КИБЕРНЕТИКА

В.И. ШИНКАРЕНКО, В.М. ИЛЬМАН, В.В. СКАЛОЗУБ

УДК 519.712.1:510.51:004.42.001

СТРУКТУРНЫЕ МОДЕЛИ АЛГОРИТМОВ В ЗАДАЧАХ ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ. I. ФОРМАЛЬНЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

Ключевые слова: *алгоритмические структуры, модель алгоритма, структура алгоритма, операции над алгоритмами, композиция алгоритмов.*

ВВЕДЕНИЕ

Алгоритмические свойства программного обеспечения (ПО) проявляются в процессе его создания, преобразования, представления и выполнения, а также в их результатах. Характеризуя современное ПО с его алгоритмической составляющей, отметим:

— многофункциональность, что предполагает объединение или композицию различных алгоритмов в единую программу, которая является неким общим алгоритмом;

— псевдопараллельное выполнение, дробление алгоритмов на порции и поочередное порционное выполнение алгоритмов. При этом эффективность алгоритма зависит от него самого, от алгоритма диспетчирования и других алгоритмов, выполняемых параллельно с ним, что требует изучения алгоритмов в их взаимосвязи;

— функционирование алгоритмов в системной программной среде — алгоритмы прикладного ПО используют алгоритмы системного ПО;

— выполнение алгоритмов на ЭВМ со сложными алгоритмами конвейеризации, кэширования и т.п., взаимодействие программно и аппаратно реализованных алгоритмов.

Программу символьных вычислений Maple, например, можно рассматривать как единый алгоритм, состоящий из ряда вычислительных алгоритмов, взаимодействующих с алгоритмами системного ПО и ПЭВМ.

Для моделирования алгоритмов и их анализа многие исследователи применяют алгебраический подход. Широко известны алгебры Глушкова [1–3], Дейкстры [4], формулы Ляпунова [5], схемы программ Янова [6, 7] и др. Модели позволяют изучить свойства алгоритмов и выработать некоторые практические рекомендации по их преобразованиям. Плодотворность такого подхода подтверждается достаточно большим количеством публикаций и результатов. Так, посредством алгоритмики Г.Е. Цейтлин показал возможности трансформации и синтеза алгоритмов [8]. К тому же существует направление алгоритмической алгебры [9, 10 и др.], связанное с представлением основных алгебраических построений в виде алгоритмически вычисляемых объектов.

С позиций прикладного программирования все исследования такого рода направлены на анализ и совершенствование отдельных алгоритмов. Они не позволяют в достаточной мере отобразить взаимосвязи алгоритмов, реализованных в виде современного ПО, в их общности.

В данной работе применяется подход к анализу алгоритмов в их совокупности. Заданные на множестве алгоритмов операции с соответствующей аксиоматикой образуют алгоритмическую структуру (АС), в которой объектами операций являются алгоритмы.

В рамках АС предлагается две взаимодополняющие формы модели алгоритмов. Первая основывается на структуре алгоритма и ассоциируется с представлением алгоритмов, вторая — на множестве путей алгоритма и ассоциирована с его выполнением.

Характеристические особенности алгоритмов и ПО, их представление и выполнение требуют особого подхода при построении соответствующих моделей. Для моделирования алгоритмов и ПО в работе предложен дедуктивный, конструктивный подход на основе формальной АС. Формальная структура с априорными свойствами и механизмами выполнения базовых алгоритмов является подходящей моделью для представления и выполнения прикладных алгоритмов. Задание конкретных АС позволяет представлять прикладные алгоритмы в формульном виде, удобном для преобразований и анализа их структуры-строения. Рассмотренные свойства АС и ее подструктур позволяют решать ряд задач эквивалентности и сходства алгоритмов и др.

В представленной работе вначале определяется конструктивный формализм — АС. Во второй части будут показаны возможности ее применения для моделирования преобразований алгоритмов в процессе разработки ПО. Наконец, проиллюстрирована целесообразность применения АС для решения практических задач моделирования алгоритмов в контексте выполнения программ.

АЛГОРИТМЫ И АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

Алгоритм является первичным понятием. Постулируем его и важные для дальнейшего изложения свойства алгоритмов.

Определим алгоритм одной из известных формулировок.

Аксиома 1. Алгоритм — это точный набор инструкций, описывающий конечную последовательность действий некоторого исполнителя для достижения определенного результата.

Отметим два существенных момента аксиомы: во-первых, в ней отражены два аспекта алгоритма — представление и выполнение; во-вторых, для предлагаемых моделей важно, что алгоритм определяется через последовательность действий, а не через порядок действий. Порядок допускает множество последовательностей действий, одинаковых с точки зрения результата, и больше подходит для представления параллельных алгоритмов.

Аксиома 2. Исполнитель способен выполнять конечное множество различных инструкций — базовых алгоритмов.

Будем обозначать такое множество W.

Согласно Геделю–Клини [11] любые алгоритмически реализуемые функции (частично рекурсивные) могут быть построены на основе базовых: нуль-функции, функции следования, функции выбора аргументов, с использованием суперпозиции, рекурсии и минимизации. Отсюда множество алгоритмов, реализующих нуль-функцию, функцию следования, функцию выбора аргументов, функцию определения минимума, является базовым для представления любых вычислительных алгоритмов (с учетом рекурсии и подстановки).

Однако представление алгоритмов на основе этих функций не практично, поэтому более приемлемы другие базовые алгоритмы. При представлении алгоритмов в виде блок-схем или схем Насси–Шнейдермана множество базовых алгоритмов может включать, например, алгоритмы, реализующие арифметические операции, операции булевой алгебры и другие стандартные и нестандартные функции. Для алгоритмов, представленных в виде программы для ЭВМ, базовый набор алгоритмов может включать алгоритмы, реализующие множество базовых операций языка программирования. В представлении алгоритма в виде исполнительного кода базовый набор алгоритмов может включать алгоритмы, аппаратно реализующие набор команд процессора.

Аксиома 3. Любой алгоритм A реализует следующее: для любого элемента x из множества определений $X(A)$ алгоритм A за конечное число шагов выполнения находит некоторый элемент y из множества значений $Y(A)$. При этом $\forall y \in Y \ \exists x \in X$ такой, что, применив к нему алгоритм A, результатом будет y.

