УДК 004.432.42

СЕМАНТИКА ТИПОВ ДАННЫХ ФУНКЦИОНАЛЬНО-ПОТОКОВОГО ЯЗЫКА ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ ПИФАГОР

***Мария Сергеевна Ушакова**, аспирант*
Тел.: 8 902 941 7077, e-mail: ksv@akadem.ru,
Сибирский федеральный университет,
Институт космических и информационных технологий
http://www.sfu-kras.ru

Работа посвящена описанию семантики типов данных для функционально-потокового языка параллельного программирования Пифагор. Рассматривается два вида семантики: математическая семантика на основе неформальной теории множеств и аксиоматическая семантика.

Ключевые слова: аксиоматическая семантика типов данных, функционально-потоковое параллельное программирование, язык программирования Пифагор.

В настоящее время наблюдается тенденция к усложнению программного обеспечения и увеличению сложности его отладки и рисков в случае сбоя. В результате, стали активно развиваться методы верификации программ, в частности формальная верификация. Под формальной верификацией понимается доказательство корректности программы, которое заключается в установлении соответствия между программой и её спецификацией, описывающей цель разработки [1]. Развитие методов формальной верификации особенно актуально для параллельного программирования. Однако методы формальной верификации достаточно трудоёмки, и их сложность существенно возрастает при верификации параллельных императивных программ.

М.С. Ушакова

Альтернативным вариантом императивного подхода является функционально-потоковая параллельная парадигма. Модель функционально-потоковых параллельных вычислений и реализующий эту модель язык программирования Пифагор описаны в [2]. Данная модель обеспечивает формирование параллелизма программ на уровне операций, что позволяет упростить процесс верификации, так как не требует анализа возникающих в традиционных архитектурах дополнительных ресурсных конфликтов. Поэтому развитие методов формальной верификации функционально-потоковых параллельных программ является актуальным.

Одной из наиболее важных и сложных задач при формальной верификации является описание спецификации программы. Для этого требуется специальный язык спецификаций, на котором формулируются требования к программе. Целью данной работы является формализация семантики типов данных языка Пифагор, для определения языка спецификаций. Формализация семантики типов данных языка программирования позволяет однозначно определять смысл программ и язык спецификаций свойств программ.

Описание семантики встроенных функций языка Пифагор на естественном языке приведено в [2]. В языке Пифагор есть следующие типы данных: int (целый), float (действительный), bool (логический), char (символьный), func (функция), error (ошибочный), spec (спецзнаковый), signal (сигнал), datalist (список данных), parlist (параллельный список), delaylist(задержанный список). В работе рассматривается математическая и аксиоматическая семантика типов данных языка Пифагор.

1. Математическая семантика типов данных

Математический подход к семантике типов данных – это описание типов на осно-

ве неформальной теории множеств. Согласно [3] семантика всякого конкретного типа с точки зрения неформальной теории множеств – это множество значений этого типа вместе с совокупностью частичных или тотальных операций, в которых эти значения могут быть аргументами или результатами, и совокупность отношений, в которых эти значения могут участвовать. Семантика родового типа – это (частичная) функция, которая по конкретным типам строит новый тип, т.е. по множествам значений с допустимыми операциями и отношениями строит «новое» множество значений со своими операциями и отношениями, в которых могут использоваться эти «новые» значения.

Обозначим через PO множество всех типов данных языка Пифагор. Для определения математической семантики типов языка Пифагор определим множества, которые назовём «простыми (конкретными) типами», и операции на них.

1. Множество целых чисел int, элементами которого являются целые числа из **Z**. На этом множестве определены: унарная операция «получение противоположного элемента» -, двуместные тотальные операции «сложение» +, «вычитание» –, «умножение» ×, частичная функция «деление нацело» /, двуместные отношения «равно» =, «меньше» <, «больше» >, «не больше» ≤, «не меньше» ≥. Функции и отношения имеют такой же смысл, что и в математике.

2. Множество действительных чисел float. Элементы этого множества – числа из **R**, с такими же операциями и отношениями, как для типа int, имеющими обычный математический смысл.

