УДК 519.6
DOI: 10.18101/2304-5728-2018-2-95-109

ОБ ОДНОМ ПОДХОДЕ К МОДЕЛИРОВАНИЮ ДИНАМИЧЕСКИХ СИСТЕМ

© Николаева Дарима Доржиевна
аспирант,
Бурятский государственный университет
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: darisha89@yandex.ru

© Ширапов Дашадондок Шагдарович
доктор физико-математических наук, профессор,
Бурятский государственный университет
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: shir48@mail.ru

© Антонов Вячеслав Иосифович
кандидат физико-математических наук, доцент,
Бурятский государственный университет
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: antonov.imi@mail.ru

При разработке комплексов компьютерных программ, моделирующих различные динамические системы, часто требуется построить математические модели той или иной предметной области. В статье для заданного языка программирования построены функции, такие, что для любой заданной программы можно построить суперпозицию функций (терм). Вычисление упомянутого терма порождает вычислительный процесс, который возникает при исполнении программы. Если программа предназначена для моделирования динамической системы, то вычисление алгебраического терма является адекватным моделированием функционирования динамической системы. Таким образом, разработана алгебраическая модель языка программирования, предназначенная для моделирования динамических систем, где вычисление алгебраического терма порождает моделирующий процесс динамических систем. Для построения указанных функций необходимо точно описать области определения и области значений этих функций. Для построения областей определения и областей значений этих функций используются контекстно-свободные грамматики, операция отождествления. Кроме указанных средств применяются понятие многоуровневости модели, концепция косвенного именования (косвенная адресация), рекурсия, а также некоторые простые средства из теории алгоритмов и теории программирования. Таким образом, найден достаточно «широкий» по практическому охвату способ компьютерного моделирования различных динамических систем, где произвольная программа может быть представлена в виде алгебраического терма универсальной алгебры с сигнатурой из указанных функций.

Ключевые слова: динамические системы; моделирование; математическая модель языка; универсальная алгебра; контекстно-свободная грамматика; рекурсия; интерпретатор; семантика.

Введение

Для того чтобы детально исследовать динамические процессы вычислений на компьютере, необходимо удобное представление алгоритмов, областей их определения и значений с точки зрения построения новых программ или функций на основе заданных базисных функций по заданным правилам. Одной из основных задач является построение базисных функций. Тогда любая программа будет представлена в виде суперпозиции базисных функций. Для того чтобы решить указанную проблему, мы используем следующие формально точные понятия (математические примитивы): конечное множество структурно согласованных контекстно-свободных грамматик; операция отождествления; универсальная функция eval (аналог интерпретатора ЛИСП) в терминах КС-грамматик; рекурсия; многоуровневость понятий, которая является естественной для формальных грамматик; формальные модели различных типов памяти: память с варьируемой структурой, с постоянной структурой и управляющая память. Данный указанный перечень математически точных понятий (примитивов) позволяет построить математическую модель языка программирования (ММЯП), которая, в свою очередь, позволяет моделировать на компьютере различные динамические системы.

1. Постановка задачи

В данной статье требуется построить модель, которая для любой заданной синтаксически правильной программы x в языке программирования общего назначения (Алгол 68, С++, Паскаль и т. д.) указывает способ представления отображения из области определения (входные данные) в область значений (выходные данные). Данное отображение индуцируется программой x в виде алгебраического терма $F \in T(\Omega)$, где $T(\Omega)$ — множество всех термов в языке L, Ω — сигнатура базисных функций языка L. Для этого необходимо представить семантику языка программирования L в виде алгебраической системы (M, Ω). При запуске программы x порождаются вычислительные процессы. Относительно этих вычислительных процессов в выбранных точках программы (алгебраического терма $T(\Omega)$) можно сформулировать интересуемые нас утверждения, представленные в различных видах: в виде позитивно образованных формул [1], в виде формул языка логики первой и второй ступеней алгебраической системы и т. д. Чтобы представить произвольную программу x языка L в виде алгебраического терма $F \in T(\Omega)$, необходимо построить систему базисных функций

$$\Omega = \{\Phi 0^{(1)}, \ldots, \Phi N^{(1)}, \Phi 0^{(2)}, \ldots, \Phi N^{(m)}\}.$$