Для обозначения алгоритмов применим следующую нотацию: $A\big|_{x\in X}^{y\in Y}$ или $A\big|_{X}^{Y}$. Здесь A — идентификатор алгоритма, который отличает этот алгоритм от других и может быть представлен именем, адресом или выражением на естественном языке. Алгоритм может иметь несколько идентификаторов: $(A,B)\big|_{x\in X}^{y\in Y}$.

Аксиома 3 устраняет неоднозначность при толковании области определений и значений, исключает случаи зацикливания и нерезультативного выполнения (прерывания).

Элементы $x \in X$ и $y \in Y$ идентифицируют входные и выходные данные однократного выполнения алгоритма. Они могут представлять собой некоторую совокупность и структуру данных разного типа, а также последовательность таких данных, распределенных во времени.

В процессе выполнения алгоритма могут быть получены промежуточные результаты, полный набор их идентификаторов обозначим q, а множество значений — $Q(A)$. Пусть набор $\langle x, q, y \rangle$ составляет словарь алгоритма, тогда множество допустимых данных алгоритма на этом словаре есть $Z(A) \subseteq X(A) \times Y(A) \times Q(A)$.

Уточним природу множеств $X(A)$, $Y(A)$ и базовых алгоритмов. Будем исходить из того, что исполнителем алгоритмов современного ПО являются ЭВМ и управляющие машины и механизмы. Условно можно выделить три категории выполняемых ими алгоритмов: вычислительные, обмена и управляющие.

Вычислительные алгоритмы, как известно, реализуют некоторую частично рекурсивную функцию $y = f(x)$.

Алгоритмы обмена предназначены для пересылки данных из одного места в другое или с одного запоминающего устройства в другое: из оперативной в кэш-память или память процессора, с магнитного диска в оперативную память, между запоминающими устройствами различных ЭВМ и т.п. При этом достаточно сложные алгоритмы (например, [12, с. 125]) реализуют тривиальную функцию присваивания $y = x$.

Базовыми алгоритмами алгоритма обмена с оперативной памятью могут быть синхронизация шины и памяти, регенерация данных в памяти и т.п., т.е. невычислительные алгоритмы.

Управляющие алгоритмы характерны тем, что результатом их работы есть не данные, а действия или предметы материального мира. Хотя промежуточными результатами можно считать управляющие сигналы и трактовать такие алгоритмы как вычислительные. Они выполняются управляющими устройствами: роботами, станками с ЧПУ, периферийными устройствами ЭВМ. Например, результатом работы алгоритма печати есть не данные, а их результат — распечатанная страница, результатом алгоритма управления роботом — последовательность его манипуляций и т.п.

Подавляющее большинство алгоритмов представляют собой некоторую смесь алгоритмов этих трех категорий. Исходя из сказанного, элементами множеств $X(A)$ и $Y(A)$ могут быть как традиционные числовые данные, так и действия и предметы материального мира. Не нарушая общности изложения, в дальнейшем эти элементы будем называть данными и вкладывать в них более широкий смысл.

Предлагаемый в работе подход на основе формализма АС существенно отличается от алгоритмических алгебр возможностями исследований алгоритмов с учетом природы базовых алгоритмов и их результатов.

Далее рассмотрим АС. Формальная структура задается упорядоченной тройкой с основным множеством V, сигнатурой Σ и аксиоматикой Λ [13]:

$$C = \langle V, \Sigma, \Lambda \rangle. \qquad (1)$$

Структура (1) не является универсальной, так как применение операций сигнатуры Σ к элементам из множества V в соответствии с аксиоматикой Λ не предполагает принадлежности результата множеству V. К тому же результат выполнения операций сигнатуры Σ может качественно отличаться от элементов множества V. Так, для грамматических структур [13] V — это алфавит, операции сигнатуры — конкатенации и подстановки, а результат операций — цепочки формального языка.

В этом проявляется конструктивизм формальной структуры. Формальная грамматическая структура позволяет конструировать свободный язык [2], свободную алгебраическую полугруппу, формальный язык.

Определение 1. Алгоритмической структурой (АС) будем называть многоосновную структуру — расширение формальной структуры (1)

$$C_A = \langle M, V, \Sigma, \Lambda \rangle, \tag{2}$$

где $V = \{A_i^0\}$ — конечное множество образующих алгоритмов, заданное на $M = \bigcup_{A_i^0 \in V} (X(A_i^0) \cup Y(A_i^0))$ — носителе, в общем случае неоднородном множестве.

Особенностью формальной АС является то, что ее операции определены на множестве алгоритмов. В рамках АС можно построить замкнутое множество алгоритмов и соответствующую многоосновную алгебру [2].

Множество всех возможных сконструированных на основе АС алгоритмов обозначим $\Omega(C)$. Естественно $V \subset \Omega(C)$.

Множество V может включать множества базовых алгоритмов W одного или нескольких исполнителей. Если множество образующих алгоритмов V совпадает с множеством базовых алгоритмов некоторого исполнителя, то такую АС будем называть базовой алгоритмической структурой (БАС).

Определение 2. Структуру $C_1 = \langle M_1, V_1, \Sigma_1, \Lambda_1 \rangle$ назовем подструктурой структуры $C_A = \langle M, V, \Sigma, \Lambda \rangle$ и обозначим $C_1 \prec C_A$, если $M_1 \subseteq M, V_1 \subseteq V, \Sigma_1 \subseteq \Sigma, \Lambda_1 \subseteq \Lambda$.

Основные особенности АС учтены в представленной далее АС с $\Omega(C)$ — некоторым подмножеством структурированных алгоритмов. Определена АС двуместными операциями $\Sigma = \{"\cdot", ":"\}$, заданными на множестве алгоритмов $V = \{A_i^0\}$ и аксиоматикой — аксиомами 1 … 5, определениями 1 … 10, описанными далее свойствами операций и общепринятыми правилами использования скобок.

ОПЕРАЦИИ НАД АЛГОРИТМАМИ

Введем операцию композиции как последовательное выполнение алгоритмов исполнителем. Будем полагать, что параллельное выполнение алгоритмов предусматривает выполнение нескольких алгоритмов несколькими независимыми исполнителями.

Определение 3. Операция композиции « $\cdot$ » определяет последовательное выполнение алгоритма $A_2|_{X_2}^{Y_2} \in V$ непосредственно после $A_1|_{X_1}^{Y_1} \in V$. Результатом операции есть алгоритм $A|_X^Y = A_1|_{X_1}^{Y_1} \cdot A_2|_{X_2}^{Y_2}$.