3. Булевское множество $\text{bool} = \{\text{true}, \text{false}\}$. На множестве определены бинарные операции «конъюнкция» $\wedge$, «дизъюнкция» $\vee$, и унарная операция «отрицание» $\neg$. Семантика операций такая же, как и в булевой алгебре.

4. Конечное множество символов char. На множестве определена унарная функция «получения кода символа» $\text{encode}: \text{char} \to \text{int}$, которая сопоставляет каждому символу некоторое целое число. Также есть обратная функция $\text{encode}^{-1}: \text{int} \to \text{char}$.

5. Конечное множество имён функций func, элементами которого являются имена встроенных и пользовательских функций.

6. Конечное множество констант ошибок error={BASEFUNCERROR, BOUNDERROR, INTERROR, REALERROR, ZERRODIVIDE, TYPEERROR}.

7. Конечное множество специальных знаков (разделителей) spec={$+_c$, $-_c$, $/_c$, $*_c$, $<_c$, $>_c$, $=_c$, $>=_c$, $<=_c$, $<>_c$, $=>_c$, $->_c$, $<-_c$, $()_c$, $\{\}_c$, $[]_c$, $|_c$, $\#_c$, $\%_c$, $.c$, $.?_c$}. Индекс «с» введён для того, чтобы не путать константы из множества spec и идентификаторы функций.

8. Множество, содержащее один элемент signal={$\cdot$}.

9. Бесконечное множество задержанных списков delaylist – множество констант, каждая из которых соответствует допустимому подграфу программы на языке Пифагор.

10. Конечное множество констант (имён) типов type={int_c, float_c, bool_c, char_c, func_c, error_c, signal_c, datalist_c, parlist_c, delaylist_c}. Индекс «с» введён для того, чтобы не путать имена типов с константами из type.

Кроме того, на каждом рассмотренном множестве S введено одноместное отношение «быть значением типа S» $\in S$. Например, для типа int – одноместное отношение «быть значением типа int» $\in \text{int}$, для типа float – $\in \text{float}$ и т.д.

На основе «простых типов» определим конкретные типы данных языка Пифагор. Введём отображение f, которое сопоставляет каждому элементу множества type одно из множеств «простого типа», с таким же именем, как и имя элемента. Тогда тип POsimple ($\text{POsimple} \subset \text{PO}$) – это множество пар вида: $< t, a >$, где $t \in \text{type}$, $a \in \text{f}(t)$ – элемент одного из множеств «простого типа», соответствующего t. Угловые скобки здесь и далее будут использоваться для обозначения кортежа. Разобьём тип POsimple на непересекающиеся подмножества, в зависимости от первого элемента пары t. Каждому подмножеству дадим имя, совпадающее с именем константы t, у которого индекс «с» заменя-

ется на «p». Каждое такое подмножество будет конкретным типом языка Пифагор: int_p, float_p, bool_p, char_p, func_p, error_p, signal_p, $\text{delaylist}_\text{p}$, spec_p.

Определим семантику родового типа «список данных» datalist_p. Будем представлять список в виде декартова произведения конечного числа типов языка Пифагор. Определим математическую семантику «простого (родового) типа», который обозначим $\text{datalist}\prod_{i=1}^{n}\text{T}_i$. Значения этого типа – кортежи длины n, $n \in \mathbf{N}$:

$$\prod_{i=1}^{n}\text{T}_i = \{< t_1, t_2, \ldots, t_n >: t_i \in \text{T}_i\}, \text{ где } \text{T}_\text{i} \subset (\text{PO}).$$

Приведём примеры элементов типа $\text{datalist}\prod_{i=1}^{n}\text{T}_i$: пустой список $< \cdot > \in \text{signal}_\text{p}$, список с одним элементом $< t > \in \text{T}$, $\text{T} \subset \text{PO}$, список из двух целых чисел $< t_1, t_2 > \in \text{int}_\text{p} \times \text{int}_\text{p}$, список из трёх целых чисел $< t_1, t_2, t_3 > \in \text{int}_\text{p} \times \text{int}_\text{p} \times \text{int}_\text{p}$.