Синтаксический анализ заданной программы x дает в качестве результата суперпозицию базисных функций $F \in T(\Omega)$. В статье показано, как можно построить алгебраическую систему для заданного языка L. При конкретной реализации языка программирования L универсальность рассматриваемой модели $T(\Omega)$ ограничивается множеством допустимых состояний памяти. При первом приближении это может быть обусловлено количеством выделяемых разрядов для представления чисел, количеством букв в строке (строковые данные) и т. д. Если более детально разобрать вопрос о множестве допустимых состояний памяти (ОДСП), то необходимо рассматривать представление структуры различных видов значений в памяти компьютера. Под различными видами данных подразумеваются числа (целые, вещественные, комплексные); логические значения; представления процедур и т. д. Для того чтобы формально описать структуру различных достаточно нетривиальных видов данных в указанном подходе используются многоуровневые грамматики видов и значений. Формально можно описать ОДСП пятеркой (V, B, G_1, G_2, G_{-1}), где V — множество переменных программы, $B = \{b : V \to M\}$ — множество всех частичных отображений множества переменных V в M, где M — множество значений, которые индуцируются грамматиками G_1, G_2, G_{-1}. Здесь G_1 — грамматика видов данных, G_2 — грамматика значений, G_{-1} — грамматика структуры представления данных во внутренней памяти, которая формально описывает структуру представления всех видов значений. Для представления сложных межвидовых логических отношений, помимо вышеуказанной пятерки, можно ввести функции приведения одних видов в другие. Ниже в приложении приведены некоторые примеры таких функций: $g_1^{(1)}, g_1^{(2)}, g_2^{(1)}, g_2^{(2)}$. Эти функции вводят частичный порядок на множествах M_{-1}, M_2, которые порождаются грамматиками G_1, G_2 соответственно.

2. Способы построения функций в терминах формальных грамматик

Для решения поставленной задачи использование КС-грамматик тесно связано с внедрением именных множеств: нетерминальные и терминальные символы образуют имена множеств, элементы которых являются словами в некоторых алфавитах. В связи с этим рассмотрим понятие имя более подробно. Понятие имя в программировании и в компьютерном моделировании играет фундаментальную роль. Если смотреть на задачу с точки зрения прагматики, то постановщик задачи имеет дело с предметно-ориентированной логикой: имена понятий, имена объектов, имена отношений между ними и т. д. С другой стороны, компьютер имеет абстрактные пронумерованные ячейки памяти, у которых номера ячеек явля-

ются их адресами, а также абстрактные машинные операции либо абстрактные операции языка программирования общего назначения, оторванные от предметной логики. Для построения модели предметной области, прежде всего, необходимо найти удачное представление элементов именного множества для максимального упрощения предметно-ориентированных операций, функций, предикатов и формул предметной области в памяти разрабатываемого приложения вместе со всеми их свойствами и отношениями (единицы измерений значений переменных). При разработке приложения разработчик всегда вынужден выполнить переход-операцию от более общих абстрактных понятий-имен к более конкретным предметно-ориентированным понятиям-именам. В связи с этим становится актуальной работа с ***именами***, обозначающими понятия и их интерпретации: 1) операция разыменования; 2) операция именования. Для построения адекватной цифровой модели предметной области с каждым именем (объектом, понятием) взаимно-однозначно связывается неизменный адрес памяти. Таким образом, в данной работе понятие имя и его адрес отождествляются. На основании вышесказанного в работе выбран язык программирования с расширенной концепцией косвенного именования, для которого строится универсальная алгебра (M,Ω), являющаяся моделью семантики заданного языка.

Рассмотрим конкретно операции разыменования и именования на примере реализации семантики этих операций средствами функциональных грамматик. Для этого рассмотрим семантику языка программирования ASPLE с многократной адресацией, где наиболее наглядно демонстрируется семантика *именованных* множеств. В [4] проводится сравнение четырех подходов к формальному описанию языков программирования на примере простого подмножества языка Алгол-68. В данной работе выбрана двухуровневая универсальная схема вида:

$$Y(M_{-1}, L(G_0), M_1, M_2, \{\Phi i^{(1)}\}, \{\Phi i^{(2)}\}, \{g_j^{(1)}\}, \{g_j^{(2)}\}),$$

где M_{-1} — множество всех переменных типа конт, G_0 — КС-грамматика, задающая синтаксис языка ASPLE, $L(G_0)$ — множество текстов программ, M_1 — множество видов, M_2 — множество значений, $\{\Phi i^{(1)}\}$ — конечное множество функций (алгоритмов) на уровне видов M_1, $\{\Phi i^{(2)}\}$ — конечное множество функций (алгоритмов) на уровне значений M_2, $\{g_j^{(1)}\}$ — конечная система преобразований видов, порождающая частичный порядок на множестве M_1, $\{g_j^{(2)}\}$ — конечная система преобразований значений, порождающая частичный порядок на множестве M_2. Множества M_{-1}, M_1, M_2 порождаются КС-грамматиками G_{-1}, G_1, G_2 соот-

ветственно. Построение алгоритмов $\{\Phi i^{(1)}\},\{\Phi i^{(2)}\},\{g_j^{(1)}\},\{g_j^{(2)}\}$ в терминах этих грамматик является основной задачей конструктивного определения математической модели языка программирования. Внешняя грамматика G_0 представляет собой четверку:

$$G_0 = < V_{T0}, V_{N0}, P_0, A_0 >,$$

где V_{T0} — множество терминальных символов, V_{N0} — множество нетерминальных символов, P_0 — множество правил вида

$$A_i \to x_1 A_{i1} x_2 A_{i2} \ldots x_k A_{ik} x_{k+1} (\Phi i^{(1)}), \qquad (1)$$

A_0 — начальный символ грамматики.