Здесь и далее знак «=» следует понимать в смысле «определено как».

При многократном применении операции композиции над алгоритмами $A_i|_{X_i}^{Y_i}$ имеем $A|_X^Y = \prod_i A_i|_{X_i}^{Y_i}$.

Выражение $A_1 \cdot A_2 \cdot A_3$ определяет выполнение операции композиции над алгоритмами $A_1 \cdot A_2$, в результате которой получен некоторый алгоритм A_4, и далее выполняется операция композиции $A_4 \cdot A_3$. При этом может быть, что алгоритм $A_4 \notin V$. Возможность многократного применения операции композиции определяется следующей аксиомой.

Аксиома 4. Операция композиции алгоритмов, заданная на множестве V алгоритмической структуры C_A, для любых двух алгоритмов $A_1, A_2 \in \Omega(C_A)$ определяется как последовательное выполнение последнего из образующих алгоритмов алгоритма A_1 и первого из образующих алгоритмов алгоритма A_2.

Аксиома 5. Входные данные алгоритма второго аргумента операции композиции могут включать выходные данные первого аргумента, а в случае многократного применения операции — выходные данные предыдущих алгоритмов, обратное (содержание во входных данных первого алгоритма выходных данных второго) недопустимо.

Порядок выполнения операции композиции можно изменить с помощью общепринятого аппарата скобок. Другой способ изменения порядка композиции — применение алгоритмов управления последовательностью выполнения.

Определим такую возможность посредством алгоритмов выбора, алгоритма и операции условного выполнения.

Определение 4. Алгоритм выбора $\widetilde{A}$ есть алгоритм с множеством определений $X_1 \cup X_2 \cup \ldots \cup X_N \quad (\forall i, j \in [1 \ldots N],\ X_i \cap X_j = \varnothing)$ и множеством значений $\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$ такой, что $\widetilde{A}\big|_{X_i}^{\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}} = \widetilde{A}\big|_{X_i}^{\{\theta_i\}}$.

В частных случаях АС множество значений может быть {0, 1} или {true, false}. В АС множество V может включать несколько алгоритмов выбора, что обуславливается возможностями исполнителей.

Определение 5. Алгоритм условного выполнения с множеством определения $\Theta_1 \times \Theta_2, \Theta_1 \subset \Theta, \Theta_2 = \{\theta_i \in \Theta\}$ заключается в выполнении следующей за ним операции условного выполнения $\vec{A}\big|_{\Theta_1, \Theta_2}^{\{:\}}$, которая и является значением алгоритма. Некоммутативная операция условного выполнения «:» — $\vec{A}\big|_{\Theta_1, \Theta_2}^{\{:\}} : B\big|_X^Y$ состоит в том, что если $\Theta_2 \subseteq \Theta_1$, то алгоритм $B\big|_X^Y$ должен выполняться, в противном случае — нет.

Алгоритм условного выполнения представляет тот случай, когда результатом выполнения алгоритма могут быть действия, как отмечалось ранее.

Основными инструкциям управления последовательностью действий в алгоритмах являются следование, ветвление и цикл. Покажем их представление с помощью введенных операций. Следование представлено операцией композиции, ветвление в алгоритме — выражением $\widetilde{A}_i\big|_{x \in X}^{y \in \{\theta_1, \theta_2\}} \cdot \vec{A}_j\big|_{y, \theta_1}^{\{:\}} : A_k\big|_{X_n}^{Y_n} \cdot \vec{A}_j\big|_{y, \theta_2}^{\{:\}} : A_l\big|_{X_m}^{Y_m}$. Многократное повторение алгоритма (цикл) запишем $\prod_{i=1}^{\infty} \widetilde{A}_i\big|_{x \in X}^{y \in \{\theta_1, \theta_2\}} \cdot \vec{A}_j\big|_{y, \theta_1}^{\{:\}} : A_k\big|_{X_n}^{Y_n}$. Согласно аксиоме 3, начиная с некоторого наперед неизвестного i у алгоритма условного выполнения $\theta_1 \neq y$ и алгоритм $A_k\big|_{X_n}^{Y_n}$ не выполняется.

Отметим, что в конкретных АС алгоритмы управления последовательностью выполнения могут определяться и другим образом (что дальше показано в примерах).

ФОРМЫ ПРЕДСТАВЛЕНИЯ (МОДЕЛИ) АЛГОРИТМОВ

Рассмотрим первую, структурную, форму модели, которая связана с аспектами создания и представления алгоритмов.

Определение 6. Пусть «⊕» — произвольная операция сигнатуры Σ и алгоритм $A\big|_X^Y \in \Omega(C_A)$ построен на основе операций Σ над образующими алгоритмами V. Последовательную запись образующих алгоритмов $(A_1^0 \oplus A_2^0 \oplus \ldots \oplus A_N^0)\big|_X^Y = A\big|_X^Y$ назовем структурой $\mathrm{Str}(A \setminus C_A)$ алгоритма $A\big|_X^Y$ в соответствующей АС C_A, или его структурной моделью.

Любую часть, ограниченную операциями композиции, началом или концом структуры алгоритма, можно выделить в отдельный алгоритм. По аналогии с подпрограммами назовем их подалгоритмами.

Рассмотрим вторую форму представления алгоритма, которая связана с последовательностью его конкретного выполнения при заданном значении $x \in X$ из области определения.

Определение 7. Пусть задан алгоритм $A|_X^Y \in \Omega(C)$ и композиция образующих алгоритмов $(A_1^0 \cdot A_2^0 \cdot \ldots \cdot A_N^0)\big|_{X^*}^{Y^*}$. Назовем эту композицию путем алгоритма $A|_X^Y$ и обозначим $P(A) = A_1^0 \cdot A_2^0 \cdot \ldots \cdot A_N^0$, если: $X^* \subseteq X$, $Y^* \subseteq Y$ и $X^* \neq \varnothing$, $Y^* \neq \varnothing$ $\forall x \in X^*$, при выполнении алгоритма образующие будут выполнены в заданной последовательности и выполнение $P(A)$ с любыми входными данными $x \in X^*$ даст тот же результат $y \in Y^*$, что и выполнение алгоритма A с теми же входными данными.

