ՀԱՅԱՍՏԱՆԻ ԳԻՏՈՒԹՅՈՒՆՆԵՐԻ ԱԶԳԱՅԻՆ ԱԿԱԴԵՄԻԱՅԻ ԶԵԿՈՒՅՑՆԵՐ
ДОКЛАДЫ НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК АРМЕНИИ

Том 95 1995 № 1

ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 519. 682.1

С. А. Нигиян

Логическая концепция языков программирования

(Представлено чл.-корр. НАН Армении Ю. Г. Шукуряном 23/VIII 1993)

В работе рассматривается логический подход к описанию языков программирования. При этом подходе логический язык программирования определяется как язык, программами которого являются формулы некоторой логики. К программе обращаются с вопросами. Интерпретатор логического языка программирования должен по программе и соответствующему ей вопросу решать: является ли вопрос логическим следствием программы или нет. Естественно, что интерпретатор не должен «лгать», т. е. если он не в состоянии ответить на поставленный вопрос, то должен либо остановиться с неопределенным ответом, либо функционировать бесконечно. В работе дано определение логического языка программирования и показано, что всякий функциональный язык программирования ([1]) может быть определен как логический язык программирования, т. е. определяется логическая семантика для функциональных языков программирования. В ([1,2]) показано, каким образом алгоритмический язык может быть определен как функциональный язык программирования. Объединяя эти два результата, получим, что всякий алгоритмический язык может быть определен как логический язык программирования, т. е. может быть задана логическая семантика для любого алгоритмического языка. Можно показать, что под предложенное в данной работе определение логического языка программирования подпадают такие языки логического программирования, как: Пролог ([3]), языки Σ-программирования ([4]), языки баз данных ([5]) и др.

1. **Формализация. Определение логического языка программирования.** Зафиксируем множество M, содержащее по крайней мере два элемента true и false. Введем понятие типа: 1) множество M есть тип; 2) любое непустое подмножество типа есть тип; 3) если β, α_1, ..., α_k $(k>0)$—типы, то множество всех отображений из α_1, ..., α_k в β (обозначим $\{\alpha_1 X \ldots X \alpha_k \to \beta\}$) есть тип; 4) других типов, кроме определенных согласно 1—3, нет.

Пусть C—множество констант, принадлежащих некоторым типам, X—множество переменных некоторых типов. Введем понятие терма. Каждому терму t сопоставим: a) множество переменных, от которых терм t зависит,—var t; b) тип терма t—type t; c) если var $t \subset \{y_1, \ldots,$

$y_n\}$, $\bar{y}_0=\langle y_1^0, \ldots, y_n^0\rangle$, где $y_i\in\alpha_i$, α_i — тип переменной y_i, то терму t сопоставим константу $val_{\bar{y}_0}(t)$, $i=1, \ldots, n$, $n>0$.

Определение терма:

1) любая переменная $x\in X$ есть терм, $var\ x=\{x\}$, $val_{\bar{y}_0}(x)=y_i^0$, если $x=y_i$ $(1\leqslant i\leqslant n)$ и, если x—переменная типа α, то $type\ x=\alpha$;

2) любая константа $c\in C$ или $c\in type\ x$, где $x\in X$, есть терм, $var\ c=\varnothing$, $val_{\bar{y}_0}(c)=c$, $type\ c=\{c\}$;

3) пусть $\tau, t_1, \ldots, t_k$—термы, $\beta, \alpha_1, \ldots, \alpha_k$—типы такие, что $type\ \tau\subset\{\alpha_1\times\ldots\times\alpha_k\to\beta\}$, $type\ t_i\subset\alpha_i$, $i=1, \ldots, k$, $k>0$, тогда $\tau(t_1, \ldots, t_k)$ есть терм и, если $t=\tau(t_1, \ldots, t_k)$, то $val_{\bar{y}_0}(t)=val_{\bar{y}_0}(\tau)(val_{\bar{y}_0}(t_1), \ldots, val_{\bar{y}_0}(t_k))$, $type\ t=\{val_{\langle y_1^0, \ldots, y_n^0\rangle}(t)\mid y_i^0\in type\ y_i,\ i=1, \ldots, n\}$;

4) пусть τ -терм, $x_1, \ldots, x_k\in X$, $k>0$ и, если $i\neq j$, то $x_i\neq x_j$ $(1\leqslant i, j\leqslant k)$, тогда $\lambda x_1\ldots x_k[\tau]$ есть терм и, если $t=\lambda x_1\ldots x_k[\tau]$, то $var\ t=var\ \tau\setminus\{x_1, \ldots, x_k\}$, $val_{\bar{y}_0}(t)\in\{type\ x_1\times\ldots\times type\ x_k\to type\ \tau\}$ и, если $var\ t=\{y_{i_1}, \ldots, y_{i_m}\}$ $(0\leqslant m\leqslant n)$, $y_0=\langle y_{i_1}^0, \ldots y_{i_m}^0\rangle$, то для всяких $x_j^0\in type\ x_j$, $j=1, \ldots, k$, $val_{\bar{y}_0}(t)(\bar{x}_0)=val_{\bar{x}_0, \bar{y}_0}(\tau)$, где $\bar{x}_0=\langle x_1^0, \ldots, x_k^0\rangle$, $type\ t$ определяется так же, как в пункте 3;

5) других термов, кроме определенных согласно 1—4, нет.