На множестве $\text{datalist}\prod_{i=1}^{n}\text{T}_i$ определена одноместная операции «нахождение длины списка» $\text{len} : \text{datalist}\prod_{i=1}^{n}\text{T}_i \to \mathbf{N}$, которая возвращает длину кортежа и формально определяется с помощью λ-нотации следующим образом: $\lambda\, x \in \text{datalist}\prod_{i=1}^{n}\text{T}_i.(n)$. Двуместная частичная операция «выбор элемента списка» $\text{select} : \left(\text{datalist}\prod_{i=1}^{n}\text{T}_i\right) \times \mathbf{N} \to \text{T}_i$, формально определяется $\lambda\, x \in \text{datalist}\prod_{i=1}^{n}\text{T}_i.\, \lambda\, j \in \mathbf{N}.(T_j)$.

Двуместная операция «добавление элемента в конец списка» $\text{add} : \left(\text{datalist}\prod_{i=1}^{n}\text{T}_i\right) \times \text{T}_{n+1} \to \text{datalist}\prod_{i=1}^{n+1}\text{T}_i$, которая формально определяется $\lambda < t_1, t_2, \ldots, t_n > \in \text{datalist}\prod_{i=1}^{n}\text{T}_i.\, \lambda\, t \in \text{PO}.(< t_1, t_2, \ldots, t_n, t >)$. Далее будем использовать выражение $\text{«createList}(a_1, a_2, \ldots, a_n)\text{»}$ в качестве аббревиатуры выражения $\text{«}\underbrace{\text{add}(\text{add}(\ldots\text{add}}_{n-1}(< a_1 >, a_2)\ldots, a_{n-1}), a_n), \text{»}$, которое заключается в $(n-1)$-применении функции add для получения списка $< a_1, a_2, \ldots, a_n >$. Также определим одноместное отношение «быть значением типа $\text{datalist}\prod_{i=1}^{n}\text{T}_i$» $\in \text{datalist}\prod_{i=1}^{n}\text{T}_i$.

Родовой тип $\text{datalist}_\text{p} \subset \text{PO}$ определяется следующим образом:

$$\text{datalist}_\text{p} = \{< \text{datalist}_\text{c}, t >: \text{datalist}_\text{c} \in \text{type}, t \in \text{datalist}\prod_{i=1}^{n}\text{T}_i\}.$$

Семантика родового типа $\text{parlist}_\text{p} \subset \text{PO}$ аналогична семантике типа datalist_p, за исключением того, что первый элемент пары – константа $\text{datalist}_\text{c} \in \text{type}$ заменяется на

константу $\text{parlist}_c \in \text{type}$: $\text{parlist}_p = \{< \text{parlist}_c, t >: \text{parlist}_c \in \text{type}, t \in \text{datalist} \prod_{i=1}^{n} T_i\}$.

Пусть $\text{POlist} = \text{datalist}_p \cup \text{parlist}_p$, тогда PO=POsimple$\cup$POlist.

Рассмотрим операции для типов языка Пифагор PO. Описание семантики встроенных функций языка Пифагор на естественном языке приведено в [2]. Отметим некоторые особенности функций языка Пифагор. Все функции языка являются тотальными и в качестве аргументов могут принимать любой элемент из PO. При этом если аргумент «корректный», то функция возвращает результат вычислений, иначе – одну из констант ошибок $< \text{error}_c, e > \in \text{error}_p$.

Одна из возможных формализаций семантики языка приведена в [4]. По этим описаниям можно однозначно определить операции для типов языка в терминах элементов множества PO. В качестве примера рассмотрим семантику функции dup, которая создаёт список из одинаковых элементов. В качества входного значения функция принимает список из двух элементов, первый из которых – любой допустимый элемент a, а второй – целочисленная константа n. Результат – список из n копий элемента a. Для любого другого аргумента функция вернёт ошибку [2].