Путь $P(A)$ — представление некоторого алгоритма $P(A)\big|_{X^* \subseteq X}^{Y^* \subseteq Y}$. Заметим, что порядок следования составляющих в пути имеет существенное значение. В пути не может быть алгоритмов и операций управления последовательностью вычислений, кроме операции композиции. В то же время необходимые алгоритмы выбора в пути должны присутствовать.

Любой путь $P(A)$, согласно аксиоме 1, может содержать сколь угодно большое, но конечное количество составляющих. Множество различных путей алгоритма также конечно, что следует из конечного числа различных образующих алгоритмов. Количество различных путей ограничено числом $(m+1)^{n+1}$, где m — мощность V, n — максимальное количество алгоритмов в пути.

Путь $P(A)\big|_{X^*}^{Y^*}$ при любом $x \in X^*$ однозначно задает последовательность выполнения алгоритма для определения $y \in Y^*$.

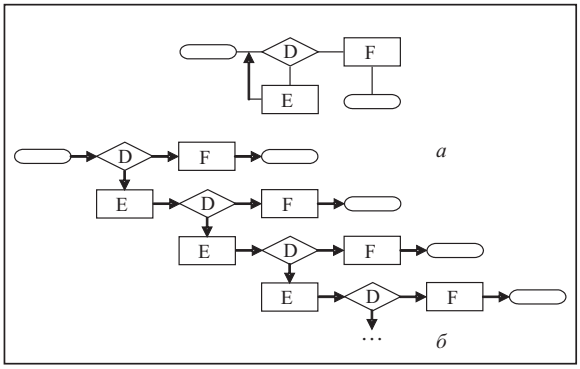

Рис. 1

Согласно определению 7 аналогом введенных путей являются пути дерева выполнения или *E*-деревья [14] (рис. 1). Пример дерева выполнения для алгоритма рис. 1, *а* дан на рис. 1, *б*.

Пути этого алгоритма: $P_1 = D \cdot F$; $P_2 = D \cdot E \cdot D \cdot F$; $P_3 = D \cdot E \cdot D \cdot E \cdot D \cdot F$ и т.д.

Назовем множество всех путей алгоритма A его путевой моделью и обозначим $\overline{P}(A) = \{P_i(A)\}$. При этом $\bigcup_{P_i(A) \in \overline{\overline{A}}} X(P_i(A)) = X(A)$ и для $(\forall j \neq i, X(P_j(A)) \cap X(P_i(A)) = \varnothing)$.

Заметим, что из пути можно исключить алгоритмы выбора. Так, для алгоритма рис. 1, *а* будут такие пути: $P_1 = F$; $P_2 = E \cdot F$; $P_3 = E \cdot E \cdot F$ и т.д. При этом для любого $x \in X(A)$ в результате выполнения пути будет найден тот же $y \in Y(A)$, что и для соответствующего пути с алгоритмами выбора. Однако множество таких путей определяет другой алгоритм, отличный от алгоритма, определяемого путями с алгоритмами выбора.

Путевая модель алгоритма достаточно громоздка и в некоторой степени избыточна. Однако порядок выполнения частей алгоритма в ряде случаев определяется

в процессе выполнения (например, при распараллеливании), и тогда такая форма модели представляется наиболее подходящей. К тому же, как видно из приведенного далее примера, по объемам она ненамного отличается от более близкой к традиционной форме — структуры алгоритма.

ОПЕРАЦИИ ДЛЯ ПАРАЛЛЕЛЬНО ВЫПОЛНЯЕМЫХ АЛГОРИТМОВ

Для моделирования алгоритмов параллельного выполнения множество образующих алгоритмов V алгоритмических структур C_A должно включать алгоритмы управления последовательностью выполнения, которые реализуют пуск и останов других алгоритмов и синхронизацию (сигналы, семафоры Дейкстры и т.п.).

Моделирование алгоритмов асинхронного параллельного выполнения только с использованием введенных выше операций невозможно, поскольку эти операции определяют последовательность выполнения действий, в то время как параллельными алгоритмами может предопределяться только их порядок. Разница заключается в том, что в последовательности для каждой составляющей известен непосредственный предшественник, а в заданном порядке — возможные предшественники.

Возможности путевой модели позволяют корректно задать операции объединения и пересечения алгоритмов

Определение 8. Назовем объединением алгоритмов $A\left|\begin{matrix}y_A\in Y_A\\ x_A\in X_A\end{matrix}\right.$ и $B\left|\begin{matrix}y_B\in Y_B\\ x_B\in X_B\end{matrix}\right.$ такой алгоритм $A\cup B=D\left|\begin{matrix}(y_A,y_B)\in Y_A\times Y_B\\ (x_A,x_B)\in X_A\times X_B\end{matrix}\right.$, что для любого пути $P(D)=\prod_{i=1}^{M}D_i^0$ алгоритма D существуют $P(A)=\prod_{i=1}^{N}A_i^0$ и $P(B)=\prod_{i=1}^{L}B_i^0$, где $D_i^0\in V, A_i^0\in V, B_i^0\in V$, такие что для рядом стоящих составляющих пути $P(D)$ возможно только одно из следующих условий:

- $D_j^0=A_i^0$ и $D_{j+1}^0=A_{i+1}^0$ $(i<M)$;
- $D_j^0=B_i^0$ и $D_{j+1}^0=B_{i+1}^0$ $(i<M)$;
- $D_j^0=A_i^0$, $D_{j+1}^0=B_k^0$ и $D_{j+2}^0=A_{i+1}^0$ $(i<M-1)$;
- $D_j^0=B_i^0$, $D_{j+1}^0=A_k^0$ и $D_{j+2}^0=B_{i+1}^0$ $(i<M-1)$.

При этом $M\le N+L$. Строгое неравенство возможно, если в обоих путях есть одинаковая последовательность базовых алгоритмов, имеющих одинаковые множества определения и значения.

Возможно также, что один из путей: $P(A)$ или $P(B)$, пуст.

Определение 9. Пересечением путей $P(A)$ и $P(B)$ алгоритмов A и B $P(D)=P(A)\cap P(B)$ есть один или несколько путей $P(D)=\prod_{i=1}^{M}D_i^0$, таких что для них существуют некоторые целые n и m, что $\forall i\in[1\ldots k]$ $D_i=A_{i+n}=B_{i+m}$, причем $X(D_1)=X(A_{1+n})=X(B_{1+m})$ и $Y(D_k)=Y(A_{k+n})=Y(B_{k+m})$.

