УДК 519.68:510

С. А. Нигиян, Л. О. Хачоян, А. В. Нигиян

К логической трактовке процедурного программирования

Представлено академиком Н.У. Аракеляном 2/XI 2006)

Ключевые слова: *процедурный, логический, язык, программирование, интерпретатор*

В работе [1] показано, что всякий функциональный язык программирования может быть определён как логический язык программирования. В данной работе сделано аналогичное для процедурных языков программирования, т. е. показано, каким образом на процедурный язык программирования можно "посмотреть" как на логический язык программирования. В связи с этим вводится понятие логического языка программирования, основанного на логике предикатов первого порядка. Идея построения логической формулы по процедурной программе берётся нами из работ [2-3]. Доказывается, что всякая процедурная программа (если "посмотреть" на неё как на логическую формулу) имеет наименьшую модель. Доказывается также, что интерпретатор процедурного языка программирования (если "посмотреть" на него как на логический язык программирования) является логически полным и непротиворечивым.

1. Используемые понятия и обозначения. Зафиксируем четыре непересекающихся множества M, V, P, BP, BF, где: M - непустое множество предметов; V - множество предметных переменных; P - множество предикатных символов, с приписанной каждому символу местностью; BP - множество встроенных предикатов, т.е. отображений вида $M^k \to$ {true, false}, $k \geq 1$; BF - множество встроенных функций, т.е. отображений вида $M^k \to M$, $k \geq 1$.

Из элементов множеств M, V, BF строятся термы: 1) каждый элемент из M есть терм; 2) каждая переменная из V есть терм; 3) если $t_1, \ldots t_k$ ($k \geq 1$) - термы и f - k-местная встроенная функция из BF, то $f(t_1, \ldots, t_k)$ есть терм; 4) других термов нет.

Атом определяется традиционным образом: 1) каждый 0-местный символ из P есть атом; 2) если $t_1, \ldots t_k$ ($k \geq 1$) - термы и p - k-местный символ из $P \cup BP$, то $p(t_1, \ldots, t_k)$ есть атом; 3) других атомов нет.

Традиционным образом определим формулу логики предикатов первого порядка: 1) каждый атом есть формула; 2) если A, B - формулы и x - переменная из V, то $(\neg A)$, $(A \& B)$, $(A \vee B)$, $(\forall xA)$, $(\exists xA)$ - формулы; 3) других формул нет.

Опишем рассматриваемые нами интерпретации. Предметным множеством рассматриваемых интерпретаций будет множество M. Каждому 0-местному символу из P сопоставляется один из элементов множества {true, false}, а каждому k-местному символу из P сопоставляется некоторое отображение: $M^k \to$ {true, false}. Обозначим описанное множество интерпретаций через $\mathrm{Int}(M)$. Отметим, что интерпретации из $\mathrm{Int}(M)$ могут отличаться одна от другой лишь значениями, сопоставляемыми символам множества P. Поэтому каждую интерпретацию $I \in \mathrm{Int}(M)$ можно отождествить с подмножеством атомов

вида q, где q - 0-местный символ из P, и $p(m_1, \ldots, m_k)$, где p - k-местный ($k > 0$) символ из P, $m_1, \ldots, m_k \in M$, значения которых на интерпретации I есть true. Легко видеть, что множество $\mathrm{Int}(M)$ будет полной решёткой, если в качестве частичного порядка на $\mathrm{Int}(M)$ взять отношение включения.

Пусть A замкнутая формула и I интерпретация из $\mathrm{Int}(M)$. Через $I(A)$ условимся обозначать значение формулы A на интерпретации I. Если $I(A)$ = true, то интерпретация I называется моделью формулы A. Формула A называется выполнимой, если для неё существует модель. Пусть A и B замкнутые формулы. Будем говорить, что формула B логически следует из формулы A, и обозначать это $A \models B$, если любая интерпретация из $\mathrm{Int}(M)$ является моделью формулы $A \supset B$.

2. Определение логического языка программирования. Логический язык программирования, основанный на логике предикатов первого порядка, определяется заданием восьмёрки M, V, P, BP, BF, $Prog$, Δ, U, первые пять компонент которой уже определены. Определим последние три: $Prog$ - множество программ, представляющее собой некоторое множество выполнимых формул; Δ - отображение, которое каждой программе $P \in Prog$ сопоставляет множество формул $\Delta(P)$, называемых множеством запросов, соответствующих программе P; U - интерпретатор, представляющий собой алгоритм, который для каждой программы $P \in Prog$ и запроса $Q \in \Delta(P)$ либо останавливается с положительным ответом, либо - с отрицательным ответом, либо функционирует бесконечно. Если U останавливается на P и Q, то результат применения U к P и Q обозначим $U(P,Q)$. Интерпретатор U должен быть логически непротиворечивым, т.е. обладать следующими свойствами:

если U останавливается на P и Q с отрицательным ответом, т. е. $U(P,Q)$ = no, то $P \not\models Q$;

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

если U останавливается на P и Q с положительным ответом со значением, т. е. $U(P,Q) = \langle m_1, \ldots, m_k \rangle$, где $m_1, \ldots, m_k \in M$, $k \geq 1$, то запрос Q имеет вид $\exists x_1 \ldots \exists x_k A(x_1, \ldots, x_k)$ и $P \models A(m_1, \ldots, m_k)$.