Формальное определение семантики функции dup:

$$\lambda\ p \in PO.(if((p \in \text{PO} \times \text{int}_p) \wedge (\text{select}(p,2) > 0))$$

$$\text{then}\ (\text{createList}(\underbrace{\text{select}(p,1),...,\text{select}(p,1)}_{\text{select}(p,2)})\ \text{else} < \text{error}_c, \text{BESEFUNCERROR} >) \cdot$$

Математическая семантика типов данных рядом достоинств [3]: преемственностью с неформальной семантикой, интерпретацией очень сложных типов, непротиворечивостью на уровне неформальной теории множеств, открытостью для дальнейших обобщений и расширений. Однако у неё есть и существенные недостатки, главный из которых – зависимость от человеческих представлений о понятии множества, которое «непереводимо» для компьютера.

2. Аксиоматическая семантика типов данных

Аксиоматический подход к семантике типов данных состоит в том, что для каждого из предопределённых (базисных) типов данных формулируется своя аксиоматическая теория первого порядка, для каждого из родовых типов формулируются свои (схемы) правил, позволяющих построить аксиоматическую теорию любого конкретного типа, построенного из базисных типов посредством родовых типов [3].

Сформулируем аксиоматические теории для каждого «простого (конкретного) типа». Для этого укажем сигнатуру, множество аксиом и правила вывода.

Все аксиоматические теории включают в себя общелогические аксиомы и правила вывода. Существуют различные эквивалентные аксиоматизации исчисления предикатов первого порядка [3; 5; 6], будем использовать аксиомы исчисления предикатов гильбертовского типа из [5].

1. Множество целых чисел int. Аксиоматическая система для целых чисел [3; 6; 7; 8], строится на основе аксиоматики натуральных чисел. Сигнатура Σ_Z включает константный символ (нульместная функция) c_n для каждого целого числа n. Функциональные и предикатные символы совпадают с операциями и отношениями множества int: -, +, –, ×, /, =, <, >, ≤, ≥, $\in$ int.

Аксиоматическая система для целых чисел включает в себя [3]:

1) общелогические аксиомы и правила вывода;

2) аксиомы теории с равенством [9]

$\forall x \in \text{int}.(x = x)$,

$\forall x \in \text{int}.\forall y \in \text{int}.((x = y) \to (y = x))$,

$\forall x \in \text{int}.\forall y \in \text{int}.\forall z \in \text{int}.(((x = y) \wedge (y = z)) \to (x = z))$;

3) аксиомы Дж. Пеано [3, 8]

$\forall x \in \text{int}.\neg(x + 1 = x)$,

$\forall x \in \text{int}.\forall y \in \text{int}.(((x + 1) = (y + 1) \to (x = y))$,

$\forall x \in \text{int}.(x + 0 = x)$,

$\forall x \in \text{int}.\forall y \in \text{int}.(x + (y + 1) = (x + y) + 1)$,

$\forall x \in \text{int}.(x \times 0 = 0)$,

$\forall x \in \text{int}.\forall y \in \text{int}.(x \times (y + 1) = (x \times y) + x)$,

правило вывода IND: $\dfrac{\varphi(0), \quad \varphi(x) \to \varphi(x+1)}{\forall x \in \text{int}.\varphi(x)}$;

4) аксиомы упорядочения [6]

$\forall x \in \text{int}.\neg(x < x)$,

$\forall x \in \text{int}.\forall y \in \text{int}.\forall z \in \text{int}.(((x < y) \wedge (y < z)) \to (x < z))$,

$\forall x \in \text{int}.\forall y \in \text{int}.((x = y) \vee (x < y) \vee (y < x))$,

$\forall x \in \text{int}.\forall y \in \text{int}.(x < y \leftrightarrow (y = (x+1)) \vee ((x+1) < y))$,

$\forall x \in \text{int}.\exists y \in \text{int}.(x = (y + 1))$;

5) аксиомы для отрицательных чисел [3]

$\forall x.((x \in \text{int}) \leftrightarrow (-x \in \text{int}))$,

$\forall x \in \text{int}.((x = 0) \vee (x < 0) \vee (-x < 0))$;

7) аксиомы, определяющие $-, /, >, \leq, \geq$ через $=, +, \times, <, -$.

2. Аксиоматизация типа действительных чисел float. Сигнатура Σ_R включает все символы $\Sigma_{\mathbf{Z}}$ (кроме символа $\in \text{int}$, который заменяется на $\in \text{float}$), а также константу c_r, для каждого рационального числа $r \in \mathbf{Q}$.