Определение 10. Пересечение алгоритмов $A\cap B$ определяется множеством всех возможных путей, состоящим из попарных пересечений их путей $P_i(A)\cap P_j(B)$.

Множество путей пересечения алгоритмов A и B $P(A\cap B)$ можно разделить на четыре части:

— с общей начальной частью алгоритмов $P_i\ (A\cap B)\left|\begin{matrix}Z: Z\cap(Y(A)\cap Y(B)=\varnothing\\ X(A)\cap X(B)\end{matrix}\right.$;

— с общей средней частью $P_i\ (A\cap B)\left|\begin{matrix}Z_2: Z_2\cap(Y(A)\cap Y(B)=\varnothing\\ Z_1: Z_1\cap(X(A)\cap X(B)=\varnothing\end{matrix}\right.$;

— с общей конечной частью $P_i\ (A \cap B)\Big|^{Y(A)\cap Y(B)}_{Z:Z\cap(X(A)\cap X(B)=\varnothing}$;

— пути с общей частью алгоритмов, связанной с началом и концом $P_i\ (A \cap B)\Big|^{Y(A)\cap Y(B)}_{X(A)\cap X(B)}$.

Структурой алгоритма A, предполагающего параллельное выполнение его частей, в соответствующей АС будет его представление в виде последовательности образующих алгоритмов, связанных операциями композиции, управления вычислениями и объединения алгоритмов. Наличие непустого пересечения алгоритмов позволяет избежать излишнего параллелизма.

СВОЙСТВА ОПЕРАЦИЙ НАД АЛГОРИТМАМИ

Для рассмотрения свойств операций сигнатуры Σ необходимо выяснить некоторые свойства их операндов.

Ранее рассматривались алгоритмы над множествами $X(A), Y(A)$, не накладывая на множества никаких ограничений. В некоторых случаях $x \in X(A)$ можно представить в виде совокупности $x = (x_1, x_2, \ \dots\ x_n), x_i \in X_i$ и $X(A) = X_1 \times X_2 \times \dots \times X_n$, т.е. x состоит из независимых, никак не связанных друг с другом данных.

В прикладном программировании для многих алгоритмов множества $X(A), Y(A), Z(A)$ содержат структурированные данные, такие как массив, дерево, список, файл с последовательными записями и т.п. В таких случаях $x_i \in X_i$ состоят из ряда составляющих, связанных между собой отношениями порядка и вложенности. Обозначим отношения непосредственного следования как «,» — s, t, опосредованного следования «,,» — $s,, t$, непосредственной вложенности «.» — $s.t$, опосредованной вложенности «..» — $s..t$. В заданной последовательности данных $s_1, s_2, \dots, s_n$ соответствующий элемент может быть задан его номером (i), например, $S.(5)$ — пятый элемент одномерного массива S.

Будем говорить, что данные s и t зависимы $s \propto t$, если существует некоторая последовательность $s \sim s_1 \sim s_2 \sim \dots \sim t$ или $t \sim s_1 \sim s_2 \sim \dots \sim s$. Здесь «~» — одно из отношений следования или вложенности.

Отношения над данными определяются некоторой конкретной АС. Вопрос алгоритмического определения отношений над данными требует отдельного анализа.

Теперь рассмотрим свойства операций сигнатуры АС C_A.

Операция композиции алгоритмов обладает частичной коммутативностью, т.е.

$$A_1\Big|^{y_{1i}\in Y_{1i},\, i=1\dots n_2}_{x_{1i}\in X_{1i},\, i=1\dots n_1} \cdot A_2\Big|^{y_{2i}\in Y_{2i},\, i=1\dots n_4}_{x_{2i}\in X_{2i},\, i=1\dots n_3} = A_2\Big|^{y_{2i}\in Y_{2i},\, i=1\dots n_4}_{x_{2i}\in X_{2i},\, i=1\dots n_3} \cdot A_1\Big|^{y_{1i}\in Y_{1i},\, i=1\dots n_2}_{x_{1i}\in X_{1i},\, i=1\dots n_1}, \quad (3)$$

так как выполняется не на всем множестве алгоритмов $\Omega(C)$, а лишь на его части.

Укажем условия, при которых свойство (3) выполняется.

Условие 1. Множество образующих алгоритмов (и подалгоритмов) может быть разбито на классы таким образом, что свойство (3) выполняется для всех алгоритмов класса независимо от их входных и выходных данных.

Например, в такие классы могут быть выделены алгоритмы, реализующие арифметические операции сложения и вычитания или умножения и деления, заданные на числовом множестве определений.

Условие 2. Свойство (3) выполняется для любых алгоритмов, если входные данные одного и выходные другого независимые, т.е.

а) $\overline{\exists} i, j\colon y_{1i} \propto x_{2j}; \quad \forall i = 1\dots n_2, j = \dots n_3;$

б) $\overline{\exists} i, j\colon y_{2i} \propto x_{1j}; \quad \forall i = 1\dots n_4, j = \dots n_1.$

Операция композиции также частично ассоциативна. Свойство (4) выполняется, если выполняется условие 2а) для алгоритмов A_1 и A_2, A_1 и A_3, а также для A_3 и A_1 либо условие 1:

$$(A_1 \cdot A_2) \cdot A_3 = A_1 \cdot (A_2 \cdot A_3). \quad (4)$$

Операция условного выполнения не коммутативна, но дистрибутивна по отно-

шению к операции композиции: $\vec{A}_1:(A_2 \cdot A_3)=(\vec{A}_1:A_2)\cdot(\vec{A}_1:A_3)$.

ПРИМЕРЫ ПРЕДСТАВЛЕНИЯ АЛГОРИТМОВ

Для демонстрации возможностей моделирования алгоритмов приведем несколько примеров. На примере 1 показаны две формы представления алгоритмов.

Пример 1. Рассмотрим алгоритм нахождения приближенного значения корня уравнения $f(x)=0$ с заданной погрешностью ε. Функция $f(x)$ монотонно возрастающая в интервале $[a,b]$. Алгоритм представлен блок-схемой на рис. 2.