Обозначим множество терминальных цепочек через V_{T0}^*, а множество сентенциональных форм через $(V_{T0} \cup V_{N0})^*$. Алгоритм $\Phi i^{(1)}$ для соответствующего нетерминала A_i будет иметь следующий вид:

$$\begin{gathered}\Phi i^{(1)} = (< mun >_{ik}\ y_{i1}, \ldots, < mun >_{ik}\ y_{ik}) < mun >: \\ \ldots, Q_{ik}) z_1 S_{t1}^{(\beta 1)} z_2 \ldots z_m S_{tm}^{(\beta m)} z_{m+1}\end{gathered} \qquad (2)$$

где z_j — терминальная цепочка одной из грамматик. Основную роль в указанных алгоритмах $\Phi i^{(1)}$ играет операция отождествления, которой можно дать следующее определение. Пусть задана КС-грамматика

$$G_M = (V_N, V_T, P, S),$$

которая порождает множество M, где

$$M = \{x \mid x - \text{ слово, порожденное грамматикой } G_M \text{ из некоторой сентенциальной формы } G_M\ \}.$$

Каждая функция Φi определена на некотором подмножестве декартовой степени множества M со значениями во множестве M. Пусть φ, ψ — слова в алфавите $V_N \cup V_T$ (сентенциальные формы грамматики G), где нетерминальные символы помечены индексами, которые являются формальными параметрами операции отождествления. Тогда операция отождествления синтаксически представляется в виде $(\varphi)\psi$ и содержательно определяется как отображение совокупности входных формальных параметров, принадлежащих к φ, на совокупность выходных (т. е. результирующих) формальных параметров, входящих в ψ. Семантически операция отождествления $(\varphi)\psi$ трактуется так: если x — фактический аргумент этой операции, то строится вывод $\varphi \Rightarrow^* x$ этого слова в грамматике G_M. Тогда нетерминальные символы слова φ получают в качест-

ве значений некоторые подслова слова x (одинаковые нетерминалы, помеченные одинаковыми символами, должны принимать одни и те же значения), слово ψ определяет результат этой операции.

В общем случае операция отождествления неоднозначна. С целью достижения однозначности введем операцию левого отождествления: самый левый нетерминал в слове φ будет получать в качестве значения минимальное по длине подслово слова x [2; 3]. Сказанное продемонстрируем на примере.

Пример 1. Пусть задана простейшая грамматика G_M:

$$M ::= AM \mid BM \mid \varepsilon;\ \ A ::= aA \mid a\ ;\ \ B ::= bB \mid b\ .$$

Рассмотрим функцию $f_i = (ABA)BAB;\ \ \varphi = ABA,\ \psi = BAB$. Пусть $x = a^n b^m a^n$. Тогда вывод $\varphi \Rightarrow x$ определит значения для A и B: $A = a^n$, $B = b^m$. Результат равен $\psi = BAB = b^m a^n b^m$.

Необходимо подчеркнуть важность операции отождествления $(\varphi)\psi$ в процессах обобщения и конкретизации функций (функциональных понятий) совместно со следующими нижеперечисленными операциями и концептуальными элементами: КС-грамматика для описания синтаксиса и структуры данных, описание суперпозиций функций и операции *eval* и *quote* (аналоги операций из Лиспа), описание множеств M_i для описания управления данными и памятью, память для описания структуры управления последовательностью действий, рекурсия, альтернативная операция $A \mid B$, управляющая память, функции приведения, многоуровневость языка.

Пример 2. Пусть задан фрагмент программы:

начало цел v, l ; имя имя цел a ; …

Рассмотрим на данном фрагменте, как работает система базисных функций

$$\Omega = \{\Phi 0^{(1)}, \ldots, \Phi N^{(1)}, \Phi 0^{(2)}, \ldots, \Phi N^{(m)}\}.$$

Результатом будет следующая лемма.

Лемма 1. В памяти формируется в соответствии с ОДСП следующая структура данных:

$$v * 1 \# v1 * \text{неопр} \# l * 1 \# l1 * \text{неопр} \# a * 1 \# a1 * 2 \# a2 * \text{неопр} \# \omega F\ .$$

Доказательство. Предположим, что начальное состояние памяти до объявления переменных равно $\varepsilon\,\omega F$, где F — файл ввода. При входных данных цел v для указанной базисной функции $\Phi 6 =$ (текст x, y) знач: (ВИД, X) r (имя ВИД, $X * 1 \# X1$, ВИД) имеем сопоставление: (цел, v) r (имя цел, $v * 1 \# v1$, цел). Для $\Phi 5 =$ (знач x, текст y) знач: (ВИД$\&M$, X) r (ВИД, $MX * 1 \# X1 *$, ВИД) имеем сопоставление: (имя цел$\&M, l$)

r (имя цел, $Ml*1\#l1*$, имя цел). Таким образом, сопоставляя каждому нетерминальному символу (формальному параметру) его фактические параметры (значения), можно вычислить остальные функции, фигурирующие в узлах синтаксического дерева разбора: $\Phi 2, \Phi 6$ и $\Phi 3$. Результат последовательного вычисления указанных функций даст следующее состояние памяти:

$$v*1\#v1*\text{неопр}\#l*1\#l1*\text{неопр}\#a*1\#a1*2\#a2*\text{неопр}\#\omega F,$$

что и требовалось доказать.