Всякий терм t такой, что $type\ t\subset\{true, false\}$, назовем элементарной формулой.

Определение формулы: 1) любая элементарная формула есть формула; 2) если A, B— формулы и $x\in X$, то $\neg(A)$, $(A)\ \&\ (B)$, $(A)\ \vee\ (B)$, $(A)\supset(B)$, $(A)\sim(B)$, $\forall x\ (A)$, $\exists x\ (A)$ есть формулы; 3) других формул, кроме определенных согласно 1—2, нет.

Некоторое вхождение переменной в формулу называется связанным, если оно находится в области действия λ или одного из кванторов $\forall$ или $\exists$. В противном случае это вхождение переменной называется свободным. Переменная x свободна в формуле A, если x имеет свободное вхождение в A. Пусть $A(\bar{x})$— формула, где $\bar{x}=\langle x_1, \ldots, x_n\rangle$— вектор различных переменных из X, $\bar{x}_0=\langle x_1^0, \ldots, x_n^0\rangle$, где $x_i^0\in type\ x_i$, $i=1, \ldots, n$, $n>0$. Через $A(\bar{x}_0)$ обозначим формулу, которая получена в результате подстановки в формулу $A(x)$ вместо каждого свободного вхождения переменной x_i значения x_i^0, $i=1, \ldots, n$. Интерпретация I формулы A определяется заданием значений всем ее свободным переменным.

Значение формулы A на интерпретации I (обозначим $I(A)$) определяется естественным образом. Формула A называется выполнимой, если существует такая интерпретация I, что $I(A)=true$. Формулу A будем называть тождественно истинной, если значение формулы A на любой интерпретации равно true. Если A, B—формулы и формула $(A)\supset(B)$ тождественно истинна, то будем говорить, что формула B является логическим следствием формулы A и обозначать $A\models B$.

Логический язык программирования определяется заданием шестерки M, C, X, S, δ, U, где

M — непустое множество, содержащее по крайней мере два элемента true и false;

C — множество констант, принадлежащих некоторым типам;

X — множество переменных некоторых типов;

S — множество программ, представляющее собой некоторое множество выполнимых формул, построенных с использованием C и X;

δ—отображение, которое каждой программе $P\in S$ сопоставляет

множество формул $\delta(P)$, называемых вопросами, соответствующими программе P;

U—интерпретатор, представляющий собой алгоритм, который для каждой программы $P \in S$ и вопроса $Q \in \delta(P)$ либо останавливается с положительным ответом, либо — с отрицательным ответом, либо — с неопределенным ответом, либо функционирует бесконечно. Положительный ответ может быть как со значением, так и без него. Если U останавливается на P и Q, то результат применения U к P и Q обозначим $U(P, Q)$. Интерпретатор U должен обладать следующими свойствами:

1) если U останавливается на P и Q с положительным ответом без значения, т. е. $U(P, Q) =$ да, то $P \models Q$;

2) если U останавливается на P и Q с отрицательным ответом, т. е. $U(P, Q) =$ нет, то $P \not\models Q$;

3) если U останавливается на P и Q с положительным ответом со значением, т. е. $U(P, Q) = \bar{y}_0$, где $\bar{y}_0$ — вектор значений вектора переменных $\bar{y}$, то Q имеет вид $\exists \bar{y} A(\bar{y})$ и $P \models A(\bar{y}_0)$.

Интерпретатор U назовем логически полным, если из того, что $P \models Q$, следует, что U останавливается на P и Q с положительным ответом.

2. Функциональная концепция языков программирования с точки зрения логической. Рассмотрим произвольный функциональный язык программирования (см. [1]), определяемый четверкой M_1, C_1, X_1, T_1, где M_1 частично упорядоченное множество, содержащее неопределенный элемент $\perp$, который является наименьшим элементом множества M_1, и каждый элемент из M_1 сравним только с $\perp$ и с самим собой; C_1 — некоторое множество констант, принадлежащих монотонным типам; X_1 — некоторое множество переменных монотонных типов; T_1 — некоторое подмножество термов, построенных с использованием C_1 и X_1.

Определим монотонный тип: 1) множество M_1 есть монотонный тип; 2) если $\beta, \alpha_1, \ldots, \alpha_k$ $(k>0)$ — монотонные типы, то множество всех монотонных отображений из $\alpha_1 \times \ldots \times \alpha_k$ в β (обозначим $[\alpha_1 \times \ldots \times \alpha_k \to \beta]$) есть монотонный тип; 3) других монотонных типов, кроме определенных согласно 1—2, нет.

Опишем шестерку M, C, X, S, δ, U, определяющую функциональный язык программирования как логический язык программирования:

$M = M_1 \cup \{\text{true, false}\}$.

