УДК 519.681

В.Г. АКУЛОВСКИЙ

АЛГЕБРА АЛГОРИТМОВ, БАЗИРУЮЩАЯСЯ НА ДАННЫХ

Ключевые слова: *модель ЭВМ, алгебра алгоритмов, данные, алгоритмические конструкции, операции алгебры алгоритмов, Д-операторы.*

ВВЕДЕНИЕ

Важнейшая, а во многих случаях определяющая, роль данных в процессе разработки алгоритмов и программ осмысливалась программистским сообществом постепенно. В результате последовательно сформировались два подхода к программированию: программирование от управления и программирование от данных. Принципиальное отличие между ними состоит в том, что в первом случае при создании алгоритма (программы) определяющей является разработка управляющих структур, а во втором — разработка структур данных. Воплощениями этих подходов стали парадигмы структурного и объектно-ориентированного программирования.

В настоящее время эти два подхода и соответственно две парадигмы программирования существуют достаточно независимо, более того, конкурируют между собой. Представляется очевидным, что имеется достаточно широкий класс программных систем, для разработки которых выбор подхода (парадигмы) неоднозначен. Это объясняется, по-видимому, тем, что каждый из подходов не универсален, а ориентирован на разработку своего класса программных систем (достаточно широкого) и не в полной мере отвечает специфике систем других классов. Кроме того, известные технологии, в рамках которых реализуются упомянутые парадигмы программирования, при всех своих достоинствах в подавляющем большинстве случаев не формализованы в достаточной степени.

Таким образом, актуальной представляется задача разработки и формализации третьего (промежуточного) подхода, позволяющего сочетать свойства двух перечисленных выше. При этом широко известным формальным аппаратом, ориентированным на разработку алгоритмов, является система алгоритмических алгебр (САА) [1] (продолжающая модифицироваться и развиваться [2, 3]), в рамках которой реализованы обе упомянутые парадигмы программирования.

Однако потенциал модели ЭВМ, положенной в основу упомянутого формального аппарата, использован не полностью. Это утверждение находит подтверждение в [4], где сказано, что модель ЭВМ Глушкова (при некоторых дополнительных предположениях) может трактоваться по-разному. Развивая эту идею, будем утверждать, что модификация указанной модели ЭВМ позволяет построить алгебраический аппарат с некоторыми заданными свойствами.

Исходя из этого утверждения, будем строить аппарат, располагающий средствами описания алгоритмов в такой форме, которая обеспечивала бы возможность одновременного и согласованного описания управляющих структур и обрабатываемых данных. При этом сформулируем дополнительные требования.

Для повышения надежности получаемого программного обеспечения необходимо, чтобы логическая составляющая формального аппарата обеспечивала определенность логических условий при любых значениях обрабатываемых данных и адекватную реакцию на некорректные данные. Поскольку алгебра алгоритмов Глушкова (ААГ) по порождающей мощности превосходит многие известные алгебраические аппараты [5], то сигнатура создаваемой алгебры должна быть не беднее сигнатуры ААГ.

Наконец, в целях повышения порождающей мощности создаваемого формального аппарата необходимо, чтобы на основе исходных (базовых) можно было строить производные алгоритмические конструкции, мощность (функциональность) которых превосходила бы исходные.

© В.Г. Акуловский, 2012

МОДЕЛЬ ЭВМ

Как известно, абстрактная модель ЭВМ Глушкова, предназначенная для описания выполнения произвольного алгоритма, представляет собой два взаимодействующих автомата: управляющий (У) и операционный (О). Поскольку для решения заявленной во введении задачи данные должны быть органично встроены в формальный аппарат, модифицируем абстрактную модель ЭВМ (в дальнейшем модель), оснастив ее памятью.

Будем полагать, что память автомата О представляет собой упорядоченное множество $\Delta^s = \{\delta_1, \delta_2, \ldots, \delta_n\}$, элементы которого — ячейки памяти. Каждая ячейка памяти δ_i принимает (хранит) в каждый момент времени одно значение $z_j \in \Psi_i$, где $\Psi_i = \{z_1, z_2, \ldots, z_j, \ldots, z_s\}$ — множество всех возможных значений, допустимых для хранения ячейкой памяти $\delta_i \in \Delta^s$.

Множество ячеек памяти $\Delta_i^s = \{\delta_{m+1}, \delta_{m+2}, \ldots, \delta_{m+p}\}$ (в частном случае $\Delta_k^s = \{\delta_r\}$) такое, что $\Delta_i^s \subseteq \Delta^s$ и $\Delta_i^s \cap \Delta_f^s = \varnothing$ при $i \neq f$, назовем элементом памяти. Элемент памяти Δ_i^s в каждый момент времени принимает (хранит) некоторый кортеж значений $Z_h = \{z_1, z_2, \ldots, z_p\}$ такой, что для $\forall z_k \in Z_h$ выполняется соотношение $z_k \in \Psi_{\delta_{m+k}}$, т.е. каждая ячейка памяти, образующая элемент памяти, в каждый момент времени хранит одно из возможных для нее значений. При $i \neq j$ допустимо, что как $z_i = z_j$, так и $z_i \neq z_j$.

Будем полагать, что все элементы памяти и значения, хранимые этой памятью, определены и доступны на каждом шаге вычислительного процесса; эту память назовем статической.