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

3. Процедурные языки программирования с логической точки зрения. Дадим несколько упрощённое, не меняющее сути дела определение процедурного языка прграммирования. Зададимся четырьмя непустыми непересекающимися множествами M, V, BP, BF, которые имеют описанный в п.1 смысл. $V = X \cup Y \cup \{z\}$, где X - множество входных переменных, Y - множество рабочих переменных, z - выходная переменная. Конструкцию вида $u = \tau$, где $u \in Y \cup \{z\}$, τ - терм, имеющий вид $f(t_1, \ldots, t_k)$, v или m, где f - k-местная функция из BF, $t_i \in M \cup X \cup Y$, $i = 1, \ldots, k$, $k > 0$, $v \in X \cup Y$, $m \in M$, назовём элементарным оператором, а переменную u его левой частью. Конечное множество элементарных операторов с различными левыми частями назовём оператором. Атом вида $p(t_1,\ldots,t_k)$ или его отрицание $\neg p(t_1, \ldots, t_k)$, где p - k-местный символ из BP, $t_i \in M \cup X \cup Y$, $i = 1, \ldots, k$, $k > 0$, назовём условием.

Программа процедурного языка в графовой форме (г.программа) представляет собой конечный ориентированный граф с одной выделенной вершиной - входом и одной выделенной вершиной - выходом. Из выхода не исходит ни одной дуги, во вход не входит

ни одна дуга. Из каждой вершины, отличной от выхода, исходит либо одна дуга, либо - две: если из вершины исходит одна дуга, то она помечена только оператором; если две, то каждая из них помечена как оператором, так и условием, причём, если одна из них помечена условием $p(t_1, \ldots, t_k)$, то другая условием $\neg p(t_1, \ldots, t_k)$. Оператор, сопоставляемый дуге, ведущей в выход, имеет вид z = τ, дуге, не ведущей в выход, - $\{u_1 = \tau_1, \ldots, u_d = \tau_d\}$, $u_i \in Y$, i = 1, …, d, d > 0. Пусть $\bar{x} = \langle x_1, \ldots, x_k \rangle$ - вектор входных переменных г.программы P_g и $\bar{x}_0 = \langle x_1^0, \ldots, x_k^0 \rangle$, где $x_i^0 \in M$, i = 1, …, k, k > 0. Выполнение г.программы P_g на $\bar{x}_0$ начинается со входа и определяется естественным образом. Выполнение P_g на $\bar{x}_0$ может быть как конечным, так и бесконечным. В случае конечного выполнения процесс завершается в выходе и результатом является полученное значение выходной переменной z.

Опишем восьмёрку *M*, *V*, *P*, *BP*, *BF*, *Prog*, Δ, *U*, определяющую процедурный язык как логический язык программирования. Здесь множества *M*, *V*, *BP*, *BF* являются соответствующими множествами процедурного языка; *P* - множество предикатных символов с приписанной каждому символу местностью, причём для любого k > 0 *P* содержит счётное число символов местности k. Множество программ *Prog* получим из множества г.программ процедурного языка, строя по каждой г.программе P_g логическую программу $P \in Prog$. Опишем алгоритм построения.

Пусть P_g - г.программа, $\bar{x} = \langle x_1, \ldots, x_k \rangle$ - вектор её входных переменных, $\bar{y} = \langle y_1, \ldots, y_h \rangle$ - вектор её рабочих переменных. Пусть P_g имеет n + 2 (n ≥ 0) вершины. Перенумеруем их числами от 0 до n + 1 следующим образом: входу сопоставим 0, выходу - n + 1, остальные вершины перенумеруем произвольным образом. Сопоставим каждой дуге (r, s) г.программы P_g формулу $A_{(r,s)}$, построенную из *M*, *V*, *P*, *BP*, *BF*, 0 ≤ r, s ≤ n + 1. Пусть q - k + 1-местный, а $q_1, \ldots, q_n$ - k + h-местные предикатные символы из *P*. Введём обозначение: если дуге (r, s), где 0 ≤ r, s ≤ n, г.программы P_g сопоставлен оператор $\{u_1 = \tau_1, \ldots, u_d = \tau_d\}$, $u_i \in Y$, i = 1, …, d, d > 0, то терм, полученный в результате подстановки в $q_s(\bar{x}, \bar{y})$ вместо каждой переменной u_i терма τ_i (1 ≤ i ≤ d), обозначим $q_s(\bar{x}, \bar{\tau}_{(r,s)})$. Определим $A_{(r,s)}$, 0 ≤ r, s ≤ n + 1. $A_{(r,s)}$ будет равна одной из следующих формул:

$q(\bar{x}, \tau)$, если r = 0, s = n + 1, из входа исходит одна дуга и она помечена оператором z = τ;

$\pi \supset q(\bar{x}, \tau)$, если r = 0, s = n + 1, из входа исходят две дуги и дуга (0, n + 1) помечена условием π и оператором z = τ;

$q_s(\bar{x}, \bar{\tau}_{(r,s)})$, если r = 0, s ≠ n + 1, из входа исходит одна дуга;

$\pi \supset q_s(\bar{x}, \bar{\tau}_{(r,s)})$, если r = 0, s ≠ n + 1, из входа исходят две дуги и дуга (0, s) помечена условием π;

$q_r(\bar{x}, \bar{y}) \supset q(\bar{x}, \tau)$, если r ≠ 0, s = n + 1, из вершины r исходит одна дуга и она помечена оператором z = τ;

$q_r(\bar{x}, \bar{y}) \,\&\, \pi \supset q(\bar{x}, \tau)$, если r ≠ 0, s = n + 1, из вершины r исходят две дуги и дуга (r, n + 1) помечена оператором z = τ и условием π;

$q_r(\bar{x}, \bar{y}) \supset q_s(\bar{x}, \bar{\tau}_{(r,s)})$, если r ≠ 0, s ≠ n + 1, из вершины r исходит одна дуга;

$q_r(\bar{x},\bar{y})\&\pi \supset q_s(\bar{x},\bar{\tau}_{(r,s)})$, если r ≠ 0, s ≠ n + 1, из вершины r исходят две дуги и дуга (r, s) помечена условием π.

Логическая программа *P* будет иметь вид

$$\forall\bar{x}\forall\bar{y}A(\bar{x},\bar{y}), \qquad (1)$$

где $A(\bar{x},\bar{y})$ - конъюнкция формул $A_{(r,s)}$, сопоставленных всем дугам г.программы P_g.

Имея программу $P \in Prog$, можно всегда восстановить г.программу P_g, из которой получена программа *P*, т.к. каждый член $A_{(r,s)}$ конъюнкции $A(\bar{x},\bar{y})$ есть изображение дуги (r,s) г.программы P_g в виде логической формулы.

Теорема 1. *Всякая программа $P \in Prog$ имеет наименьшую модель.*

Доказательство. Пусть программа *P* имеет вид (1). Построим интерпретацию I_0, такую, что $I_0(P)$ = true. Каждому из предикатных символов q, q_1, …, q_n сопоставим отображения следующим образом: пусть $\bar{x}_0 = \langle x_1^0, \ldots, x_k^0\rangle$, где $x_i^0 \in M$, i = 1, …, k, $\bar{y}_0 = \langle y_1^0, \ldots, y_h^0\rangle$, где $y_j^0 \in M$, j = 1, …, h, тогда $I_0(q_i(\bar{x}_0,\bar{y}_0))$, 1 ≤ i ≤ n, (соответственно $I_0(q(\bar{x}_0,z_0))$) равно true, если при выполнении г.программы P_g на $\bar{x}_0$ на некотором шаге функционирования мы попадаем в вершину с номером i (соответственно в выход) со значением рабочих переменных $\bar{y}$ равным $\bar{y}_0$ (соответственно со значением переменной z равным z_0) и false в противном случае. Можно показать, что I_0 будет наименьшей моделью программы *P*.

Теорема 1 доказана.

Пусть $P \in Prog$ и имеет вид (1). Определим множество запросов Δ(*P*), соответствующих программе *P*:

$$\Delta(P) = \{\exists z q(\bar{x}_0, z) \mid \bar{x}_0 = \langle x_1^0, \ldots, x_k^0\rangle, \text{ где } x_i^0 \in M,\ i = 1, \ldots, k\}.$$

Теорема 2. *Пусть $P \in Prog$, $Q \in \Delta(P)$ и имеет вид $\exists z q(\bar{x}_0, z)$, тогда:*

a) $P \models Q \Leftrightarrow$ существует единственное $z_0 \in M$ такое, что $P \models q(\bar{x}_0, z_0)$;

b) выполнение P_g на $\bar{x}_0$ конечно и результат равен $z_0 \in M \Leftrightarrow P \models q(\bar{x}_0, z_0)$;

c) $P \models Q \Leftrightarrow$ выполнение P_g на $\bar{x}_0$ конечно.