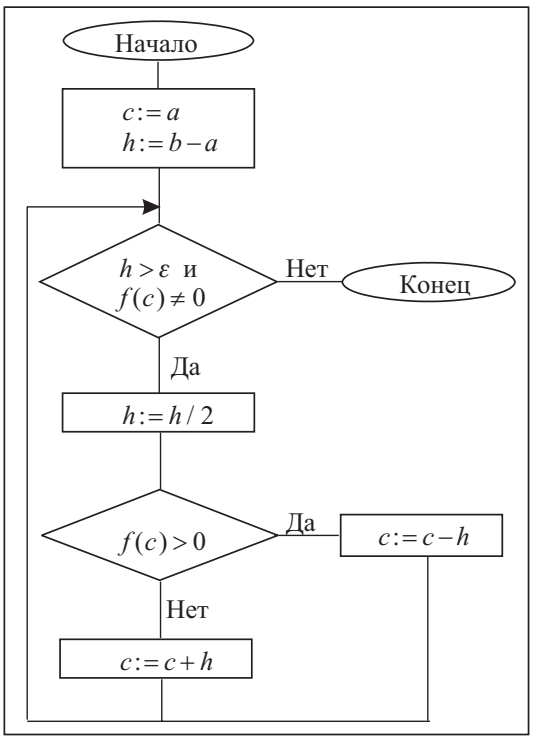

Рис. 2

Пусть конкретная АС C^* задана следующим образом: $M^*=\mathbb{R}$; множество образующих алгоритмов V^* содержит алгоритмы табл. 1 и $\vec{A}_9^0\big|_{a,b\in\{0,1\}}^{\{:\}}$ условного выполнения (определение 5); Σ^* — две двуместные операции « $\cdot$ » и «:»; аксиоматика Λ^* определена аксиомами 1 … 5, определениями 1…10 и свойствами операций.

Путями этого алгоритма, например, при $f(x)=x^2-4$, будут:

- путь без выполнения цикла —

$$P_1(A)=A_3^0\big|_a^c\cdot A_2^0\big|_{b,a}^h\cdot\widetilde{A}_6^0\big|_{h,\varepsilon}^{t_1}\cdot$$
$$\cdot A_4^0\big|_{x,x}^{t_2}\cdot A_2^0\big|_{t_2,4}^{t_3}\cdot\widetilde{A}_7^0\big|_{t_3,0}^{t_4}\cdot\widetilde{A}_8^0\big|_{t_1,t_4}^0;$$

- два пути при однократном выполнении цикла:

$$P_2(A)=A_3^0\big|_a^c\cdot A_2^0\big|_{b,a}^h\cdot\widetilde{A}_6^0\big|_{h,\varepsilon}^{t_1}\cdot A_4^0\big|_{x,x}^{t_2}\cdot A_2^0\big|_{t_2,4}^{t_3}\cdot\widetilde{A}_7^0\big|_{t_3,0}^{t_4}\cdot\widetilde{A}_8^0\big|_{t_1,t_4}^1\cdot\vec{A}_9^0\big|_{1,1}^{\{:\}}:A_5^0\big|_{h,2}^h\cdot$$

Таблица 1

Образующий алгоритм	Название	Описание
$A_1^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "+"$\big|_{a,b}^c$)	Сложение	$c:=a+b$
$A_2^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "–"$\big|_{a,b}^c$)	Вычитание	$c:=a-b$
$A_3^0\big\|_{a\in\mathbb{R}}^{c\in\mathbb{R}}$ (или ":="$\big|_a^c$)	Тождество	$c:=a$
$A_4^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "*"$\big|_{a,b}^c$)	Умножение	$c:=a*b$
$A_5^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "/"$\big|_{a,b}^c$)	Деление	$c:=a/b$
$\widetilde{A}_6^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или ">"$\big|_{a,b}^c$)	Сравнение на больше	$c:=\begin{cases}1, & \text{если } a>b\\ 0, & \text{если } a\le b\end{cases}$
$\widetilde{A}_7^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "≠"$\big|_{a,b}^c$)	Сравнение на неравенство	$c:=\begin{cases}1, & \text{если } a\neq b\\ 0, & \text{если } a=b\end{cases}$
$\widetilde{A}_8^0\big\|_{a,b\in\mathbb{R}}^{c\in\mathbb{R}}$ (или "&"$\big|_{a,b}^c$)	Логическое “и”	$c:=\begin{cases}1, & \text{если } a=1 \text{ и } b=1\\ 0, & \text{если } a\neq 1 \text{ или } b\neq 1\end{cases}$

$$\cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_6^0|_{t_3,0}^{1} \cdot \vec{A}_9^0|_{1,1}^{\{:\}} : A_2^0|_{c,h}^{c} \cdot \widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{0}$$

и

$$P_3(A) = A_3^0|_{a}^{c} \cdot A_2^0|_{b,a}^{h} \cdot \widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{1} \cdot \vec{A}_9^0|_{1,1}^{\{:\}} : A_5^0|_{h,2}^{h} \cdot$$

$$\cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \vec{A}_6^0|_{t_3,0}^{0} \cdot \vec{A}_9^0|_{0,0}^{\{:\}} : A_1^0|_{c,h}^{c} \cdot \widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{0}$$

и т.д.

В общем виде множество всех путей алгоритма $A\big|_{a,b,\varepsilon \in [0,2]\times[2,\infty)\times(0,1)}^{c\in[0,\infty)}$ можно представить как $\overline{P}(A) = \{P_{mnk}(A) \mid m,n,k = 0 \ldots \infty\}$, где:

$$P_{mnk}(A) = A_3^0|_{a}^{c} \cdot A_2^0|_{b,a}^{h} \cdot \prod_0^k \Bigg(\prod_0^m (\widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot$$

$$\cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{1} \cdot \vec{A}_9^0|_{1,1}^{\{:\}} : A_5^0|_{h,2}^{h} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_6^0|_{t_3,0}^{1} \cdot$$

$$\cdot \vec{A}_9^0|_{1,1}^{\{:\}} : A_2^0|_{c,h}^{c}) \cdot \Bigg(\prod_0^n (\widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{1} \cdot \vec{A}_9^0|_{1,1}^{\{:\}} : A_5^0|_{h,2}^{h} \cdot$$

$$\cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_6^0|_{t_3,0}^{0} \cdot \vec{A}_9^0|_{0,0}^{\{:\}} : A_1^0|_{c,h}^{c} \Bigg) \Bigg) \cdot$$

$$\cdot \widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{0}.$$

Приведем структуру алгоритма:

$$\mathrm{Str}(A \setminus C^*) = A_3^0|_{a}^{c} \cdot A_2^0|_{b,a}^{h} \cdot \prod_{i=1}^k (\widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{t_5} \cdot$$

$$\cdot \vec{A}_9^0|_{1,1}^{\{:\}} : (A_5^0|_{h,2}^{h} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_6^0|_{t_3,0}^{t_6} \cdot \vec{A}_9^0|_{t_6,1}^{\{:\}} : A_2^0|_{c,h}^{c} \cdot \vec{A}_9^0|_{t_6,0}^{\{:\}} : A_1^0|_{c,h}^{c})).$$