Таким образом, статическую память операционного автомата Δ^s будем рассматривать как множество элементов памяти $\Delta^s = \{\Delta_1^s, \Delta_2^s, \ldots, \Delta_r^s\}$, а это множество элементов памяти, принявших в некоторый момент времени кортеж значений Z_i^s, — как текущее состояние статической памяти (в дальнейшем изложении под обозначением Δ^s будем понимать именно это).

Память операционного автомата дополним еще одним видом памяти, которая обеспечивает краткосрочное хранение принятых значений. Эта память представляет собой элемент памяти Δ^d такой, что $\Delta^d \cap \Delta^s = \varnothing$, и она принимает некоторый кортеж значений, аналогичный рассмотренному выше случаю. Будем полагать, что элемент памяти Δ^d и соответственно хранимые им значения определены (доступны) не на каждом шаге вычислительного процесса. Эту память назовем динамической. Элемент памяти Δ^d, когда он в некоторый момент времени определен и принял кортеж значений Z_j^d, представим как текущее состояние динамической памяти (в дальнейшем изложении под обозначением Δ^d будем понимать именно это). Свойства такого вида памяти, определяемые указанным ограничением, будут понятны из дальнейшего изложения.

Заметим, что статическую память можно интерпретировать как оперативную, а динамическую — как регистровую (сверхоперативную).

Из изложенного видно, что текущее состояние рассматриваемой модели, которое обозначим Δ, определяется текущими состояниями статической Δ^s и динамической Δ^d памяти, т.е. $\Delta = \{\{\Delta^s\}, \{\Delta^d\}\}$. При этом состояние $\Delta^s \subseteq \Delta$ выполняется в течение всего времени вычислительного процесса в силу статичности этого типа памяти, в то время как состояние $\Delta^d \subset \Delta$ выполняется не всегда в силу ее динамичности.

Кроме того, будем полагать, что операционный автомат вырабатывает некоторое логическое условие α, которое характеризует состояние статической или динамической памяти. Это условие определено только на некоторых этапах функционирования модели, нигде не хранится, но в случае его определения является составляющей текущего состояния модели, т.е. $\{\alpha\} \subset \Delta$.

Наконец, будем полагать, что в памяти управляющего автомата хранится алгоритм, т.е. жестко заданная последовательность Д-операторов и операций.

Далее рассмотрим процесс функционирования модели, из описания которого будут понятны функции упомянутых Д-операторов и операций. Д-операторы, входящие в указанную последовательность, являются управляющими воздействиями, поступающими с выхода автомата У на вход автомата О. Операционный автомат, выполняя каждый поступивший Д-оператор, который в общем виде запишем как $A(\Delta)$, реализует очередной шаг функционирования модели. На каждом таком шаге модель переходит из некоторого исходного для этого шага состояния Δ в новое состояние Δ', т.е. $A(\Delta) = \Delta'$. Различия в этих состояниях модели, вызванные выполнением очередного Д-оператора, определяются изменениями значений, хранимых статической памятью, значениями, полученными динамической памятью, и/или значением логического условия. При этом на некоторых шагах вычислительного процесса динамические данные и/или логическое условие не определены, т.е. в некотором состоянии Δ'' $\{\alpha\} = \varnothing$ и/или $\Delta^d = \varnothing$.

Рассмотрим очередной шаг вычислительного процесса для трех возможных типов Д-операторов:

• $A(\Delta^s) = \Delta'^s$ — изменяет состояние статической памяти Δ^s и переводит ее в новое состояние Δ'^s, т.е. формирует новое состояние модели $\Delta' = \Delta'^s$, где $\{\alpha\} = \varnothing$ и $\Delta^d = \varnothing$;

• $A(\Delta^s) = \Delta^d$ — продуцирует и, таким образом, определяет значения, хранимые динамической памятью; в результате формируется новое состояние модели $\Delta' = \{\{\Delta^s\}, \{\Delta^d\}\}$, где состояние статической памяти остается неизменным и $\{\alpha\} = \varnothing$;

• $A(\Delta) = \alpha$ — продуцирует и, таким образом, определяет значение логического условия α, характеризующего состояние статической, если $\Delta = \Delta^s$, и динамической, если $\Delta = \Delta^d$, памяти; в результате формируется новое состояние модели $\Delta' = \{\{\Delta^s\}, \{\alpha\}\}$, где состояние статической памяти остается неизменным и $\Delta^d = \varnothing$.

Логические условия, полученные в результате выполнения Д-операторов третьего типа, передаются с выхода операционного автомата на вход управляющего, замыкая обратную связь между этими автоматами. Управляющий автомат выполняет некоторую операцию, в результате в соответствии со значением полученного логического условия выбирается один из нескольких возможных Д-операторов, который становится следующим в кортеже Д-операторовов, поступающих с выхода автомата У на вход автомата О.

Таким образом, каждым следующим шагом функционирования модели является поступление очередного Д-оператора из их последовательности, хранимой в памяти управляющего автомата. Такой пошаговый процесс выполнения алгоритма продолжается до его завершения, а в некоторых случаях количество таких шагов не ограничено.

