Серия «Математика»
2012. Т. 5, № 1. С. 13—25

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 510.5

Обобщенная операторная нотация и ее свойства *

А. А. Гаврюшкина, А. С. Москвина
Иркутский государственный университет

Аннотация. В работе предлагается обобщенная операторная нотация как средство синтаксической настройки языков программирования на предметные области, а также определяются и исследуются алгоритмы, реализующие понятие обобщенной операторной нотации.

Ключевые слова: обобщенная операторная нотация; предметно-ориентированный язык; язык программирования Libretto.

1. Введение

В современных информационных технологиях большую роль играют предметно-ориентированные языки (domain specific languages) [1, 2], которые ориентированы на работу в определенной предметной области. Примерами таких языков являются XPath, HTML, Logo, VHDL, YACC, SQL и мн. др. Ориентированность языка на предметную область облегчает написание и поддержку программных систем, делает код более компактным и читабельным и, в конечном счете, благотворно сказывается на стоимости разработки и ресурсных затратах.

В последнее время одной из актуальных тенденций стало добавление в универсальные языки программирования средств гибкой настройки синтаксиса языка на предметную область. Эти средства позволяют программисту превращать код, написанный на универсальном языке программирования, в предметно-ориентированный код. Такими средствами обладают многие современные языки, включая Scala, Ruby, Groovy и многие другие. Появилось даже понятие «языкового верста-

* Работа выполнена при финансовой поддержке Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг., государственный контракт № 16.740.11.0574 от 30 мая 2011 г.

ка» (language workbench, см., например, проект [3]) как универсального инструмента для построения предметно-ориентированных языков.

Универсальный язык программирования Libretto [4, 5], разрабатываемый в нашей исследовательской группе, также оснащен средствами для построения предметно-ориентированных языков. Некоторые из этих средств являются стандартными, другие же — оригинальные — требуют дополнительных исследований. Обобщенная операторная нотация, рассматриваемая в данной работе, является одним из таких многообещающих средств.

2. Обобщенная операторная нотация

Операторная нотация как инструмент синтаксической настройки часто включается в языки программирования. В частности, в языке Prolog [6], кроме предопределенных операций (таких, как `'+'`), можно задать собственные операции, например, логические связки:

```
:- op(100, yfx, ∨).
:- op(200, yfx, ∧).
:- op(50, fx, ¬).
```

Здесь конъюнкция и дизъюнкция задаются как левоассоциативные инфиксные операции, а отрицание — как префиксная. Числа задают приоритет операции, а выражение `yfx` — левую ассоциативность. Теперь

```
a ∨ ¬ b ∧ c
```

интерпретируется как операторная запись, эквивалентная терму

```
'∨'(a, '∧'('¬'(b), c))
```

В языке Flang [7, 8] было введено обобщение этой схемы, допускавшее произвольное количество аргументов слева и справа от операции, например,

```
:- op(100, 0, 2, fx, plus).
:- op(200, 0, 2, fx, times).

(plus 2 (times 3 4))
```

Здесь `plus` и `times` — префиксные двухаргументные операции (0 аргументов слева и 2 — справа). Видно, что их использование позволяет вводить во флэнге скобочную нотацию, применяемую в языке Lisp.

Данная идея может быть развита дальше. В частности, ассоциативность операции может задаваться с помощью ее приоритетов, если

разрешить различные приоритеты у операции слева и справа. Например, левую ассоциативность у операции + можно интерпретировать как то, что ее приоритет слева является более низким, чем справа, что влечет следующую расстановку скобок:

```
1 + 2 + 3 = (1 + 2) + 3
```

Поскольку справа приоритет у плюса выше, то 2 относится к первому вхождению +.

Таким образом, вместо задания приоритета и ассоциативности, можно задавать два приоритета — слева и справа. Например, плюс может быть определен так:

```
:- op(100, 99, 0, 2, '+').
```

Первые два аргумента — приоритет слева и справа, соответственно.

Возможность задавать разные приоритеты слева и справа является весьма выразительным инструментом. Например, задавая такие приоритеты, можно интерпретировать в операторной нотации операцию цикла:

```
for i := 1 to 5 do x := x + i
```

Определяем:

```
:- op(0, 1200, 0, 2, for).
:- op(0, 1200, 0, 1, do).
:- op(1, 1200, 1, 1, :=).
:- op(1199, 1199, 1, 1, to).
:- op(100, 99, 1, 1, +).
```

Данное определение сделает выражение цикла эквивалентным следующему терму

```
for( ':='(i, to(1, 5)), do(':='(x, '+'(x,i))))
```

Нулевой приоритет слева (например, у do) позволяет интерпретировать операцию с левой стороны как обычную константу, в то время, как справа ее влияние может распространяться на большие подтермы.