Пусть грамматики G_{-1}, G_1, G_2, порождающие множества M_{-1}, M_i, M_2 соответственно, имеют вид $G_{-1} = (V_{T-1}, V_{N-1}, P_{-1}, A_{-1})$, $G_1 = (V_{T1}, V_{N1}, P_1, A_1)$, $G_2 = (V_{T2}, V_{N2}, P_2, A_2)$. Общим и наиболее естественным требованием к построению грамматик G_{-1}, G_1, G_2 является требование об их структурной согласованности. Грамматика G_i $(L(G_i) = M_i)$ называется структурно согласованной по терминальным символам $\{a_{i1}, \ldots, a_{ik}\} = V_{Ti}^{(i,j)} \subset V_{Ti}$ с грамматикой G_j $(L(G_i) = M_i)$ тогда и только тогда, когда каждому $a_{it} \in V_{Ti}^{(i,j)}$ сопоставлено однозначно некоторое подмножество $M_j^{(it)}$ подслов слов множества M_j так, что каждому слову $x = a_{t1}a_{t2}\ldots a_{tm} \in (V_{Ti}^{(i,j)})^*$ будет соответствовать подмножество $M_j^{(i1)}, \ldots, M_j^{(im)}$ множества M_j.

Структурная согласованность грамматик G_{-1}, G_1, G_2 требуется для выразительного построения логики базисных функций языка. Под структурной согласованностью будем понимать логическую и смысловую связь именованных нетерминалов и терминальных строк, которые диктуют необходимость построения семантических отображений $\{\Phi i^{(1)}\}, \{\Phi i^{(2)}\}, \{g_j^{(1)}\}, \{g_j^{(2)}\}$. Другими словами, $V_{N\text{-}1} \cap V_{N1} \cap V_{N2} \neq \varnothing$, $V_{T\text{-}1} \cap V_{T1} \cap V_{T2} \neq \varnothing$. Сказанное продемонстрируем на следующих грамматиках G_1 и G_2, где G_1 — грамматика видов переменных, а G_2 — грамматика значений переменных языка ASPLE. Содержательно нетерминал-понятие АБСТР из грамматики G_1 обозначает уровень абстракции или двойственное ему понятие — уровень конкретизации.

Грамматика $G_1 = (V_{T1}, V_{N1}, P_1,$ ПРАВИД) — структура видов.

ПРАВИД : : = ВИД | ничто; ВИД : := АВИД | АБСТР ВИД;

АВИД : := лог | цел ; АБСТР : := ε | АБСТР имя ;
Упорядоченность на M_1 : $g_1^{(1)}$ = (имя ВИД) ВИД.

Пара структурно согласованных нетерминалов АБСТР из грамматики G_1 и ЦЕПЬ из грамматики $G_2^{(1)}$ моделирует концепцию косвенного именования. ЦЕПЬ является конкретизацией АБСТР. $G_2^{(1)}$:

ПРАЗНАЧ : := ЗНАЧ | неопр ; ЗНАЧ : := АЗНАЧ | ЦЕПЬ АЗНАЧ ;
АЗНАЧ : := ЛОГ | ЦЕЛ ; ЦЕПЬ : : = ε | ЦЕПЬ $XH\#XH*$;
ЦЕЛ : := ЦЦЦ | неопр ; ЛОГ : := истина | ложь | неопр ;
H : := ε | Ц | ЦЦ | ЦЕЛ ; Ц : := 0 | 1 | 2 . . . | 9 .
Упорядоченность на M_{-1} :

$g^{(2)}$ = (конт t , знач x) знач : $(M\#XH*PM^{(1)}\omega F,\ XH)\ P$.

Из представленных грамматик видно, что существует квазиизоморфизм, т. е. некоторое подобие изоморфизма, а именно:

ПРАВИД $\to$ ПРАЗНАЧ ; ВИД $\to$ ЗНАЧ; АВИД $\to$ АЗНАЧ;
АБСТР $\to$ ЦЕПЬ; имя $\to H\#XH*$;
лог $\to$ ЛОГ : : = истина | ложь | неопр ; цел $\to$ ЦЕЛ : : = ЦЦЦ .

Из вышесказанного видно, что грамматика $G_2^{(1)}$ является продолжением грамматики G_1 : терминальные символы грамматики G_1 находят дальнейшую конкретизацию. Грамматику $G_2^{(1)}$ назовем продолжением грамматики G_1 по терминалам имя, лог, цел.