Аксиоматическая система состоит из [6]:

1) общелогических аксиом;

1) аксиом коммутативного поля;

2) аксиом упорядочения

$\forall x \in \text{float}.\forall y \in \text{float}.(((x > 0) \wedge (y > 0)) \to (x + y > 0))$,

$\forall x \in \text{float}.((x = 0) \vee (x > 0) \vee (-x > 0))$,

$\forall x \in \text{float}.(\neg((x > 0) \wedge (\neg x > 0)))$,

$\forall x \in \text{float}.\forall y \in \text{float}.(((x > 0) \wedge (y > 0)) \to (x \times y > 0))$;

3) $\forall x \in \text{float}.\exists y \in \text{float}.((x = y^2) \vee (-x = y^2))$,

$\forall x_0 \in \text{float}.\forall x_1 \in \text{float}\ldots\forall x_{2n} \in \text{float}.\exists x \in \text{float}.(x_0 + x_1 \times x + \ldots + x_{2n}x^{2n} + x^{2n+1} = 0)$, $\forall n \geq 1$;

4) аксиом, определяющих $-, /, <, >, \leq, \geq$ через $=, +, \times, <0, -$.

3. Аксиоматическая семантика булевского множества bool включает в себя общелогические аксиомы, аксиомы теории с равенством и аксиому $(\text{true} \in \text{bool}) \wedge (\text{false} \in \text{bool})$.

Аксиоматические системы для остальных «простых типов» t включают в себя общелогические аксиомы, аксиомы теории с равенством и символы одноместных отношений «быть значением типа t» $\in t$.

Рассмотрим аксиоматическую семантику простого родового типа datalist $\prod_{i=1}^{n} \mathrm{T}_i$. Если аксиоматическая семантика для всех типов T_i построена, одноместные отноше-

ния $\in \mathrm{T}_i$ аксиоматизированы, то можно задать аксиоматику типа $\mathrm{datalist}\prod_{i=1}^{n}\mathrm{T}_i$.

Сигнатура Σ_L содержит символ =, символ c для функции конструктора и n символов $s_1,\ldots,s_n$ для функций выбора, функциональный символ l для функции длины списка и символ одноместного отношения $\in \mathrm{datalist}\prod_{i=1}^{n}\mathrm{T}_i$. Аксиомы включают общелогические аксиомы, аксиомы теории равенства и аксиомы теории списков [7, 10]:

1) Аксиомы конструирования списка

$c(s_1(x),s_2(x),\ldots,s_n(x))=x$;

2) Аксиомы выбора элемента

$s_1(c(x_1,\ldots,x_n))=x_1$,

$s_2(c(x_1,\ldots,x_n))=x_2$,

…

$s_n(c(x_1,\ldots,x_n))=x_n$,

3) Аксиомы отсутствия зацикливания (символ $\neq$ является аббревиатурой отрицания формулы)

$s_1(x)\neq x$,

$s_2(x)\neq x$,

…

$s_n(x)\neq x$,

$s_1(s_1(x))\neq x$,

$s_2(s_2(x))\neq x$,

…

4) Аксиома определения длины списка

$l(x)=n$.

Рассмотрим аксиоматизация типов языка Пифагор PO. Для всех объектов типа PO, кроме общелогических аксиом и аксиом теории с равенством, введём аксиому, позволяющую связать объект PO со значением его «простого типа». Также введём одноместный функциональный символ extract и предикатный символ $\in \mathrm{PO}$. Аксиома будет иметь вид: $\forall <t,a>\in \mathrm{PO}.(extract(<t,a>)=a)$.

Функции языка Пифагор являются тотальными и могут работать с любыми аргументами из PO. Для описания аксиоматической семантики такой функции требуется задать несколько аксиом, которые будут отражать зависимость результата от свойств аргументов. На основе семантики из [4] можно однозначно определить все аксиомы, описывающие свойства встроенных функций.