Последовательность Д-операторов имеет следующие особенности:

— если за Д-оператором первого типа следует Д-оператор вида $A(\Delta^d) = \alpha$, то значение логического условия α будет неопределенным, т.е. $\{\alpha\} = \varnothing$, так как в этом случае неопределенными являются значения, хранимые в динамической памяти ($\Delta^d = \varnothing$);

— если за Д-оператором второго типа следует любой Д-оператор, отличный от $A(\Delta^d) = \alpha$, то значения, хранимые в динамической памяти (Δ^d), будут утрачены безвозвратно и безрезультатно;

— если за Д-операторами третьего типа не следует операция, то значения логического условия будут утрачены безвозвратно и безрезультатно;

— если выполнению любой операции предшествует Д-оператор первого или второго типа, то результат выполнения этой операции, т.е. выбор следующего Д-оператора, будет неопределенным, так как не определено значение логического условия $\{\alpha\} = \varnothing$;

— если в результате выполнения Д-оператора состояние модели не изменяется: $A(\Delta) = \Delta$, то такой Д-оператор назовем тождественным.

На основе предложенной модели будем строить алгебраический аппарат.

Построение формального аппарата начнем с формализации данных.

Элемент памяти Δ_i назовем носителем данных, кортеж значений $З_i$ — значениями данных. Элементом данных (множеством данных, данными) будем называть множество $D_i = \langle\{\Delta_i\}, \{З_i\}\rangle$, представляющее собой упорядоченную пару, где Δ_i — носитель данных, а $З_i$ — значения, принятые (хранимые) этим носителем данных. В частном случае, когда $D_i = \langle\{\{\delta_p\}, \{з_m\}\rangle$, где δ_p — ячейка памяти, а $з_m$ — значение, принятое (хранимое) этой ячейкой, элемент данных назовем элементарным.

Поскольку носители данных бывают статическими и динамическими, то статическими и динамическими будем называть также данные с приписыванием верхнего индекса s и d соответственно. Совокупность всех элементов статических данных $D^s = \{D_1^s, D_2^s, \ldots, D_n^s\}$, в том числе элементарных, носителем которых является множество Δ^s, образованное всеми элементами и ячейками статической памяти, назовем соответственно статическими данными и статической памятью. Элемент динамических данных D^d, носителем которых является элемент динамической памяти Δ^d, назовем соответственно динамическими данными и динамической памятью.

Для сокращения и упрощения дальнейших рассуждений, в которых необходимо одновременно рассматривать множества данных и множества носителей данных, а также множества значений, принятых этими носителями данных, введем следующие обозначения теоретико-множественных операций и их интерпретации для операций, которые будут использоваться ниже.

Пусть D'' и D''' — множества данных, Δ'' и Δ''' — множества носителей этих данных, $З''$ и $З'''$ — множества текущих значений, принятых этими носителями. Обозначения $D'' \widetilde{\cap} D'''$ будем понимать как $\Delta'' \cap \Delta'''$; $D'' \cong D'''$ — как $\Delta'' = \Delta'''$; $D'' \widetilde{/} D'''$ — как Δ'' / Δ'''; $D'' \not\approx D'''$ — как $З'' \neq З'''$ и $\Delta'' = \Delta'''$.

Заметим, что традиционные обозначения используются традиционно. Например, равенство данных $D'' = D'''$ означает равенство множеств носителей данных $\Delta'' = \Delta'''$ и равенство соответствующих множеств значений $З'' = З'''$.

Далее рассмотрим и определим Д-операторы, которые в общем случае будем записывать в виде $(D)\mathcal{R}(D')$. Множества данных D и D', специфицированные на входе и выходе Д-оператора, назовем соответственно входными и выходными данными. При этом полагаем, что именно данные, специфицированные на входе Д-оператора, подвергаются обработке, а данные, специфицированные на его выходе, являются результатом этой обработки.

Исходя из предложенной модели с использованием введенных обозначений определим Д-операторы.

Определение 1. Определим три типа Д-операторов:

• $(D)A(D')$ такой, что $D, D' \subseteq D^S$ и для множеств D и D' допустимы соотношения $D \widetilde{\cap} D' = \varnothing$, $D \widetilde{\cap} D' \neq \varnothing$, функционирует следующим образом:

— при $D \cong D'$ изменяет входные данные, т.е. в результате выполнения Д-оператора получаем $D \neq D'$, так как $З \neq З'$, и в этом случае множество данных D назовем изменяемым, а множество данных D' — измененным;

— при $D \widetilde{\cap} D' = \varnothing$ множество данных D не изменяется и его назовем неизменным, а о множестве данных D' будем говорить, что оно продуцируется Д-оператором, назовем его новым;

— при $D \widetilde{\cap} D' \neq \varnothing$ изменяется некоторое подмножество входных данных, т.е. данные $D' \widetilde{/} D$ — новые, $D \widetilde{\cap} D'$ — измененные, а $D \widetilde{/} D'$ — неизмененные;

• $(D)\Lambda(D')$ такой, что $D \subseteq D^S$, $D' \subseteq D^d$, определяет значения (продуцирует) множество динамических данных D', а множество данных D оставляет без изменений;

• $(D)P(\alpha)$, который в дальнейшем будем называть предикатом, полагая, что он проверяет выполнение некоторого отношения на множестве статических (если $D \subseteq D^S$) или динамических (если $D \subseteq D^d$) данных и продуцирует логическое условие α, характеризующее эти данные с точки зрения выполнения упомянутого отношения.

Рассмотрим определенные предикаты и логические условия более подробно.