Для улучшения читабельности семантической функции присваивания $\Phi 7^{(1)}$ и $\Phi 7^{(2)}$ (верхние индексы (1) и (2) указывают, что данные функции рассматриваются на уровне видов и значений соответственно, однако, возможно, индексы будут опускаться, если это не будет приводить к путанице) необходимо преобразовать в сторону упрощения грамматику $G_2^{(1)}$, используя идею косвенного именования: громоздкий нетерминал ЦЕПЬ заменим на XH для упрощения алгоритмической обработки и понимания сложноструктурированной информации, которая находится в сжатом виде в списке. Для этого достаточно иметь имя XH (head-list) вышеупомянутого списка. Семантические смыслы нетерминалов ЦЕПЬ и H неразрывно связаны: ЦЕПЬ является списком; нетерминалы X и H вместе являются начальным адресом отрезка списка ЦЕПЬ в момент исполнения функций $\Phi 5, \Phi 6, \Phi 7$; XH есть ячейка, содержащая ссылочный адрес на следующий элемент списка. Конкретное значение есть список, однако для однозначного задания значения достаточно указать на-

чальный адрес списка XH. В связи с этим выполним преобразование, которое упрощает грамматики $G_2^{(1)}$ в грамматику G_2:

ПРАЗНАЧ : := ЗНАЧ ; ЗНАЧ : := ЦЕЛ | ЛОГ | XH | неопр | ε ;
ЦЕЛ : := ЦЦЦ ; ЛОГ : := истина | ложь ;
H : := ЦЕЛ ; Ц : := 0 | 1 | 2 | 3 … | 9 ; X : := <бкв> | X <бкв> .

Принципиальным является то, что семантическая нагрузка на XH увеличивается, так как в данном случае XH является значением и одновременно косвенным именем всей сложной цепи, представляющей собой сложноструктурированное значение. При этом описание функций $\Phi 5^{(2)}, \Phi 6^{(2)}, \Phi 7^{(2)}$ на уровне значений значительно упростится. Данный прием косвенного именования через XH дает нам гибкий инструмент для управления областями видимости объектов при многоуровневом моделировании.

При статическом объявлении переменных в блоке программы для каждой переменной отводятся фиксированные места в памяти, например, если объявлена переменная имя цел v, то в памяти образуется структурированный список следующего вида:

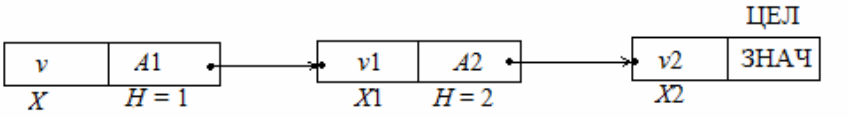

Рис. 1. Структура переменной в памяти

Адрес $A1$ отождествлен с переменной X и указывает или именует место в памяти следующего элемента списка. В данном случае $A1$ (при $H = 1$) является адресом второго элемента списка $(v1, A2)$. Адрес $A2$ (при $H = 2$) является адресом третьего элемента списка. Данный алгоритм обрабатывает объявления переменных и формирует ОДСП. ОДСП имеет ограничения, продиктованные структурой представления переменных внутри памяти. В качестве примера ограничения ОДСП можно сформулировать следующую лемму. Пусть τ_0 принадлежит набору элементарных типов {цел, вещ, компл, литера, проц, лог}.

Лемма 2. Если переменная X в программе объявлена в виде «имя $^n \tau_0 X$», то в соответствии с алгоритмической моделью $\Omega = \{\Phi 0^{(1)}, \ldots, \Phi N^{(1)}, \Phi 0^{(2)}, \ldots, \Phi N^{(m)}\}$ формируется список вида $X * 1 \# X1 * 2 \# \ldots \# XH * \tau_0$, где $H = n + 1$.

Доказательство проводится методом математической индукции в соответствии с алгоритмами $(\Phi 2^{(2)}, \Phi 5^{(2)}, \Phi 6^{(2)}, r)$, которые приведены в разделе 3.

Рассмотрим операции приведения данных от одного вида к другому. Операция привидения формально описывает логистику типов данных для упрощения построения алгоритмов над типами данных. Для этого рассмотрим пример в виде фрагмента программы на ASPLE:

начало {к$_0$} имя^2 цел l ; цел v ; {к$_1$} $l := 2$; $v := 15$; {к$_2$} $l := v$ {к$_3$} конец.

В фигурных скобках мы расставили точки для комментариев.

{к$_0$}: в данной точке состояние памяти будет следующим $\varepsilon\,\omega F$, где ε — пустая память.

{к$_1$}: состояние памяти согласно леммам 1, 2 будет:

$$l * 1 \# l1 * 2 \# l2 * 3 \# l3 * \text{неопр} \# v * 1 \# v1 * \text{неопр} \#$$

{к$_2$}: $$l * 1 \# l1 * 2 \# l2 * 3 \# l3 * 2 \# v * 1 \# v1 * 15 \#$$

{к$_3$}: в данной точке типы заданных аргументов для операции присваивания отличаются.