Приведём пример аксиом для функции dup.

$$\forall <t,p>\in \mathrm{PO}.\left((p\in \mathrm{datalist}\prod_{i=1}^{n}\mathrm{T}_i)\wedge(l(p)=2)\wedge(\mathrm{s}_2(p)\in \mathrm{int})\wedge(\mathrm{s}_2(p)>0)\right)\rightarrow$$

$$\rightarrow(\mathrm{dup}(<t,p>)=r)\wedge(r\in \mathrm{datalist}\prod_{i=1}^{l(s_2(p))}\mathrm{T}_i)\wedge$$

$$\wedge\left(s_1(r)=s_1(p)\wedge s_2(r)=s_1(p)\wedge\ldots\wedge s_{l(s_2(p))}(r)=s_1(p)\right),$$

$$\forall <t,p>\in \mathrm{PO}.\neg\left((p\in \mathrm{datalist}\prod_{i=1}^{n}\mathrm{T}_i)\wedge(l(p)=2)\wedge(\mathrm{s}_2(p)\in \mathrm{int})\wedge(\mathrm{s}_2(p)>0)\right)\rightarrow$$

$$\rightarrow\left((\mathrm{dup}<t,p>=r)\wedge(r=\mathrm{BASEFUNCERROR})\right).$$

Аксиоматическая семантика даёт возможность «работать» со свойствами значений типов данных на чисто синтаксическом уровне.

Заключение

Автор считает, что полученная в данной работе математическая и аксиоматическая формальная семантика типов данных языка Пифагор может быть использована в качестве языка спецификаций к программе на языке Пифагор, при формальном доказательстве её корректности.

Литература

1. *Непомнящий В.А., Рякин О.М.* Прикладные методы верификации программ. – М.: Радио и связь, 1988. 255 с.

2. *Легалов А.И.* Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. № 1 (10). С. 71-89.

3. *Шилов Н.В.* Основы синтаксиса, семантики, трансляции и верификации программ. – Новосибирск: Издательство СО РАН, 2009. 340 с.

4. *Кропачева М.С.* Формализация семантики функционально-потокового языка параллельного программирования Пифагор // Проблемы информатизации региона. ПИР-2011: материалы XII Всероссийской научно-практической конференции. – Красноярск: Сибирский федеральный университет, 2011. С. 144-148.

5. *Ершов Ю.Л., Палютин Е.А.* Математическая логика: учебн. Пособие для вузов. – 2-е изд., испр. и доп. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. 336 с.

6. *Kreisel G., Krivine J.L.* Elements of Mathematical Logic. Studies in Logic and the Foundations of Mathematics. – North-Holland Publishing Company, 1967. – 221 p.

7. *Manna Z., Zarba C.* Combining decision procerures // LNCS. 2003. Vol. 2787. P. 453-468.

8. *Enderton H.B.* A Mathematical Introduction to Logic. 2nd ed., – Academic Press, 2001. 326 p.

9. *Скобелев В. В.* Автоматы на алгебраических структурах. Модели и методы их исследования. – Донецк: ИПММ НАН Украины, 2013. 307 с.

10. *Oppen D.C.* Reasoning about recursively defined data structures // Journal of the ACM. – 1980. N 3. P. 403-411.

Semantics of program data types for functional data-flow parallel programming language Pifagor

***Mariya Sergeevna Ushakova**, Postgraduate Student*

The article is devoted to formal semantics for data types of functional data-flow parallel programming language Pifagor. Mathematical semantics based on naive (informal) set theory and axiomatic semantics are concerned.

Keywords: axiomatic semantics of program data types, functional data-flow parallel programming, Pifagor programming language.

УДК 614.88

ОПРЕДЕЛЕНИЕ ПОТРЕБНОСТИ НАСЕЛЕНИЯ В АНЕСТЕЗИОЛОГО-РЕАНИМАЦИОННЫХ ВИДАХ СКОРОЙ МЕДИЦИНСКОЙ ПОМОЩИ

***Алексей Александрович Шумкин**, зав. подстанции Новоильинского района – врач скорой медицинской помощи*
Тел.: 8 904 572 7339, e-mail: a.a.shumkin@mail.ru
МБЛПУ «Станция скорой медицинской помощи» г. Новокузнецка