Предикатами $(D)P(\alpha)$ назовем отображение множества D в двух- и трехэлементные множества $\{1, 0\}$ и $\{1, 0, \mu\}$, а логические условия, продуцируемые предикатами $\alpha^2 \in \{1, 0\}$ и $\alpha^3 \in \{1, 0, \mu\}$, принимают, таким образом, значения 0 — ложь, 1 — истина, μ — промежуточное значение, т.е. будем рассматривать два вида предикатов: $(D)P(\alpha^2)$ и $(D)P(\alpha^3)$. В связи с этим ниже логические условия в общем случае обозначим α^x (в частных случаях α^2 и α^3) и соответственно предикаты для общего случая запишем в виде $(D)P(\alpha^x)$ (в частных случаях $(D)P(\alpha^2)$ и $(D)P(\alpha^3)$).

Предикат $(D)P(\alpha^x)$ в общем случае частично определен на множестве D, в частности, он будет не определен, когда анализируемые с помощью предиката данные некорректны. В том случае, когда предикат не определен на множестве D, неопределенным будет и логическое условие α^x. Чтобы исключить такую ситуацию, введем и определим понятие алгоритмического предиката.

Определение 2. Алгоритмический предикат $(D)P(\alpha^x)$ проверяет выполнение некоторого отношения на множестве D. Если отношение на этом множестве выполняется, то продуцируемое этим алгоритмическим предикатом условие $\alpha^x = 1$. В противном случае предикат $(D)P(\alpha^2)$, в частности $(\varnothing)P(\alpha^2)$, продуцирует логическое условие $\alpha^2 = 0$, а предикат $(D)P(\alpha^3)$ проверяет выполнимость упомянутого отношения. В случае, когда это отношение не выполняется, он продуцирует логическое условие $\alpha^3 = 0$. В противном случае, т.е. когда предикат не определен на этом множестве, в частности $(\varnothing)P(\alpha^3)$, он продуцирует логическое условие $\alpha^3 = \mu$.

Из определения следует, что получены логические условия, определенные во всех возможных случаях.

Далее будем использовать традиционный термин «предикат», трактуя его в соответствии с приведенным определением. Рассмотрим операции, определенные на множестве логических условий. Будем полагать, что $L = \{\{L_2\}, (L_3\}\}$ — множество всех возможных предикатов, где $L_2 = \{(D_1)P_1(\alpha_1^2), (D_2)P_2(\alpha_2^2), \ldots$ $\ldots, (D_n)P_n(\alpha_n^2)\}$, $L_3 = \{(D_1)P_1(\alpha_1^3), (D_2)P_2(\alpha_2^3), \ldots, (D_k)P_k(\alpha_k^3)\}$; $l = \{\{l_2\}\{l_3\}\}$ — множество всех возможных логических условий, продуцируемых предикатами, где $l_2 = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2\}$ и $l_3 = \{\sigma_1^3, \sigma_2^3, \ldots, \sigma_k^3\}$ — множества двузначных и трехзначных логических условий соответственно.

На множестве двухзначных и трехзначных логических условий определены известные [4] операции соответственно двухзначной и трехзначной логики: дизъюнкция $\vee$, конъюнкция $\wedge$, отрицание $\neg$. Эти операции образуют множество операций $\Omega_1 \subseteq \Omega$, где Ω — сигнатура создаваемого алгебраического аппарата.

Определив операции на множестве логических условий, перейдем к построению операций, определенных на множестве Д-операторов. Будем полагать, что Д-операторы образуют множество U.

Композиция Д-операторов $(D_1)\mathcal{R}_1(D'_1)*(D_2)\mathcal{R}_2(D'_2)$ (операция $*$) означает последовательное выполнение сначала Д-оператора $(D_1)\mathcal{R}_1(D'_1)$, затем Д-оператора $(D_2)\mathcal{R}_2(D'_2)$, т.е. Д-оператор $(D_1)\mathcal{R}_1(D'_1)$ непосредственно предшествует Д-оператору $(D_2)\mathcal{R}_2(D'_2)$, а Д-оператор $(D_2)\mathcal{R}_2(D'_2)$ непосредственно следует за Д-оператором $(D_1)\mathcal{R}_1(D'_1)$.

Остальные операции, определенные на множестве U, рассматриваемые далее, основаны на анализе логических условий, что вытекает из описания модели ЭВМ. Из этого описания также следует, что логические условия доступны для анализа только операциям алгебраического аппарата. В качестве источника таких логических условий в соответствии с определением 1 могут выступать только предикаты.

Далее в определениях операций будем использовать предикат $(D)P(\alpha^x)$ такой, что $D \subseteq D^S$.

Операция p_3-дизъюнкции:

$$[(D)P(\alpha^3)]((D_A)A(D'_A)\vee(D_A)B(D'_B)\vee(D_C)C(D'_C))=$$

$$=\begin{cases}(D_A)A(D'_A), & \text{если } \alpha^3=1;\\ (D_B)B(D'_B), & \text{если } \alpha^3=0;\\ (D_C)C(D'_C), & \text{если } \alpha^3=\mu.\end{cases} \tag{1}$$

Результатом выполнения этой операции является один из трех возможных Д-операторов, который выбирается в соответствии со значением логического условия α^3.