На начальном этапе рассматривается данный вопрос на уровне видов. В данной операции участвуют на уровне видов следующие функции: операция присваивания $\Phi 7^{(1)}$ = (знач x, y) знач: (имя ВИД, ВИД) имя ВИД и операция преобразования видов $g_1^{(1)}$ = (имя ВИД) ВИД. На вход $\Phi 7^{(1)}$ первый фактический аргумент l имеет тип имя^2 цел, а второй фактический аргумент v имеет тип цел. Однако формальные аргументы функции $\Phi 7^{(1)}$ имеют типы: первый аргумент имя ВИД, а второй аргумент ВИД. Непосредственное сопоставление фактических аргументов к формальным не удается: у первого аргумента одно лишнее имя. В таком случае система автоматически однократно вызывает вспомогательную функцию $g_1^{(1)}$ = (имя ВИД) ВИД и преобразует первый аргумент к виду имя цел. И при этом когда на уровне значений запускается функция $\Phi 7^{(2)}$, то она автоматически аналогично однократно вызовет для первого аргумента вспомогательную функцию на уровне значений

$$g_2^{(2)} = (\text{конт } t,\ \text{знач } x)\ \text{знач} : (M \# XH * P M^{(1)} \omega F,\ XH)\ P.$$

Если бы переменная l имела вид «имя^n цел l», то в таком случае вспомогательная функция на уровне видов $g_1^{(1)}$ = (имя ВИД) ВИД встраивалась бы в текущую суперпозицию на вход операции присваивания автоматически $(n-1)$ раз, и соответственно на уровне значений функция $g_2^{(2)}$ = (конт t, знач x) знач : $(M \# XH * P M^{(1)} \omega F,\ XH)\ P$ вызывалась бы автоматически $(n-1)$ раз. Другими словами, вычисление на уровне видов дополняет суперпозицию базисных функций необходимым количеством вспомогательных функций для преобразования видов, чтобы на уровне значений мы получили изоморфную суперпозицию базисных функций, являющуюся моделью программы.

Если обозначить за x текст программы, F_1 — суперпозицию базисных функций на уровне видов, F_2 — суперпозицию базисных функций на уровне значений, $\{g_i^{(1)}\}$ и $\{g_i^{(2)}\}$ — функции преобразования видов, то схему работы построенной модели можно представить следующим образом

$$x \xrightarrow[\text{синтаксического разбора}]{\text{алгоритм}} F_1^{(1)} \xrightarrow{\text{исполнение } F_1^{(1)}} F_1^{(1)} + \{g_i^{(1)}\} \xrightarrow[\text{отображение}]{\text{изоморфное}} F_2^{(2)} + \{g_i^{(2)}\}$$

Рис. 2. Схема работы построенной модели

Готовая для исполнения программа порождает процесс моделирования динамической системы.

3. Алгебраическая модель языка программирования

Используя вышеописанное, построим полное формальное определение синтаксиса и семантики языка ASPLE.

Грамматика $G_{-1} = (V_{T-1}, V_{N-1}, P_{-1}, E)$ — модель памяти типа конт.

1. $E ::= M \omega F$; M — внутренняя память, F — входной и выходной файлы;
2. $M ::=$ ВИД $X \# M \mid X$ * ЗНАЧ $\# M \mid \varepsilon$;
3. $F ::= B \S B$;
4. $B ::= B$ * ЗНАЧ $\mid \varepsilon$;

Начальное состояние памяти: $E = \varepsilon\, \omega\, B \S \varepsilon$, где B — файл ввода.

$V_{T-1} = \{\omega,\ \S,\ *, \#, \varepsilon, X, \text{ВИД}, \text{ЗНАЧ}\}$, $V_{N-1} = \{E, M, F, B\}$.

Грамматика $G_0 = (V_{T0}, V_{N0}, P_0, <\textit{программа}>)$ — синтаксис.

<*программа*> : : = начало <*описания*> ; <*операторы*> конец *Ф*1
<*описания*> : := <*описание*> ; *Ф*2 | <*описания*> <*описание*> ; *Ф*3
<*операторы*> : := <*оператор*> ; *Ф*0 | <*операторы*> <*оператор*> ; *Ф*4
<*описание*> : := <*описание*> , *X* *Ф*5 | ВИД *X* *Ф*6
<*оператор*> : := <*оператор присваивания*> *Ф*0 | <*условный*> *Ф*0 |
<*цикл*> *Ф*0 | <*условный*> *Ф*0 | <*ввод-вывод*> *Ф*0
<*оператор присваивания*> : := <*идент*> := <*выражение*> *Ф*7
<*условный*> : : если <*выражение*> то <*операторы*> все *Ф*8
если <*выражение*> то <*операторы*> иначе <*операторы*> все *Ф*9
<*цикл*> : := пока <*выражение*> цикл <*операторы*> конец *Ф*10
<*ввод-вывод*> : := *ввод* *X* *Ф*11 | *вывод* <*выражение*> *Ф*12
<*выражение*> : := <*множитель*> *Ф*0 |
<*выражение*> + <*множитель*> *Ф*13

<*множитель*> : := <*первичное*> $\Phi 0$ | <*множитель*> * <*первичное*> $\Phi 14$
<*первичное*> : := <*идент*> $\Phi 0$ | <*константа*> $\Phi 15$ |
(<*выражение*>) $\Phi 0$ | (<*сравнение*>) $\Phi 0$
<*сравнение*> : := <*выражение*> = <*выражение*> $\Phi 16$ |
<*выражение*> ≠ <*выражение*> $\Phi 17$
X : := <*идент*> $\Phi 18$
<*константа*> : := ЛОГ | ЦЕЛ
<*идент*> : := <*буква*> | <*идент*> <*буква*>
<*буква*> : := *а* | *б* | *в* | *г*…| *я* .