Обозначив

$$B_1|_{a,b,c,h,\varepsilon}^{c} = \prod_{i=1}^k (\widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot$$

$$\cdot \widetilde{A}_8^0|_{t_1,t_4}^{t_5} \cdot \vec{A}_9^0|_{1,1}^{\{:\}} : (A_5^0|_{h,2}^{h} \cdot A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3} \cdot$$

$$\cdot \widetilde{A}_6^0|_{t_3,0}^{t_6} \cdot \vec{A}_9^0|_{t_6,1}^{\{:\}} : A_2^0|_{c,h}^{c} \cdot \vec{A}_9^0|_{t_6,0}^{\{:\}} : A_1^0|_{c,h}^{c}),$$

алгоритм A можно представить с помощью композиции подалгоритмов

$$A = A_3^0|_{a}^{c} \cdot A_2^0|_{b,a}^{h} \cdot B_1|_{a,b,c,h,\varepsilon}^{c}$$

или $B_2|_{x}^{t_3} = A_4^0|_{x,x}^{t_2} \cdot A_2^0|_{t_2,4}^{t_3}$: $A = A_3^0|_{a}^{c} \cdot A_2^0|_{b,a}^{h} \cdot$

$$\cdot \prod_{i=1}^k (\widetilde{A}_6^0|_{h,\varepsilon}^{t_1} \cdot B_2|_{x}^{t_3} \cdot \widetilde{A}_7^0|_{t_3,0}^{t_4} \cdot \widetilde{A}_8^0|_{t_1,t_4}^{t_5} \cdot$$

$$\cdot \vec{A}_9^0|_{1,1}^{\{:\}} : (A_5^0|_{h,2}^{h} \cdot B_2|_{x}^{t_3} \cdot \widetilde{A}_6^0|_{t_3,0}^{t_6} \cdot \vec{A}_9^0|_{t_6,1}^{\{:\}} : A_2^0|_{c,h}^{c} \cdot \vec{A}_9^0|_{t_6,0}^{\{:\}} : A_1^0|_{c,h}^{c}).$$

Следующий пример показывает возможности построения структурных моделей алгоритмов обработки структурированных данных.

Пример 2. Алгоритм сортировки массива вещественных чисел — сортировка пузырьком [15]. Для краткости алгоритм в виде блок-схемы не приводим, определим лишь его структуру:

$$\mathrm{Str}(A_S\,|^{X,n}_{X,n}\backslash C^*) = A_3^0\,|^i_1 \cdot \prod_{k=1}^{\infty} \widetilde{A}_6^0\,|^{t_1}_{n,i} \cdot \vec{A}_9^0\,|^{\{:\}}_{t_1,1} : (A_1^0\,|^j_{i,1} \cdot A_1^0\,|^{t_2}_{n,1} \cdot$$

$$\cdot \prod_{i=1}^{\infty} \widetilde{A}_6^0\,|^{t_3}_{t_2,j}\, \vec{A}_9^0\,|^{\{:\}}_{t_3,1} : (\widetilde{A}_6^0\,|^{t_4}_{X.(i),X.(j)} \cdot$$

$$\cdot \vec{A}_9^0\,|^{\{:\}}_{t_4,1} : (A_3^0\,|^{t_5}_{X.(i)} \cdot A_3^0\,|^{X.(i)}_{X.(j)} \cdot A_3^0\,|^{X.(j)}_{t_5}) \cdot A_1^0\,|^j_{j,1}) \cdot A_1^0\,|^i_{i,1}))).$$

В теории и практике особое место занимают рекурсивные алгоритмы. Естественно рассмотреть возможности их моделирования.

Пример 3. Рассмотрим рекурсивный алгоритм вычисления факториала натурального числа n:

$$F(n) = \begin{cases} n \cdot F(n-1) \text{ при } n > 1, \\ 1 \text{ при } n = 1. \end{cases} \tag{5}$$

В АС C^{**} такой же, как C^*, но с множеством образующих алгоритмов V^{**}, таких же, как в примере 1, но заданных на множестве $M^{**} = \mathbb{N}$ натуральных чисел, этот алгоритм как композицию подалгоритмов можно представить в виде

$$A_r\,|^{n!}_n = \widetilde{A}_6^0\,|^{t_1}_{n,1} \cdot \vec{A}_9^0\,|^{\{:\}}_{t_1,1} : (A_2^0\,|^{q_1}_{n,1} \cdot A_r\,|^{f_1}_{g_1} \cdot A_4^0\,|^{n!}_{t_1,n}) \cdot \vec{A}_9^0\,|^{\{:\}}_{t_1,0} : A_3^0\,|^{n!}_1. \tag{6}$$

После многократной подстановки в правую часть выражения (6) значения его левой части и обобщения, с учетом свойств операций, имеем:

$$\mathrm{Str}(A_r\,|^{n!}_n \backslash C^{**}) = A_3^0\,|^{q_1}_n \cdot \prod_{i=1}^{n} (\widetilde{A}_6^0\,|^{t_i}_{q_{i,1}} \cdot \vec{A}_9^0\,|^{\{:\}}_{t_i,1} : A_2^0\,|^{q_{i+1}}_{q_i,1} \cdot \vec{A}_9^0\,|^{\{:\}}_{t_i,0} : A_3^0\,|^{f_n}_1) \cdot$$

$$\cdot \prod_{i=1}^{n-1} A_4^0\,|^{f_{n-i}}_{f_{n-i+1},q_{n-i}} \cdot A_3^0\,|^{n!}_{f_1}. \tag{7}$$

При этом формируются словарь $Z(A) = \{t_i\} \times \{q_i\} \times \{f_i\}$. Сравним (7) со структурой вычисления функции (5) итерационным алгоритмом:

$$\mathrm{Str}(A_r\,|^m_n \backslash C^{**}) = A_3^0\,|^i_1 \cdot A_3^0\,|^m_1 \cdot \prod_{i=1}^{n} \widetilde{A}_6^0\,|^{t_i}_{n,i} \cdot \vec{A}_9^0\,|^{\{:\}}_{t_i,0} : (A_4^0\,|^m_{m,i} \cdot A_1^0\,|^i_{i,1}) \cdot A_4^0\,|^{n!}_{m,i}.$$

В АС с множеством V, включающим базовые алгоритмы исполнителя — процессора ЭВМ, не обойтись без управляющего алгоритма, обеспечивающего изменение последовательности выполнения алгоритмов — алгоритма перехода. На следующем примере покажем, как такой алгоритм определяется и как при этом изменяется его структура.