Операция p_2-дизъюнкции:

$$[(D)P(\alpha^2)]((D_A)A(D'_A)\vee(D_A)B(D'_B))=\begin{cases}(D_A)A(D'_A), & \text{если } \alpha^2=1,\\ (D_B)B(D'_B), & \text{если } \alpha^2=0.\end{cases} \tag{2}$$

Результатом выполнения этой операции является один из двух возможных Д-операторов, который выбирается в соответствии со значением логического условия α^2.

Введем к рассмотрению следующую операцию — p-итерацию.

Операция p-итерации

$$[(D)P(\alpha^2)]\{(D_A)A(D'_A)\} \tag{3}$$

состоит в вычислении предиката и проверке полученного логического условия α^2. Если α^2 истинно, выполняется Д-оператор $(D_A)A(D'_A)$ и вновь вычисляется предикат и проверяется условие α^2. Циклический процесс, состоящий в проверке условия α^2 и выполнении Д-оператора, происходит до тех пор, пока имеет место условие $\alpha^2=1$, в противном случае операция p-итерации завершается. Необходимым для завершения операции условием является выполнение соотношения $D\cap D'_A\neq\varnothing$.

Введенные операции, определенные на U, образуют множество $\Omega_2\subseteq\Omega$, где Ω — сигнатура создаваемого алгебраического аппарата.

Теперь обратимся к операциям, определенным на множестве логических условий, возможности которых до сих пор не использовались.

Отметим, что предикаты могут быть связаны операцией композиции $(D_1)P_1(\alpha_1^3)*(D_2)P_2(\alpha_2^3)$ в соответствии с определением этой операции, а логические условия, продуцируемые этими предикатами, могут быть связаны логическими операциями из множества Ω_1.

Необходимо, чтобы при выполнении композиции предикатов $(D)P(\alpha^x)$, вне зависимости от их количества, формировалось логическое условие как результат применения соответствующих булевых операций ко всем условиям, продуцируемым этими предикатами.

В соответствии с этим требованием операцию p_3-дизъюнкции запишем в виде

$$[(D_1)P_1(\alpha_1^3)\circ(D_2)P_2(\alpha_2^3)\circ\ldots\circ(D_k)P_k(\alpha_k^3)]((D_A)A(D'_A)\vee$$

$$\vee(D_A)B(D'_B)\vee(D_C)C(D'_C))=\begin{cases}(D_A)A(D'_A), & \text{если } \alpha_\Sigma^3=1;\\ (D_B)B(D'_B), & \text{если } \alpha_\Sigma^3=0;\\ (D_C)C(D'_C), & \text{если } \alpha_\Sigma^3=\mu,\end{cases} \tag{4}$$

где $\alpha_\Sigma^3=\alpha_1^3\circ\alpha_2^3\circ\ldots\circ\alpha_k^3$, операция $\circ$ определяет одну из булевых операций. При этом будем полагать, что предикаты выполняются в соответствии со свойствами композиции, а логическое условие α_Σ^3 является результатом при-

менения булевых операций ко всем логическим условиям $\alpha_1^3 \circ \alpha_2^3 \circ \ldots \circ \alpha_k^3$, связанным этими операциями. Для операций (2) и (3) запись аналогична.

Построенная система алгоритмических алгебр (САА/Д) представляет собой двухосновную алгебраическую систему $< U, l, \Omega >$, основами которой являются множество Д-операторов U и множество логических условий l, а $\Omega = \Omega_1 \cup \Omega_2$ — ее сигнатура, состоящая из Ω_1 — логических операций, принимающих значения на множестве l, и множества Ω_2 — операций, принимающих значения на множестве Д-операторов U.

Сигнатура построенного алгебраического аппарата соответствует сигнатуре алгебры Дейкстры и позволяет описывать алгоритмы произвольной сложности. При этом на входе и выходе предложенных Д-операторов специфицируются подлежащие обработке данные, что позволяет одновременно описывать управляющие структуры и обрабатываемые данные.

Кроме того, в предлагаемом формальном аппарате по построению логические условия (в частности, трехзначные) всегда принимают определенные значения, в том числе и в случае неопределенных и некорректных данных. В результате последовательность Д-операторов, задаваемая операциями из Ω_1, всегда определена, а операция p_3-дизъюнкции обеспечивает возможность адекватной реакции на случай появления некорректных данных, т.е. предусмотрена возможность реализации элементов защитного программирования. Таким образом, удовлетворены два требования, предъявленные во введении к создаваемому формальному аппарату.

Однако в построенной САА/Д отсутствует принципиально важная операция, позволяющая осуществить прогнозирование вычислительного процесса. Эта операция введена В.М. Глушковым и названа им левым умножением условия на оператор. Именно эта операция отличает и обеспечивает превосходство по порождающей мощности алгебры Глушкова над алгеброй Дейкстры [5].

В следующем разделе устраним этот пробел с помощью тех средств и возможностей, которые ранее не использовались.

ПРОГНОЗИРОВАНИЕ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА.

Для решения задачи прогнозирования вычислительного процесса введем несколько новых понятий.

Будем исходить из того, что вычислительный процесс носит дискретный характер, а каждым шагом этого процесса является выполнение очередного Д-оператора. Между завершением выполнения Д-оператора $(D_1)\mathcal{R}_1(D_1')$ и началом выполнения Д-оператора $(D_2)\mathcal{R}_2(D_2')$ никакие действия не выполняются. Причем после (до) выполнения каждого i-го ($i+1$-го) шага вычислительный процесс находится в некотором состоянии, которое по аналогии с термином «состояние модели», назовем текущим информационным состоянием вычислительного процесса (в дальнейшем — состояние вычислительного процесса). Определим это понятие и его свойства.