Назовем вариант нотации, базирующийся на двух приоритетах — слева и справа — обобщенной операторной нотацией. Алгоритмы для обычной операторной нотации хорошо известны (например, базирующиеся на обратной польской записи). Для обобщенной операторной нотации такие алгоритмы в существенной степени усложняются. Это обуславливает необходимость в исследовании их свойств. В частности,

реализованный в свое время подобный алгоритм в интерпретаторе языка Flang содержал ошибки. В данной работе нами дается формальное определение обобщенной операторной нотации, доказывается корректность соответствующих алгоритмов, и рассматриваются вопросы их сложности.

3. Термы

Рассмотрим конечное множество $F = \{f_1, \ldots, f_n\}$, элементы которого будем называть *операциями*, и конечное множество *констант* C (считаем, что $F \cap C = \emptyset$). Слова алфавита $F \cup C$ будем называть *выражениями*. Рассмотрим набор функции $\nu = (l, r, p, d)$, где функция $l : F \to \mathbb{N}$ (считаем $0 \in \mathbb{N}$) задает количество аргументов операции слева, $r : F \to \mathbb{N}$ задает количество аргументов операции справа, $p : F \to \mathbb{N}$ задает количественное значение приоритета операции над другими операциями и функция $d : F \to \{L, R\}$ определяет, является ли операция право-ассоциативной (случай, когда $d(f) = R$) или лево-ассоциативной ($d(f) = L$). Тройку (F, ν, C) назовем *порождающим набором*.

Пусть $\mathcal{P} = (F, \nu, C)$ порождающий набор. Рассмотрим минимальное множество слов X в алфавите $F \cup C \cup \{(,)\}$, удовлетворяющее следующим условиям:

1) $C \subseteq X$;

2) если $w_1, \ldots, w_n \in X$ и $f \in F$, то $(w_1 \ldots w_k f w_{k+1} \ldots w_n) \in X$.

Слова множества X будем называть *термами* порождающего набора $\mathcal{P}$. Подслово терма, которое само является термом, будем называть *подтермом*.

Из определения терма легко видно, что любой терм либо является константой, либо единственным образом представим в виде

$$(w_1 \ldots w_k f w_{k+1} \ldots w_n)$$

для некоторой $f \in F$ и некоторых термов $w_1, \ldots, w_n$, такую запись терма для краткости будем обозначать $(\ldots f \ldots)$.

Терм w будем называть *корректным термом* порождающего набора $\mathcal{P}$, если для любого его подтерма вида $(w_1 \ldots w_k f w_{k+1} \ldots w_{k+n})$ выполнено:

1) $k = l(f)$, $n = r(f)$;

2) для любого $i \in \{1, \ldots, k+n\}$, если $w_i = (\ldots g \ldots)$, то либо $g = f$, либо $p(g) < p(f)$;

3) если $d(f) = R$ ($d(f) = L$) и $w_i = (\dots g \dots)$ для $i \in \{1, \dots, k\}$ ($i \in \{k+1, \dots, k+n\}$), то $g \neq f$.

Очевидно, что любой подтерм корректного терма сам является корректным термом.

Таким образом, порождающий набор задает множество корректных термов, что придает смысл слову «порождающий» в термине «порождающий набор».

Будем говорить, что терм $\hat{w}$ *соответствует* выражению w, если выражение, полученное из $\hat{w}$ опусканием всех скобок, совпадает с w.

Далее будем считать, что любой порождающий набор $\mathcal{P} = (F, \nu, C)$ обладает свойством *однозначности*, то есть $p(f) \neq p(g)$ при $f \neq g$ для всех $f, g \in F$.

3.1. Случай с одним аргументом

Теорема 1. *Пусть $\mathcal{P} = (F, \nu, C)$ — порождающий набор. Тогда существует линейный алгоритм, который для выражения вида*

$$a_1 f_1 a_2 \dots a_k f_k a_{k+1},$$

где $a_i \in C$ и $f_i \in F$, находит соответствующий ему корректный терм.