Обратим внимание, что в предыдущих примерах структура алгоритма близка к известным алгебраическим моделям, а в следующем — к графовым. В этом проявляется универсальность алгоритмических грамматик в задачах моделирования алгоритмов.

Пример 4. В АС C^{***}, основанной на C^* с той разницей, что добавлен пустой алгоритм $A_{10}\,|^{\varnothing}_{\varnothing}$ и заменен алгоритм $\vec{A}_9^0$ следующим образом.

Множество определения алгоритма $\vec{A}_9^0$, как и ранее, $\Theta_1 \times \Theta_2$, $\Theta_1 \subset \Theta$, $\Theta_2 = \{\theta_i \in \Theta\}$. Множество значений состоит из двух уникальных имен алгоритмов. Выполнение алгоритма заключается в изменении последовательности выполнения операции композиции: если $\Theta_2 \subseteq \Theta_1$ то следующим должен быть выполнен первый алгоритм из списка в области значений, в противном случае — второй.

Структура алгоритма, приведенного на рис. 2, представлена в следующем виде:

$$\mathrm{Str}(A \backslash C^{***}) = \text{"="}\,|^c_a \cdot \text{"–"}\,|^h_{b,a} \cdot \prod_{i=1}^{k} (\text{">"}\,|^{t_1}_{h,\varepsilon} \cdot \text{"*"}\,|^{t_2}_{x,x} \cdot \text{"–"}\,|^{t_3}_{t_2,4} \cdot$$

$$\cdot"\neq"\Big|_{t_3,0}^{t_4}\cdot"\&"\Big|_{t_1,t_4}^{t_5}\cdot\vec{A}_9^{\,0}\Big|_{1,1}^{\{B_1,B_2\}}\cdot(B_1,"/")\Big|_{h,2}^{h}\cdot$$

$$\cdot"*"\Big|_{x,x}^{t_2}\cdot"-"\Big|_{t_2,4}^{t_3}\cdot">"\Big|_{t_3,0}^{t_6}\cdot\vec{A}_9^{\,0}\Big|_{t_6,1}^{\{B_3,B_4\}}\cdot$$

$$\cdot(B_3,"-")\Big|_{c,h}^{c}\cdot\vec{A}_9^{\,0}\Big|_{1,1}^{\{B_5,B_5\}}\cdot(B_4,"+")\Big|_{c,h}^{c}\cdot(B_5,A_{10})\Big|_{\varnothing}^{\varnothing})\cdot(B_2,A_{10})\Big|_{\varnothing}^{\varnothing}.$$

ЗАКЛЮЧЕНИЕ

Благодаря тому, что природа образующих алгоритмов в АС может быть различной и не ограничена вычислительными алгоритмами, открывается достаточно широкая область применения АС.

На основе АС может выполняться анализ естественных алгоритмов, созданных и реализуемых без участия человека, например алгоритмов формирования и развития растений. С ними связаны задачи восстановления объектов по известным частям.

Возможно изучение алгоритмов из других областей деятельности человека, таких как алгоритмы лечения отдельных болезней, борьбы с детской преступностью, процессов развития истории и т.п. Здесь интересны задачи структуризации алгоритмов и повышения их функциональной эффективности.

Особый интерес представляют алгоритмы, созданные системами искусственного интеллекта. Модели АС могут использоваться для решения проблемы о том, реализует ли искусственно созданный алгоритм метод, неизвестный создателям системы искусственного интеллекта. Они позволят разделить задачи построения оптимального, в некоторой области применения, алгоритма и оптимизации существующего.

Возможности применения АС к моделированию алгоритмических свойств программного обеспечения и процессов его разработки будут рассмотрены во второй части работы.

СПИСОК ЛИТЕРАТУРЫ

1. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм // Кибернетика. — 1965. — № 5 — С. 3–11.
2. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. — Киев: Наук. думка, 1989. — 376 с.
3. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгебро-алгоритмические модели и методы параллельного программирования. — Киев: Академпериодика, 2007. — 634 с.
4. Цейтлин Г.Е. Введение в алгоритмику. — Киев: Сфера, 1998. — 310 с.
5. Ляпунов А.А. К алгебраической трактовке программировани // Проблемы кибернетики. — 1962. — Вып. 8. — С. 235–243.
6. Ляпунов А.А. О логических схемах программ // Там же. – 1958. — Вып. 1. — С. 46–74.
7. Янов Ю.И. О логических схемах алгоритмов // Там же. – 1958. — Вып. 1. — С. 75–127.
8. Цейтлин Г.Е. Трансформационная сводимость и синтез алгоритмов и программ символьной обработки // Кибернетика и системный анализ. — 2006. — **42**, № 5. — С. 165–173.
9. Ноден П., Китте К. Алгебраическая алгоритмика. — М.: Мир, 1999. — 720 с.
10. Mishra B. Algirithmic Algebra, — N.Y.: Springer Verlag, 1993. — 416 p.
11. Катленд Н. Вычислимость. Введение в теорию рекурсивных функций. — М: Мир, 1983. — 256 с.
12. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.
13. Ільман В.М., Шинкаренко В.І. Структурний підхід до проблеми відтворення граматик // Проблемы программирования. — 2007. — № 1. — С. 5–16.
14. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. — М.: Мир, 1982. — 406 с.
15. Кнут Д.Э. Искусство программирования. Т 3. Сортировка и поиск. — М.: Издательский дом «Вильямс», 2000. — 832 с.

Поступила 28.05.2008