Определение 3. Текущим состоянием вычислительного процесса на i-м шаге его выполнения назовем множество данных D_i^T, которое может находиться в одном из трех состояний: $D_i^T = D_i^s$, $D_i^T = \{\{D_i^s\}, \{\alpha_i^x\}\}$, $D_i^T = \{\{D_i^s\}, \{D_i^d\}\}$, где D_i^s — текущее состояние множества статических данных, D_i^d — текущее состояние множества динамических данных, α_i^x — логическое условие, характеризующее состояние множества D_i^s или D_i^d. Подмножества, образующие множество D_i^T, обладают следующими свойствами:

— для $\forall i$ выполняется $D_i^s \subseteq D_i^T$ и допустимо как соотношение $D_i^T \supseteq D_i^s \not\approx D_{i+1}^s \subseteq D_{i+1}^T$, так и соотношение $D_i^T \supseteq D_i^s = D_{i+1}^s \subseteq D_{i+1}^T$. Последнее соотношение может быть переписано как в виде $D_i^T \supseteq D_i^s \subseteq D_{i+1}^T$, так и в виде $D_i^T \supseteq D_{i+1}^s \subseteq D_{i+1}^T$, т.е. статическая составляющая состояния вычислительного процесса может изменяться, но имеет место на каждом его шаге, а именно $D_i^T \supseteq D_i^s \cong D_{i+1}^s \subseteq D_{i+1}^T$;

— динамическая составляющая состояния D_i^T, множество динамических данных D_i^d и логическое условие α_i^x определены только на некоторых шагах вычислительного процесса, и в этом и только этом случае $D_i^d \subset D_i^T$ и $\{\alpha_i^x\} \subset D_i^T$. При $D_i^d \subset D_i^T$ и $\{\alpha_i^x\} \subset D_i^T$ выполняется $D_{i+1}^T \supset D_i^d = \varnothing$ и $D_{i+1}^T \supset \{\alpha_i^x\} = \varnothing$, т.е. динамическая составляющая текущего состояния вычислительного процесса полностью теряется на каждом следующем его шаге.

Из определения вытекает, что для $\forall\, i$ выполняется $D_i^T \supset D_i^d \neq D_{i+1}^d \subset D_{i+1}^T$ и $D_i^T \supset \{\alpha_i^x\} \neq \{\alpha_{i+1}^x\} \subset D_{i+1}^T$, так как D_i^d и D_{i+1}^d — разные множества, а α_i^x и α_{i+1}^x — разные логические условия. В связи с этим нижний индекс при записи динамических данных и логических условий будем в очевидных случаях опускать, т.е. записывать их в виде D^d и α^x.

Докажем наличие существенного для дальнейшего изложения свойства операции композиции, для чего предварительно определим понятие тождественного Д-оператора.

Определение 4. Д-оператор $(D)\mathcal{R}(D')$ назовем тождественным и обозначим Z, если в результате его выполнения состояние алгоритма не изменяется, т.е. для состояния вычислительного процесса D_i^T, имевшего место до выполнения Д-оператора, и состояния D_{i+1}^T, полученного после его выполнения, имеет место соотношение $D_i^T = D_{i+1}^T$. Если $D_i^T \neq D_{i+1}^T$, но $D_i^s = D_{i+1}^s$, а динамическая составляющая $D^d \subset D_{i+1}^T$ или $\alpha^x \subset D_{i+1}^T$ не обрабатывается и, таким образом, не оказывает влияния на последующие шаги вычислительного процесса, то полагаем, что $D_{i+1}^T = D_{i+1}^s$, и такой Д-оператор также будем считать тождественным.

Теорема 1. Операция композиции Д-операторов обладает свойством $(D)\mathcal{R}(D') * Z = Z * (D)\mathcal{R}(D') = (D)\mathcal{R}(D')$.

Доказательство. Пусть в результате выполнения Д-оператора $(D)\mathcal{R}(D')$ в соответствии с определением 3 вычислительный процесс из исходного состояния D_i^T переходит в состояния D_{i+1}^T. Предположим, что Д-оператор $(D)\mathcal{R}(D')$ предшествует Д-оператору $(D_2)\mathcal{R}_2(D_2')$, тогда в результате выполнения композиции Д-операторов $(D)\mathcal{R}(D') * (D_2)\mathcal{R}_2(D_2')$ вычислительный процесс из состояния D_i^T перейдет в состояние D_{i+2}^T. Если Д-оператор $(D_2)\mathcal{R}_2(D_2')$ тождественный, то в соответствии с определением 4 $D_{i+2}^T = D_{i+1}^T$, т.е. композиция $(D)\mathcal{R}(D') * Z$ переводит вычислительный процесс из состояния D_i^T в состояние D_{i+1}^T, откуда следует $(D)\mathcal{R}(D') * Z = (D)\mathcal{R}(D')$. Предположим, что Д-оператору $(D)\mathcal{R}(D')$ предшествует Д-оператор $(D_1)\mathcal{R}_1(D_1')$. Тогда в результате выполнения композиции Д-операторов $(D_1)\mathcal{R}_1(D'_1) * (D)\mathcal{R}(D')$ вычислительный процесс из состояния D_{i-1}^T перейдет в состояние D_{i+1}^T. Если Д-оператор $(D_1)\mathcal{R}_1(D_1')$ тождественный, то в соответствии с определением 4 $D_{i-1}^T = D_i^T$, т.е. композиция $Z * (D)\mathcal{R}(D')$ переводит вычислительный процесс из состояния D_i^T в состояние D_{i+1}^T, откуда следует $Z * (D)\mathcal{R}(D') = (D)\mathcal{R}(D')$.