Доказательство. На каждом шаге алгоритма будем строить *предтерм* — слово в алфавите $F \cup C \cup \{(,)\}$ следующего вида:

$$(w_1 g_1 \dots (w_{t-1} g_{t-1} (w_t g_t w_{t+1} g_{t+1} \dots g_n w_{n+1}$$

где $g_i \in F$ для $1 \leq i \leq n$, а w_i — термы для $1 \leq i \leq n+1$. Предтерм однозначно задается набором $(n, w_1, \dots, w_{n+1}, g_1, \dots, g_n, t)$, где $n > 0$ и $1 \leq t \leq n$.

Алгоритм

На шаге 0 строим предтерм $\mathbf{t_0}$ следующим образом. Положим $n_0 = k$; $w_i^0 = a_i$ для $1 \leq i \leq n_0 + 1$; $g_i^0 = f_i$ для $1 \leq i \leq n_0$ и $t_0 = 1$, то есть

$$\mathbf{t_0} = (a_1 f_1 a_2 f_2 \dots f_k a_{k+1}$$

Пусть на шаге s был построен предтерм:

$$\mathbf{t_s} = (w_1^s g_1^s \dots (w_{t^s-1}^s g_{t^s-1}^s (w_{t^s}^s g_{t^s}^s w_{t^s+1}^s g_{t^s+1}^s \dots g_{n^s}^s w_{n^s+1}^s$$

На шаге $s+1$, в зависиомости от случая, выполним следующие действия.

сл. 1 $t^s < n^s$, $p(g_{t^s}) < p(g_{t^s+1})$ или $g_{t^s} = g_{t^s+1}$ и $d(g_{t^s+1}) = L$. Этот случай разбивается на 2 подслучая.

сл. 1.1 $t^s = 1$. В предтерме $\mathbf{t_s}$ поставим две скобки следующим образом:

$$((w^s_{t^s} g^s_{t^s} w^s_{t^s+1}) g^s_{t^s+1} \dots g^s_{n^s} w^s_{n^s+1}$$

то есть положим: $n^{s+1} = n^s - 1$; $w^{s+1}_1 = (w^s_{t^s} g^s_{t^s} w^s_{t^s+1})$; $w^{s+1}_i = w^s_{i+1}$ для $2 \le i \le n^{s+1} + 1$ и $g^{s+1}_i = g^s_{i+1}$ для $1 \le i \le n^{s+1}$; $t^{s+1} = 1$. Шаг $s+1$ завершен, переходим к следующему шагу.

сл. 1.1 $t^s > 1$. Поставим одну закрывающую скобку:

$$(w^s_1 g^s_1 \dots (w^s_{t^s-1} g_{t^s-1} (w^s_{t^s} g^s_{t^s} w^s_{t^s+1}) g^s_{t^s+1} \dots g^s_{n^s} w^s_{n^s+1}$$

то есть положим: $n^{s+1} = n^s - 1$; $w^{s+1}_i = w^s_i$ и $g^{s+1}_i = g^s_i$ для $1 \le i \le t^s - 1$; $w^{s+1}_{t^s} = (w^s_{t^s} g^s_{t^s} w^s_{t^s+1})$, $w^{s+1}_i = w^s_{i+1}$ для $t^s + 1 \le i \le n^{s+1}$; $g^{s+1}_i = g^s_{i+1}$ для $t^s \le i \le n^{s+1}$; $t^{s+1} = t^s - 1$. Шаг $s+1$ завершен, переходим к следующему шагу.

сл. 2 $t^s < n^s$, $p(g_{t^s}) > p(g_{t^s+1})$ или $g_{t^s} = g_{t^s+1}$ и $d(g_{t^s+1}) = R$. В этом случае в предтерме $\mathbf{t_s}$ стваим одну открывающую скобку:

$$(w^s_1 g^s_1 \dots (w^s_{t^s-1} g^s_{t^s-1} (w^s_{t^s} g^s_{t^s} (w^s_{t^s+1} g^s_{t^s+1} \dots g^s_{n^s} w^s_{n^s+1}$$

то есть кладем: $n^{s+1} = n^s$; $w^{s+1}_i = w^s_i$ для $1 \le i \le n^s + 1$; $g^{s+1}_i = g^s_i$ для $1 \le i \le n^s$; $t^{s+1} = t^s + 1$. Шаг $s+1$ завершен, переходим к следующему шагу.

