отмечалось во введении, на возможность их параллельного исполнения. Можно выделить следующие виды параллелизма:

$\vee$-параллелизм - обусловливается необходимостью в общем случае "исполнения" дизъюнктивных членов формул вида $(\varphi_1 \vee \ldots \vee \varphi_n)$; это исполнение можно осуществлять параллельно;

$\wedge$-параллелизм - возникающий вследствие того, что имеются конъюнктивные описания; при таком описании, скажем $(\varphi_1 \wedge \ldots \wedge \varphi_n)$, должны быть "исполнены" (в любом порядке) все конъюнктивные члены φ_i, $i = 1, \ldots, n$, и это, естественно, можно делать параллельно; заметим, что если при $\vee$-параллелизме информационных обменов между ветвями нет, то при $\wedge$-параллелизме есть точки синхронизации, соответствующие выполнению семантического аналога процедуры унификации термов;

"ограниченные" $\exists$- и $\forall$-параллелизмы - похожи на $\wedge$- и $\vee$-параллелизмы, но имеют свои особенности; эти виды параллелизма возникают из-за кванторов $\exists x \in t$, $\exists x \subseteq t$, $\forall x \in t$ и $\forall x \subseteq t$.

неограниченный $\exists$-параллелизм - возникающий из-за возможности "запускать" при проверке истинности формул вида $\exists x.\varphi$ неограниченное число ветвей: $\varphi(\mathrm{Prod}(1)), \ldots, \varphi(\mathrm{Prod}(n)), \ldots$;

термальный параллелизм - соответствующий возможности параллельно "исполнять" объектные (замкнутые) термы и, в частности, процедуры генерации элементов;

процедурный параллелизм - параллелизм, соответствующий возможности независимо осуществлять генерацию элементов модели и проверку истинности утверждений на порождаемых фрагментах модели;

модульный параллелизм - обусловливаемый возможностью при выполнении семантических программ параллельно исполнять составляющие эту программу модули.

Естественно, что каждый из этих видов параллелизма нуждается в тщательном теоретическом и экспериментальном исследовании. Заметим, что указанные выше возможности являются в некотором смысле неявными, т.е. выявляемыми только при дополнительном логическом анализе текстов программ. Поэтому естественно возникает желание иметь языковую конструкцию, позволяющую явным образом организовы-

вать и запускать процесс вычислений. Наличие такой(их) конструкции(й) дало бы возможность во многих случаях проще и естественнее моделировать, скажем, функционирование сложных динамических систем. Конечно, это можно сделать уже имеющимися средствами, но, как нетрудно понять, потребуются достаточно богатая исходная сигнатура и весьма сложные и громоздкие Σ^+-определения. А ведь такая конструкция в нашем языке уже имеется - это запрос $?\bar{y}.$ вход $\bar{x}.\varphi(\bar{x},\bar{y},\dots)$; нужно только разрешить использовать ее при построении Σ^+-определений и "внутри" запросов.

Содержательная интерпретация этой конструкции ясна: "запустить" процесс, вычисляющий все те $\bar{y}$, для которых на модели истинно φ". Заметим, что с денотационной точки зрения это уже, вообще говоря, не "первопорядковая" логическая конструкция. Так, например, если формула φ определяет некоторое Σ^+-множество, то в теоретико-модельном смысле $?y.$вход $x.\varphi(x,y)$ представляет собой уже (параметрическое) семейство Σ^+-предикатов.

Читатель, знакомый с языком DL динамической логики Ершова [I], легко узнает в конструкции $?\bar{x}.\varphi$ n-программу $[\bar{x}.\varphi]$. Таким образом, напрашивается естественное обобщение понятия Σ^+-программ до понятия n-программ Ершова с некоторыми ограничениями на употребление конструкции $[P,\alpha]$. В этом случае первое замечание, касающееся использования подсказок в Σ^+-программах, обобщается до замечания, указывающего на важность в качестве таких подсказок использовать DL-формулы. Что касается процедурной семантики n-программ, то она легко строится включением в процедурную семантику Σ^+-программ соответствующих правил оперирования с n-программами из [I]. Здесь хотелось бы сделать одно неформальное замечание.