Функции Φi $(i = 1,...,18)$ на уровне видов:

$\Phi 0$ = (знач x) знач: (ПРАВИД) ПРАВИД
$\Phi 1$ = (конт t , знач x, y) конт: $(M\ \omega F, \varepsilon,$ ПРАВИД) $\varepsilon\ \omega F$
$\Phi 2$ = (конт t , знач x) конт: (E, M) $\Phi 3(E, \varepsilon, M)$
$\Phi 3$ = (конт t , знач x, y) конт: $((M$ ВИД $X \# E,\ \varepsilon,$ ВИД$^{(1)}$ $X \# M^{(1)})\ \perp |$
$(E, \varepsilon,$ ВИД $X \# M)$ $\Phi 3($ имя ВИД $X \# E, \varepsilon, M) \mid (E, \varepsilon,\ \varepsilon)$ знач (ε))
$\Phi 4$ = (знач x, y) знач: (ПРАВИД, ПРАВИД$^{(1)}$) ПРАВИД$^{(1)}$
$\Phi 5$ = (знач x , текст y) знач: (ВИД $X \# M, X^{(1)}$) ВИД $X \# M$ ВИД $X^{(1)}$ #
$\Phi 6$ = (текст x, y) знач: (ВИД, X) имя ВИД X #
$\Phi 7$ = (знач x, y) знач: (имя ВИД, ВИД) имя ВИД
$\Phi 8$ = (знач x, y) знач : (лог ПРАВИД) ничто
$\Phi 9$ = (знач x, y) знач: ((лог, ПРАВИД, ПРАВИД) ПРАВИД | (лог, ПРАВИД, ПРАВИД$^{(1)}$) ничто))
$\Phi 10 = \Phi 8$; $\Phi 11 = \Phi 12 = \Phi 0$
$\Phi 13$ = (знач x, y) знач: ((цел, цел) цел | (лог, лог) лог)
$\Phi 14 = \Phi 13$; $\Phi 15$ = (текст x) знач: ((ЦЕЛ) цел | (ЛОГ) лог)
$\Phi 16$ = (знач x, y) знач: ((ВИД, ВИД) лог) ; $\Phi 17 = \Phi 16$
$\Phi 18$ = (конт t , текст x) знач: $(M$ ВИД $X \# E,\ X)$ ВИД

Функции Φi $(i = 1,...,18)$ на уровне значений:

$\Phi 0$ = (знач x) знач: (ПРАЗНАЧ) ПРАЗНАЧ
$\Phi 1$ = (конт t , знач x, y) конт: $(M\ \omega\ F,\ \varepsilon,$ ЗНАЧ) $\varepsilon\ \omega\ F$
$\Phi 2$ = (знач x) знач: (ВИД & M) M
$\Phi 3$ = (конт t , знач x, y) конт, функ: $(M\ \omega\ F, M^{(1)},$ ВИД & $M^{(2)})$
$MM^{(1)}M^{(2)}\omega\ F,$ знач (ε)
$\Phi 4$ = (знач x, y) знач: (ПРАЗНАЧ, ПРАЗНАЧ$^{(1)}$) ПРАЗНАЧ$^{(1)}$
$\Phi 5$ = (знач x , текст y) знач: (ВИД & M , X) r (ВИД, MX * 1# X1*, ВИД)

$\Phi 6$ = (текст x, y) знач: (ВИД, X) r (имя ВИД, X *1#X1*, ВИД)

r = ((ВИД, MXH , имя ВИД$^{(1)}$)

r (ВИД, MXH * $eval(H+1)$#X $eval(H+1)$*, ВИД$^{(1)}$)

| (ВИД, MXH, ВИД$^{(1)}$) ВИД & MXH *неопр#)

$\Phi 7$ = (конт t , знач x, y) конт: (MXH * ЗНАЧ #М$^{(1)}F$,

XH , ЗНАЧ$^{(1)}$) MXH * ЗНАЧ$^{(1)}$ #$M^{(1)}\omega F$

$\Phi 8$ = (знач x , функ y) знач : ((истина, y) y) | (ложь, y) ε)

$\Phi 9$ = (знач x , функ y, z) знач : ((истина, y, z) y) | (ложь, y, z) z);

$\Phi 10$ = (функ x, y) знач: (x, y) $r_1(x, x, y)$

r_1 = (знач x , функ y, z) знач: ((истина, y, z) $f_4(z, r_1(y, y, z))$ | (ложь, $y, z,$) ε;

$\Phi 12$ = (конт t , знач x) конт: ($M\omega B$ § $B^{(1)}$, ЗНАЧ) МωB § $B^{(1)}$ * ЗНАЧ