сл. 3 $t^s = n^s$. В предтерме $\mathbf{t_s}$ ставим одну закрывающую скобку:

$$(w^s_1 g^s_1 \dots (w^s_{t^s-1} g_{t^s-1} (w^s_{t^s} g^s_{t^s} w^s_{t^s+1})$$

то есть, если $t^s > 1$, то кладем: $n^{s+1} = n^s - 1$, $w^{s+1}_i = w^s_i$ и $g^{s+1}_i = g^s_i$ при $1 \le i \le t^s - 1$; $w^{s+1}_{t^s} = (w^s_{t^s} g^s_{t^s} w^s_{t^s+1})$; $t_{s+1} = t^s - 1$. Шаг $s+1$ завершен, переходим к следующему шагу. Если же $t^s = 1$, то ставим закрывающую скобку и получаем терм $(w^s_{t^s} g^s_{t^s} w^s_{t^s+1})$, который является искомым. Шаг $s+1$ завершен. Конец.

На каждом шаге ставится минимум одна скобка. Всего скобок в терме может быть не более чем удвоенное количество операций. Таким образом, для исходного выражения алгоритм завершиться за не более чем $2k$ шагов. На каждом шаге происходит проверка не более чем трех условий и переприсвоение не более чем $2k + 3$ переменных, то есть временная сложность алгоритма равна $2k(2k + 6)$.

□

Следствие 1. *Пусть $\mathcal{P} = (F, \nu, C)$ — порождающий набор такой, что $l(f) = r(f) = 1$ для любой $f \in F$. Тогда для любого выражения w вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$, где $a_i \in C$ и $f_i \in F$, существует корректный терм, соответствующий w.*

3.2. Случай с несколькими аргументами

Утверждение 1. *Пусть $\mathcal{P} = (F, \nu, C)$ — порождающий набор. Тогда если для выражения $w = u_1 f_1 u_2 \dots u_k f_k u_{k+1}$, где $u_i \in C^*$ для $1 \le i \le k+1$, существует корректный терм $\hat{w}$, то он единственный.*

Доказательство. Индукция по k. При $k = 1$ утверждение теоремы очевидно.

Пусть для всех $l < k$ утверждение теоремы верно, покажем что оно верно и для k. Будем писать $i <_w j$, если верно одно из следующих условий:

1) $p(f_i) < p(f_j)$;

2) $f_i = f_j$, $i < j$ и $d(f_i) = R$;

3) $f_i = f_j$, $i > j$ и $d(f_i) = L$.

Заметим, что всегда найдется такое i_0, для которого выполнено одно из следующих условий:

1) $i_0 <_w i_0 - 1$ и $i_0 <_w i_0 + 1$;

2) $i_0 = 1$ и $1 <_w 2$;

3) $i_0 = k$ и $k <_w k - 1$.

Заметим, что если $\hat{w}$ корректный терм соответствующий w, то он содержит корректный подтерм $(u f_{i_0} v)$, где u суффикс u_{i_0}, а v префикс u_{i_0+1}, такие, что $|u| = l(f_{i_0})$ и $|v| = r(f_{i_0})$. Заменим в выражении w подслово $u f_{i_0} v$ (точнее его вхождение, содержащее вхождение номер i_0 операции в выражение w) на некоторую константу $a \in C$. Обозначим полученное выражение w', оно содержит $k - 1$ операцию.

Теперь предположим, что для выражения w существуют два корректных терма $\hat{w}_1$ и $\hat{w}_2$. Заменим тогда в них соответсвующие вхождения терма $(u f_{i_0} v)$ на константу a. Очевидно, что после такой замены мы получим различные корректные термы $\hat{w}'_1$ и $\hat{w}'_2$. Несложно видеть, что они оба являются корректными термами для выражения w', что по предположению индукции невозможно. □

Теорема 2. *Пусть $\mathcal{P} = (F, \nu, C)$ — порождающий набор. Тогда существует полиномиальный алгоритм, который для выражения w находит соответствующий ему корректный терм, если он существует, и выдает отрицательный ответ в противном случае.*

Доказательство. Модифицируем алгоритм, описанный в доказательстве теоремы 1. Предтермами в данном случае будут являться слова следующего вида:

$$(u_1 g_1 \dots (u_{t-1} g_{t-1} (u_t g_t u_{t+1} g_{t+1} \dots g_n u_{n+1}$$

где $g_i \in F$ для $1 \leq i \leq n$, а u_i есть последовательность термов вида $w_{i1} \dots w_{im_i}$ для $1 \leq i \leq n+1$.