Рассмотрим предикат $P(x,y)$ как некое устройство, связанное с "внешним миром" тремя "каналами": каналом x, каналом y и каналом истинностных значений. Своеобразие логической интерпретации предиката P заключается в том, что эти каналы являются неориентированными в том смысле, что каждый из них, в зависимости от ситуации, может считаться как входным, так и выходным. "Квантор" $?$ и уточняет такую ситуацию. Например, когда мы спрашиваем - истинен ли предикат P на паре $\langle c_1,c_2\rangle$ (т.е. $?.P(c_1,c_2)$ в наших обозначениях и $[x,y,P](c_1,c_2)$ - в обозначениях [I]), то тем самым мы неявно подразумеваем, что каналы x и y являются входными и на них поданы значения c_1 и c_2, а канал истинностных значений объявля-

ется выходным.Таким образом, можно сказать, что "квантор" ? как бы придает ориентацию тому каналу, на который он навешивается - с такого канала информацию можно только считывать. Подобное нефор - мальное пояснение конструкции ? указывает на потенциальную возможность построения в рамках семантического программирования теории "логических" (параллельных) процессов.

И, наконец, еще одно направление исследований связано с возможностью вместо языка Σ^*-формул в семантических программах использовать язык Σ-выражений Ершова [4]. Заметим, что в этом языке, наряду с отмеченными ранее видами параллелизма, появляется еще один - параллелизм, соответствующий возможности независимых редукций Σ-выражений и напоминающий параллелизм редукций λ-термов. Прежде всего, необходимо описать процедурную семантику такого языка. Об одном подходе к решению этой проблемы будет идти речь в одной из последующих публикаций.

Литература

I. ЕРШОВ Ю.Л. Динамическая логика над допустимыми множествами //Докл. АН СССР. - 1983. - Т.273, № 5. - С. 1045-1048.

2. Его же. Σ-определимость в допустимых множествах //Докл.АН СССР. - 1985. - Т.285, №4. - С. 792-793.

3. Его же. Σ-предикаты конечных типов над допустимыми множествами //Алгебра логика. - 1985. -Т.24, №5. - С. 563-602.

4. Его же. Язык Σ-выражений //Логические вопросы теории типов данных. - Новосибирск, 1986. - Вып. 114: Вычислительные системы. - С.3-10.

5. Его же. Σ-допустимые множества //Там же. - С. 35-39.

6. Его же. Об f_A-пространствах //Алгебра и логика. - 1986. -Т.25,№ 5.- С.532-543.

7. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. Σ-программирование.//Логико-математические проблемы МОЗ. - Новосибирск, 1985. Вып. 107: Вычислительные системы. - С. 3-29.

8. Их же. Математические основы семантического программирования //Докл. АН СССР. - 1986. - Т.289, № 6. - С. 1324-1328.

9. GONCHAROV S.S., SVIRIDENKO D.I. Theoretical aspects of Σ -programming// Lect.Notes of Comp.Science.- Berlin a.o., 1986.- V.215: Mathematical methods of specification and synthesis of software systems'85.- P.169-179.

10. GONCHAROV S.S., ERSHOV Yu.L., SVIRIDENKO D.I. Semantic programming// Proc.10th World Congress Information Processing 86, Dublin, Oct.1986.- Amsterdam a.o., 1986.- P.1093-1100.

11. СВИРИДЕНКО Д.И. Проектирование Σ-программ. Постановка проблемы //Методы анализа данных. - Новосибирск, 1985. - Вып.111: Вычислительные системы. - С.108-127.

I2. Его же. Проектирование Σ-программ. Σ-оцениваемость //Логические вопросы теории типов данных.-Новосибирск,1986.-Вып.114: Вычислительные системы. - С.59-83.

I3. ГОНЧАРОВ С.С. Замечания об аксиомах списочной надстройки //Там же. - С. II-I5.

I4. САЗОНОВ В.Ю., СВИРИДЕНКО Д.И. Денотационная семантика языка Σ-выражений. //Там же. - С. I6-34.

I5. СВИРИДЕНКО Д.И. Об одном варианте теории списочных надстроек //Прикладная логика. - Новосибирск, I986. - Вып.II6: Вычислительные системы. - С.67-79.

I6. СВИРИДЕНКО Д.И. Об одном обогащении языка Σ^+-программ //Настоящий сборник. - С. 97-I04.

I7. БОРЩЁВ В.Б. Семантика параметрических конструкций в логическом программировании //Настоящий сборник. - С. 3-I3.

I8. КРИЦКИЙ С.П., ХАНДРОС В.Л. Средства описания данных в языке спецификаций СЛЭНГ //Прикладная логика. - Новосибирск, I986. - Вып. II6: Вычислительные системы. - С. I0I-II9.

Поступила в ред.-изд.отд.
23 июня I987 года