Теорема доказана.

Теперь обратимся к прогнозированию вычислительного процесса, т.е. предсказанию результата выполнения некоторого Д-оператора $(D_A)A(D_A')$, которое определим следующим образом.

Определение 5. Прогнозом выполнения Д-оператора $(D_A)A(D_A')$, который переводит вычислительный процесс из состояния $D_i^T = D_i^s$ в состояние $D_{i+1}^T = D_{i+1}^s$, назовем состояние $D_j^T = \{\{D_i^s\}, \{\alpha_j^x\}\}$ такое, что логическое условие

α_j^x, полученное в этом состоянии, характеризует состояние множества данных D_{i+1}^s при неизменном (сохраненном) D_i^s.

Покажем, что композиция Д-операторов вида $(D)\Lambda(D')^*(\hat{D})P(\alpha^x)$, где $D \subseteq D^s$ и D', $\hat{D} \subseteq D^d$, позволяет такой прогноз осуществить. Предварительно докажем наличие некоторого свойства этой композиции Д-операторов.

Теорема 2. Для данных в композиции Д-операторов $(D)\Lambda(D')^*(\hat{D})P(\alpha^x)$ выполняется соотношение $D' = D^d = \hat{D}$.

Доказательство. Предположим, что выполнение указанной композиции Д-операторов начинается в некотором исходном состоянии D_i^T таком, что в соответствии с определением 3 $D_i^s \subseteq D_i^T$. В соответствии с определениями 1 и 3 после выполнения Д-оператора $(D)\Lambda(D')$, где $D' \subseteq D^d$, вне зависимости от исходного состояния D_i^T вычислительный процесс перейдет в состояние $D_{i+1}^T = \{\{D_i^s\}, \{D_{i+1}^d\}\}$. Но из определения 3 следует, что исходные динамические данные $D_i^d \subset D_{i+1}^T$, если они имели место, в результате выполнения Д-оператора $(D)\Lambda(D')$ были утрачены полностью, т.е. $D_{i+1}^T \supset D_i^d = \varnothing$. Таким образом, все динамические данные $D_{i+1}^d \subset D_{i+1}^T$ являются результатом выполнения только Д-оператора $(D)\Lambda(D')$, откуда вытекает $D' = D_{i+1}^d$.

Следующим шагом вычислительного процесса, в результате выполнения которого он переходит из состояния D_{i+1}^T в состояние D_{i+2}^T, является выполнение предиката $(\hat{D})P(\alpha^x)$ такого, что $\hat{D} \subseteq D_{i+1}^d$. Поскольку в соответствии с определением 3 после выполнения этого шага динамические данные будут утрачены полностью, т.е. $D_{i+2}^T \supset D_{i+1}^d = \varnothing$, из предположения о том, что $\hat{D} \subseteq D_{i+1}^d$, следует, что некоторое подмножество динамических данных $D_{i+1}^d / \hat{D}$ будет безвозвратно и безрезультатно утрачено. Исключение такой потери данных возможно только при наличии ограничения вида $\hat{D} = D^d$. Таким образом, $D' = D^d = \hat{D}$ и теорема доказана.

Докажем теперь возможность прогнозирования вычислительного процесса, предварительно определив понятие эквивалентности Д-операторов.

Определение 6. Д-оператор $(D_A)A(D'_A)$ эквивалентен Д-оператору $(D_B)B(D'_B)$ в том и только в том случае, когда при $\forall D_A = D_B$ выполняется $D'_B = D'_A$. Д-оператор $(D)\Lambda(D^d)$ эквивалентен Д-оператору $(D_A)A(D'_A)$ в том и только в том случае, когда при $\forall D = D_A$ выполняется $D^d = D''_A \subseteq D'_A$.

Теорема 3. Композиция Д-операторов $(D)\Lambda(D^d)^*(D^d)P(\alpha^x)$ позволяет выполнять прогнозирование результата выполнения Д-оператора $(D_A)A(D'_A)$ при условии, что Д-оператор $(D)\Lambda(D^d)$ эквивалентен этому Д-оператору.

Доказательство. Предположим, что нас интересует с точки зрения прогноза результатов выполнения Д-оператора $(D_A)A(D'_A)$ некоторое подмножество $D''_A \subseteq D'_A$. В результате выполнения этого Д-оператора вычислительный процесс перейдет из некоторого исходного состояния D_i^T в состояние D_{i+1}^T. Полученное состояние таково, что если Д-оператор $(D_A)A(D'_A)$ не тождествен в соответствии с определением 4, то согласно определениям 1 и 3 $D''_A \subseteq D'_A \subseteq D_{i+1}^s$ и $D_i^T \supseteq$ $\supseteq D_i^s \not\approx D_{i+1}^s \subseteq D_{i+1}^T$, т.е. в результате выполнения Д-оператора статические данные некоторым образом изменились.