Доказательство. a) Пусть $P \models \exists z\ q(\bar{x}_0, z)$, тогда выполнение г.программы P_g на $\bar{x}_0$ конечно, так как в противном случае для любого $z_0 \in M$, $I_0(q(\bar{x}_0,z_0))$ = false, но $I_0(P)$ = true, где I_0 - наименьшая модель программы *P*. Пусть выполнение P_g на $\bar{x}_0$ завершается с результатом $z_0 \in M$. Покажем, что $P \models q(\bar{x}_0, z_0)$. Пусть *I* произвольная модель программы *P*, тогда $I(q(\bar{x}_0,z_0))$ должно равняться true, так как выполнение P_g на $\bar{x}_0$ конечно и результат равен z_0. Пусть для некоторого $z_1 \neq z_0$ $P \models q(\bar{x}_0, z_1)$. Но этого не может быть, так как $I_0(P)$ = true, а $I_0(q(\bar{x}_0,z_1))$

= false.

Если $P \models q(\bar{x}_0, z_0)$ для некоторого $z_0 \in M$ то очевидно, что $P \models Q$.

b) Пусть выполнение P_g на $\bar{x}_0$ конечно и результат равен $z_0 \in M$, тогда, используя доказательство из пункта (a), получим, что $P \models q(\bar{x}_0, z_0)$.

Пусть $P \models q(\bar{x}_0, z_0)$ для некоторого $z_0 \in M$, тогда из того, что $I_0(P)$ = true, будет следовать, что $I_0(q(\bar{x}_0, z_0))$ = true, а это означает, что выполнение P_g на $\bar{x}_0$ конечно и результат равен z_0.

c) Доказательство этого пункта следует из пунктов (a) и (b).

Теорема 2 доказана.

Определим интерпретатор U. Пусть $P \in Prog$, $Q \in \Delta(P)$ и имеет вид $\exists z q(\bar{x}_0, z)$, тогда U на P и Q останавливается с положительным ответом со значением $z_0 \in M$, если выполнение P_g на $\bar{x}_0$ конечно и результат равен z_0; U на P и Q функционирует бесконечно, если выполнение P_g на $\bar{x}_0$ бесконечно.

Теорема 3. *Пусть $P \in Prog$, $Q \in \Delta(P)$ и имеет вид $\exists z\, q(\bar{x}_0, z)$, тогда:*

a) если $U(P,Q) = z_0 \in M$, то $P \models q(\bar{x}_0, z_0)$;

b) если $P \models Q$, то $U(P,Q) = z_0 \in M$.

Доказательство следует из теоремы 2 и свойств интерпретатора U.

Следствие теоремы 3. Интерпретатор U является логически полным и непротиворечивым.

Ереванский государственный университет

Литература

1. *Нигиян С. А.* - ДНАН Армении. 1995. Т. 95. N1. С. 26-29.

2. *Manna Z.* - Journal of Computer and System Sciences. 1969. N2. P.119-127. (рус. пер. *Манна З.* - Кибернетический сборник (нов. сер.). 1970. Вып. 7. С. 85-93).

3. *Chang Ch., Lee R.* Symbolic Logic and Mechanical Theorem Proving. Academic Press. Inc. 1973. (рус. пер. *Чень Ч., Ли Р.* Математическая логика и автоматическое доказательство теорем. М. Наука. 1983. 358 с.).

Ս. Ա. Նիգիյան, Լ. Օ. Խաչոյան, Ա. Վ. Նիգիյան

Պրոցեդուրային ծրագրավորման տրամաբանական մեկնաբանության մասին

Աշխատանքում ցույց է տրված, թե ինչպես պրոցեդուրային ծրագրավորման լեզուն կարող է սահմանվել որպես տրամաբանական ծրագրավորման լեզու: Այդ նպատակով մտցվում է առաջին կարգի պրեդիկատների տրամաբանության վրա հիմնված տրամաբանական ծրագրավորման լեզվի գաղափարը: Ապացուցվում է, որ կամայական պրոցեդուրային ծրագիրը (եթե այն դիտարկել որպես տրամաբանական բանաձև) ունի փոքրագույն մոդել: Ապացուցվում է նաև, որ պրոցեդուրային ծրագրավորման լեզվի ինտերպրետատորը (եթե այն դիտարկել որպես տրամաբանական ծրագրավորման լեզու) տրամաբանորեն լրիվ է և անհակասելի:

S. A. Nigiyan, L. O. Khachoyan, A. V. Nigiyan

On Logical Interpretation of Procedural Programming

In this paper it is shown how the procedural programming language can be defined as the logical programming language. For this purpose the notion of the logical programming language based on the first-order predicate logic is introduced. It is proved that every procedural program (if it is regarded as the logical formula) has the least model. It is also proved that the interpreter of the procedural programming language (if it is regarded as the logical programming language) is logically complete and consistent.