Тогда на шагах, на которых ставится открывающая скобка, будем ставить ее таким образом, чтобы количество термов между открывающей скобкой и операцией $g^s_{t^s}$ равнялось $l(g^s_{t^s})$. Если же термов недостаточно, то есть количество термов в $u^s_{t^s}$ меньше чем $l(g^s_{t^s})$, то алгоритм прекращает работу и выдает отрицательный ответ. Аналогично, на шагах, на которых ставится закрывающая скобка, ставим ее на расстоянии $r(g^s_{t^s})$ термов от операции $g^s_{t^s}$. □

4. Разделение приоритетов на левый и правый

В данном пункте вводится обобщенная операторная нотация, которая заключается в том, что левый и правый приоритеты в операциях задаются явно. Будем называть *обобщенным порождающим набором* набор (F, μ, C), где $\mu = (l, r, pl, pr)$, множества F, C и функции l, r определяются также как для порождающего набора, а функции $p_l : F \to \mathbb{N}$ и $p_r : F \to \mathbb{N}$ определяют количественные значения приоритета операции слева и справа, соответственно.

Определения выражения и терма остаются прежними. В определении корректного терма пункты 2 и 3 следует заменить на следующие:

2) для любого $i \in \{1, \dots, k\}$, если операция g содержится (как буква) в терме w_i, то $p_r(g) < p_l(f)$;

3) для любого $i \in \{k+1, \dots, k+n\}$, если операция g содержится (как буква) в терме w_i, то $p_r(f) > p_l(g)$.

Опять же, любой подтерм корректного терма сам является корректным термом.

Замечание 1. Для любого порождающего набора $\mathcal{P}$ существует обобщенный порождающий набор $\mathcal{P}'$, такой, что множества корректных термов, порождаемых наборами $\mathcal{P}$ и $\mathcal{P}'$, совпадают.

Доказательство. Положим:

$$p_l(f) = 2(p(f)+1) \quad \text{и} \quad p_r(f) = \begin{cases} p_l(f) - 1, & \text{если } d(f) = R; \\ p_l(f) + 1, & \text{если } d(f) = L. \end{cases}$$

□

Далее будем считать, что любой обобщенный порождающий набор $\mathcal{P} = (F, \mu, C)$ обладает свойством *однозначности*, то есть $p_r(f) \neq p_l(g)$ для всех $f, g \in F$.

4.1. Случай с одним аргументом

Утверждение 2. *Существует обобщенный порождающий набор $\mathcal{P} = (F, \mu, C)$, такой, что $l(f) = r(f) = 1$ для любой $f \in F$, и выражение вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$, которому не соответствует ни один корректный терм.*

Доказательство. Рассмотрим порождающий набор:

$$F = \{f, g, h\}, \quad C = \{a\}, \quad p_l(g) < p_r(f) < p_l(h) < p_r(g)$$

Тогда для $a\,f\,a\,g\,a\,h\,a$ нет соответствующего корректного терма. □

Пусть $\mathcal{P} = (F, \mu, C)$ — обобщенный порождающий набор, такой, что $l(f) = r(f) = 1$ для любой $f \in F$. Определим множество $L(w)$ *вложенных левых аргументов* и множество $R(w)$ *вложенных правых аргументов* терма w набора $\mathcal{P}$:

1) если $w \in C$, тогда $L(w) = R(w) = \emptyset$;

2) если $w = (w_1 f w_2)$, то $L(w) = L(w_1) \cup \{f\}$ и $R(w) = \{f\} \cup R(w_2)$.

Введем определение *почти-корректного терма* набора $\mathcal{P}$, заменив условия 2 и 3 определения корректного терма на следующие. Для любого подтерма $(w_1 f w_2)$ выполнено:

2) $p_r(g) < p_l(f)$ для любой $g \in R(w_1)$,

3) $p_r(f) > p_l(f)$ для любой $g \in L(w_2)$.