$C = C_1 \cup \{=\}$, где $= \in [M_1^2 \to \{\text{true, false}\}]$ и для любых $m, m' \in M_1$ значение $=(m, m')$ есть true, если m совпадает с m', и false в противном случае.

$X = X_1 \cup \{y\}$, где y — переменная типа M_1.

S есть множество, состоящее из программ P вида

$$(F_1 = \tau_1) \& \ldots \& (F_n = \tau_n), \tag{1}$$

где $F_i \in X_1$, $F_i \neq F_j$, если $i \neq j$, $\tau_i \in T_1$, type $\tau_i =$ type F_i, var $\tau_i \subseteq \{F_1, \ldots, F_n\}$, type $F_i = [M_1^k \to M_1]$ $(k \geqslant 1)$, $i, j = 1, \ldots, n$, $n > 0$.

Формула (1) будет выполнимой, так как система уравнений

$$\begin{cases} F_1 = \tau_1 \\ \vdots \\ F_n = \tau_n \end{cases} \tag{2}$$

имеет решение. Решением системы (2) мы называем вектор $\bar{f} = \langle f_1, \ldots, f_n \rangle$, такой, что $f_i \in$ type F_i и $\text{val}_{\bar{f}}(\tau_i) = f_i$, $i = 1, \ldots, n$. Из [1] следует, что система (2) имеет наименьшее решение. Если $\bar{f}$ — наименьшее решение системы (2), то через f_P условимся обозначать функцию f_1, яв-

ляющуюся семантикой функциональной программы (2).

Пусть P программа вида (1), тогда

$$\delta(P)=\{\exists y F_1(\bar{x}_0)=y \ \& \ y\neq\perp) \mid \bar{x}_0=\langle x_1^0,\ldots,x_k^0\rangle,\ x_i^0\in M,\ i=1,\ldots,k\}$$

Определим интерпретатор U. Пусть P программа вида (1), $Q\in\delta(P)$ и имеет вид $\exists y(F_1(\bar{x}_0)=y \ \& \ y\neq\perp)$. Тогда, если $f_P(\bar{x}_0)=y_0\neq\perp$, то $U(P, Q)=y_0$, если же $f_P(\bar{x}_0)=\perp$, то U на P и Q либо функционирует бесконечно, либо U(P, Q)=нет.

Пусть P—программа вида (1), $Q\in\delta(P)$ и имеет вид $\exists y(F_1(\bar{x}_0)=y \ \& \ y\neq\perp)$, тогда имеют место теоремы 1 и 2.

Теорема 1.

a) $P\models Q\Leftrightarrow$ существует единственное $y_0\in M$, такое, что $P\vdash F_1(\bar{x}_0)=y_0 \ \& \ y_0\neq\perp$;

b) $f_P(\bar{x}_0)=y_0$ и $y_0\neq\perp\Leftrightarrow P\models F_1(\bar{x}_0)=y_0 \ \& \ y_0\neq\perp$;

c) $P\models Q\Leftrightarrow f_P(\bar{x}_0)\neq\perp$.

Теорема 2.

a) Если $U(P, Q)=y_0\in M_1$, то $P\models F_1(\bar{x}_0)=y_0 \ \& \ y_0\neq\perp$;

b) если $U(P, Q)=$нет, то $P\not\models Q$;

c) если $P\models Q$, то $U(P, Q)=f_P(\bar{x}_0)\neq\perp$.

Следствие теоремы 2. Интерпретатор U удовлетворяет условиям, налагаемым на интерпретатор логического языка программирования, и является логически полным.

Ереванский государственный университет

Ս. Ա. ՆԻԳԻՅԱՆ

Ծրագրավորման լեզուների տրամաբանական կոնցեպցիա

Տրված է ծրագրավորման տրամաբանական լեզվի և նրա սեմանտիկայի սահմանում։ Ապացուցված է, որ յուրաքանչյուր ծրագրավորման ֆունկցիոնալ լեզու կարող է սահմանվել որպես ծրագրավորման տրամաբանական լեզու, այսինքն որոշվում է տրամաբանական սեմանտիկա ֆունկցիոնալ ծրագրավորման լեզուների համար։ Ստացված արդյունքը տալիս է ալգորիթմական լեզուների տրամաբանական սեմանտիկայի նկարագրման միջոց, քանի որ յուրաքանչյուր ալգորիթմական լեզու կարող է սահմանվել որպես ծրագրավորման ֆունկցիոնալ լեզու։

ЛИТЕРАТУРА—ԳՐԱԿԱՆՈՒԹՅՈՒՆ

[1] С. А. Нигиян. Программирование. № 5. с. 77—86. 1991. [2] С. А. Нигиян. Программирование. № 2. с. 58—68. 1993. [3] У. Клоксин, К. Меллиш. Программирование на языке Пролог. М., Мир. 1987. [4] С. С. Гончаров, Д. И. Свириденко. Логико-математические проблемы МОЗ. Новосибирск, вып. 107. Вычислительные системы. с. 3—29, 1985. [5] Д. Мейер. Теория реляционных баз данных. М., Мир. 1987.