$\Phi 13.1$ = (знач x, y) знач: (ЦЕЛ, ЦЕЛ$^{(1)}$) eval (ЦЕЛ+ЦЕЛ$^{(1)}$)

$\Phi 13.2$ = (знач x, y) знач: ((истина, ЛОГ) истина | (ложь, истина) истина | (ложь, ложь) ложь)

$\Phi 14.1$ = (знач x, y) знач: (ЦЕЛ, ЦЕЛ$^{(1)}$) eval (ЦЕЛ*ЦЕЛ$^{(1)}$)

$\Phi 14.2$ = (знач x, y) знач: ((ложь, ЛОГ) ложь | (ЛОГ, ложь) ложь

| (истина, истина) истина)

$\Phi 15$ = (текст x) знач: (ЗНАЧ) ЗНАЧ;

$\Phi 16$ = ((знач x, y) знач: (ЦЕЛ, ЦЕЛ) истина | (ЦЕЛ, ЦЕЛ$^{(1)}$) ложь)

$\Phi 17$ = ((знач x, y) знач: (ЦЕЛ, ЦЕЛ) ложь (ЦЕЛ, ЦЕЛ$^{(1)}$) истина

| (ЛОГ, ЛОГ) ложь (ЛОГ+ЛОГ$^{(1)}$) истина)

$\Phi 18$ = (конт t , текст x) знач: (MX * ЗНАЧ § $M^{(1)}\omega F, X$) ЗНАЧ .

Заключение

В данной работе разработан подход к компьютерному моделированию динамических систем. Для этого был построен класс функций, являющийся алгебро-логической моделью языка программирования общего назначения. Тогда любая компьютерная программа, предназначенная для моделирования динамической системы, порождает моделирующий процесс. При данном подходе указанная программа представляется в виде алгебраического выражения (терма) универсальной алгебры, являющейся семантикой языка программирования. Имея терм для моделирования динамической системы, можно сформулировать и математически корректно доказать те или иные теоремы о процессах моделирования.

Литература

1. Интеллектное управление динамическими системами / С. Н. Васильев [и др.]. М.: Физико-математическая литература, 2000. 352 с.
2. Семантика языков программирования: пер. с англ. / под ред. В. М. Курочкина. М.: Мир, 1980. 395 с.
3. Тузов В. А. Математическая модель языка. Л.: Изд-во ЛГУ, 1984. 176 с.
4. Тузов В. А. Подход к построению универсальной схемы языка. Синтаксис // Программирование. 1980. № 5. С. 17–25.

ON A CERTAIN APPROACH TO MODELING DYNAMIC SYSTEMS

Darima D. Nikolaeva
Research Assistant,
Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: darisha89@yandex.ru

Dashadondok Sh. Shirapov
Dr. Sci. (Phys. and Math.), Prof.,
Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: shir48@mail.ru

Vyacheslav I. Antonov
Cand. Sci (Phys. and Math.), A/Prof.,
Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: antonov.imi@mail.ru

When developing complexes of computer programs modeling various dynamic systems, it is often required to build mathematical models of a particular subject area. In the article we constructed such functions that it is possible to construct a superposition of functions (term) for any given programming language. Calculating the above term generates a computational process that occurs when the program is executed. If the program is designed to model a dynamic system, then the calculation of the algebraic term is an adequate simulation for the dynamic system functioning. Thus, an algebraic model of a programming language is developed for modeling of dynamic systems, where calculation of algebraic terms generates modeling process of dynamical systems. To construct these functions, it is necessary to accurately describe the domain of definition and the range of values of these functions. To construct domains of definition and ranges of values for these functions we used context-free grammars, and identification operation. In addition to these tools, the concept of a multi-level model, the concept of indirect naming (indirect addressing), recursion, and also some simple tools from the theory of algorithms and programming theory are used. Thus, a method of computer modeling of various dynamic systems, where an arbitrary program can be represented as an algebraic term of a universal algebra with a signature from the indicated functions, is found to be sufficiently broad in practical coverage.

Keywords: dynamic systems; modeling; mathematical model of language; universal algebra; context-free grammar; recursion; interpreter; semantics.

References

1. Vasil'ev S. N., Zherlov A. K., Fedosov E. A., Fedunov B. E. *Intellektnoe upravlenie dinamicheskimi sistemami* [Intelligent Control of Dynamic Systems]. Moscow: Fiziko-matematicheskaya literatura Publ., 2000. 352 p.

2. *Semantika yazykov programmirovaniya* [Semantics of Programming Languages]. Moscow: Mir Publ., 1980. 395 p. (Transl. from English)

3. Tuzov V. A. *Matematicheskaya model' yazyka* [Mathematical Model of Language]. Leningrad: Leningrad State University, 1984. 176 p.

4. Tuzov V. A. *Podkhod k postroeniyu universal'noi skhemy yazyka. Sintaksis. Programmirovanie* [Approach to Construction of a Universal Language Scheme. Syntax. Programming]. 1980. No. 5. Pp. 17–25.