Теорема 3. *Пусть (F, μ, C) — обобщенный порождающий набор, такой, что $l(f) = r(f) = 1$ для любой $f \in F$. Тогда для любого выражения w вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$ существует и притом единственный соответствующий ему почти-корректный терм $\hat{w}$. Кроме того, существует полиномиальный алгоритм, строящий этот почти-корректный терм.*

Доказательство. Если в алгоритме, описанном в теореме 1, заменить условие «$p(g_{t^s}) < p(g_{t^s+1})$ или $g_{t^s} = g_{t^s+1}$ и $d(g_{t^s+1}) = L$» на условие «$p_r(g_{t^s}) < p_l(g_{t^s+1})$», а условие «$p(g_{t^s}) > p(g_{t^s+1})$ или $g_{t^s} = g_{t^s+1}$ и $d(g_{t^s+1}) = R$» — на условие «$p_r(g_{t^s}) > p_l(g_{t^s+1})$», то полученный алгоритм будет строить необходимый почти-корректный терм.

Доказательство единственности почти-корректного терма аналогично доказательству предложения 1. □

Рассмотрим выражение вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$, определим для каждого вхождения операции i левую $B_l(i)$ и правую $B_r(i)$ границы преобладания этого вхождения операции следующим образом:

$B_l(i) = 1$, если $i = 1$;

$B_l(i) = j$, где $j \le i$, $p_r(f_{j-1}) > p_l(f_i)$, а $max\{p_r(f_j), \dots, p_r(f_{i-1})\} < p_l(f_i)$, если $i > 1$.

Аналогично:

$B_r(i) = j + 1$, где $i \le j$, $p_r(f_i) < p_l(f_{j+1})$,
а $p_r(f_i) > max\{p_l(f_{i+1}), \dots, p_l(f_j)\}$, если $i < k$;

$B_r(k) = k + 1$, если $i = k$.

Заметим, что $B_l(i) \le i < B_r(i)$ для любого $i \in \{1, \dots, k\}$.

Теорема 4. *Пусть (F, μ, C) — обобщенный порождающий набор, такой, что $l(f) = r(f) = 1$ для любой $f \in F$. Тогда для того, чтобы у выражения w вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$ существовал соответствующий ему корректный терм $\hat{w}$ необходимо и достаточно, чтобы для любых $1 \le i < j \le k$ либо $B_l(i) < B_l(j) < B_r(j) \le B_r(i)$, либо $B_l(j) \le B_l(i) < B_r(i) < B_r(j)$;*

Доказательство. Необходимость легко следует из определений. Чтобы доказать достаточность условия, необходимо предъявить корректной терм $\hat{w}$ для выражения w. Для этого в выражении w для каждого $i \in \{1, \dots, k\}$ поставим открывающую скобку перед $a_{B_l(i)}$ и закрывающую скобку после $a_{B_r(i)}$. Индукцией по k несложно показать, что полученное слово $\hat{w}$ является корректным термом. □

Теорема 5. *Пусть (F, μ, C) — обобщенный порождающий набор, такой, что $l(f) = r(f) = 1$ для любой $f \in F$. Тогда если для выражения вида $a_1 f_1 a_2 \dots a_k f_k a_{k+1}$ существует соответствующий ему корректный терм, то он единственный. Кроме того, существует полиномиальный алгоритм, который находит этот корректный терм, если он существует, и выдает отрицательный ответ в противном случае.*

Доказательство. Так как корректный терм является почти-корректным, то единственность следует из единственности почти-корректного терма.

Используя алгоритм построения почти-корректного терма, строится почти корректный терм. Затем поочередно, начиная с самой внешней операции, происходит проверка условий 2 и 3 корректности терма.

Сложность проверки терма на корректность квадратичная, так как для проверки каждой операции на удовлетворение условиям 2 и 3 корректности терма требуется не более чем $O(k)$ действий. □

4.2. Случай с несколькими аргументами

Обобщим определение почти-корректного терма для произвольного обобщенного порождающего набора $\mathcal{P} = (F, \mu, C)$. Множества $L(w)$ и $R(w)$ определяются как:

1) если $w \in C$, тогда $L(w) = R(w) = \emptyset$;

2) если $w = (w_1 \dots w_k f w_{k+1} \dots w_{k+n})$, то $L(w) = \bigcup\limits_{i=1}^{k} L(w_i) \cup \{f\}$ и $R(w) = \{f\} \cup \bigcup\limits_{i=k+1}^{k+n} R(w_i)$.

Терм является почти-корректным, если для него выполнено условие 1 корректности терма и для любого его подтерма вида

$$(w_1 \dots w_k f w_{k+1} \dots w_{k+n})$$

выполнено:

2) $p_r(g) < p_l(f)$ для любой $g \in \bigcup\limits_{i=k+1}^{k+n} R(w_i)$,

3) $p_r(f) > p_l(f)$ для любой $g \in \bigcup\limits_{i=1}^{k} L(w_i)$.

Тогда верны следующие обобщения теоремы 3 и теоремы 5.

Теорема 6. *Пусть $\mathcal{G} = (F, \mu, C)$ — обобщенный порождающий набор. Тогда если для выражения w вида $u_1 f_1 u_2 \dots u_k f_k u_{k+1}$, где $u_i \in C^*$ для $1 \le i \le k+1$, существует соответствующий ему почти-корректный терм $\hat{w}$, то он единственный. Кроме того, существует алгоритм, который находит этот почти-корректный терм, если он существует, и выдает отрицательный ответ в противном случае.*

Теорема 7. *Пусть $\mathcal{G} = (F, \mu, C)$ — обобщенный порождающий набор. Тогда если для выражения $u_1 f_1 u_2 \dots u_k f_k u_{k+1}$, где $u_i \in C^*$ для $1 \le i \le k+1$, существует корректный терм, то он единственный. Существует полиномиальный алгоритм, который находит этот корректный терм, если он существует, и выдает отрицательный ответ в противном случае.*

5. Заключение

В данной работе рассматривается обобщенная операторная нотация как инструмент для настройки синтаксиса универсального языка программирования и построения предметно-ориентированных языков. В работе сформулировано определение обобщенной операторной нотации и исследованы алгоритмы, проверяющие выражения на корректность и расставляющие в них скобки. По результатам данной работы будет принято решение о включении конструкций, поддерживающих обобщенную операторную нотацию в универсальный динамический язык программирования Libretto. Для практического тестирования обобщенной операторной нотации и соответствующих алгоритмов, нами также была реализована ее экспериментальная версия в виде библиотеки языка Libretto 0.9.

Список литературы

1. Mernik M. When and how to develop domain-specific languages / Marjan Mernik, Jan Heering and Anthony M. Sloane // ACM Computing Surveys. – 2005. – Vol. 37, N 4. – P. 316–344.
2. Spinellis D. Notable design patterns for domain specific languages / Diomidis Spinellis // Journal of Systems and Software. – 2001. – Vol. 56, N 1. – P.91–99.
3. The Language Workbench Challenge. URL: http://www.languageworkbenches.net.
4. Malykh A. Query Language for Logic Architectures / A. Malykh, A. Mantsivoda // Proceedings of 7th International Conference «Perspectives of System Informatics». – Springer-Verlag Berlin Heidelberg, Lecture Notes in Computer Science 5947, 2010. – P. 294–305.
5. Малых А. А. Объектно-ориентированная дескриптивная логика / А. А. Малых, А. В. Манцивода // Изв. Иркут. гос. ун-та. Сер.: Математика. – 2011. – Т. 4, № 1. – С. 57–72.
6. Клоксин У. Программирование на языке Пролог / У. Клоксин, К. Меллиш. – М. : Мир, 1987. – 336 с.
7. Mantsivoda A. Flang: A Functional-Logic Language / A. Mantsivoda // Lecture Notes in Computer Science. – Berlin : Springer, 1992. – Vol. 567. – P.257-270.
8. Манцивода А. В. Флэнг – язык искусственного интеллекта / А. В. Манцивода // Кибернетика. – 1993. – № 5. – С. 350–367.
9. Ershov Yu. L. Semantic Programming / Yu. L. Ershov, S. S. Goncharov, D. I. Sviridenko // Information processing, Proc. IFIP 10th World Comput. Congress. – Dublin, 1986. – Vol. 10. – P. 1093-1100,

---

A. A. Gavryushkina, A. S. Moskvina

A Generalized Operator Notation and its Properties

Abstract. A generalized operator notation is considered in this paper as a tool for programming language syntactic tuning to domains of discourse. Also some algorithms dealing with the generalized operator notation are introduced and investigated.

Keywords: generalized operator notation, domain specific language, programming language Libretto

Гаврюшкина Александра Анатольевна, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1. тел.: (3952)242210 (gavryushkina@gmail.com)

Москвина Анастасия Сергеевна, Иркутский государственный университет, 664003, Иркутск, ул. К. Маркса, 1. тел.: (3952)242210

Alexandra A. Gavryushkina, Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210 (gavryushkina@gmail.com)

Anastasia S. Moskvina, Irkutsk State University, 1, K. Marks St., Irkutsk, 664003 Phone: (3952)242210