В случае, когда выполняется Д-оператор $(D_A)\Lambda(D^d)$, эквивалентный Д-оператору $(D_A)A(D'_A)$ (см. определение 6), вычислительный процесс из того же исходного состояния D_i^T в соответствии с определениями 1 и 3 перейдет в состоя-

ние $D^T_{i+1} = \{\{D^s_i\}, \{D^d\}\}$, где согласно теореме 2 и определению 6 $D^d = D''_A \subseteq$ $\subseteq D'_A \subseteq D^s_{i+1}$. В связи с этим предикат может быть записан в виде $(D''_A)P(\alpha^x)$ и после его выполнения вычислительный процесс перейдет в состояние $D^T_{i+2} = \{\{D^s_i\}, \{\alpha^x\}\}$, где логическое условие α^x является некоторой характеристикой (оценкой) множества данных $D''_A \subseteq D'_A$, представляющих собой результат (некоторую часть результата) выполнения Д-оператора $(D_A)A(D'_A)$.

Таким образом, получили характеристику результата выполнения Д-оператора $(D_A)A(D'_A)$ при неизменной статической составляющей D^s_i состояния вычислительного процесса D^T_{i+2}, что в соответствии с определением 5 является прогнозом результатов выполнения этого Д-оператора.

Теорема доказана.

Прежде чем использовать возможности прогнозирования вычислительного процесса, отметим, что результатом выполнения композиции $(D)\Lambda(D^d)*$ $*(D^d)P(\alpha^x)$ является логическое условие, что позволяет ее использовать в операциях (1)–(3).

В дальнейшем изложении с учетом доказанных теорем 2 и 3 рассмотренную композицию Д-оператора и предиката будем записывать в виде $(D)\Lambda(D^d)P(\alpha^x)$.

Продемонстрируем возможности прогнозирования вычислительного процесса, полагая осуществить прогноз выполнения Д-оператора $(D_A)A(D'_A)$, которому эквивалентен в соответствии с определением 6 Д-оператор $(D_A)\Lambda(D^d)$.

Операцию p_3-дизъюнкции в данном случае запишем в виде

$$[(D_A)\Lambda(D^d)P(\alpha^3)]((D_A)A(D'_A)\vee(D_A)B(D'_B)\vee(D_C)C(D'_C)) =$$

$$= \begin{cases} (D_A)A(D'_A), & \text{если } \alpha^3 = 1; \\ (D_B)B(D'_B), & \text{если } \alpha^3 = 0; \\ (D_C)C(D'_C), & \text{если } \alpha^3 = \mu. \end{cases}$$

В рассмотренном случае если прогноз выполнения Д-оператора $(D_A)A(D'_A)$ удовлетворительный, то выполняется этот Д-оператор, в противном случае — Д-оператор $(D_A)B(D'_B)$. Если прогнозирование осуществить не удалось, то выполняется Д-оператор $(D_C)C(D'_C)$.

Для операции p_2-дизъюнкции запись аналогична.

Прогнозирование может быть использовано и для операции p-итерации. В этом случае операция записывается в виде

$$[(D_A)\Lambda(D^d)P(\alpha^2)]\{(D_B)B(D'_B)\},$$

где $(D_A)\Lambda(D^d)$ эквивалентен Д-оператору $(D_A)A(D'_A)$, для которого осуществляется прогноз, а необходимым для завершения операции условием является выполнение соотношения $D_A \cap D'_B \neq \varnothing$. При этом прогнозирование выполняется в цикле при различных входных данных. В частном случае эта операция может быть записана в виде $[(D_A)\Lambda(D^d)P(\alpha^2)]\{(D_A)A(D'_A)\}$, и тогда перед каждым выполнением Д-оператора $(D_A)A(D'_A)$ осуществляем прогнозирование его выполнения. В случае, когда прогноз неудовлетворительный, выполнение этого Д-оператора прекращается.

Возможности рассматриваемого формального аппарата относительно реализации прогнозирования вычислительного процесса на этом не исчерпываются. Они могут быть использованы и в случае прогнозирования выполнения нескольких Д-операторов. Для этого p_3-дизъюнкцию, по аналогии с (4), запишем в виде

$$[(D_1)\Lambda_1(D^d_1)P_1(\alpha^3_1)\circ\ldots\circ(D_k)\Lambda_k(D^d_k)P_k(\alpha^3_k)]((D_A)A(D'_A)\vee$$

$$\vee(D_A)B(D'_B)\vee(D_C)C(D'_C))=\begin{cases}(D_A)A(D'_A), & \text{если } \alpha_\Sigma^3=1;\\ (D_B)B(D'_B), & \text{если } \alpha_\Sigma^3=0;\\ (D_C)C(D'_C), & \text{если } \alpha_\Sigma^3=\mu,\end{cases}$$

где $\alpha_\Sigma^3=\alpha_1^3\circ\alpha_2^3\circ\ldots\circ\alpha_k^3$, операция $\circ$ — одна из булевых операций. При этом будем полагать, что прогнозирование выполняется для множества Д-операторов $(D_{G_1})G_1(D'_{G_1}),\ldots,(D_{G_k})G_k(D'_{G_k})$, которым эквивалентны Д-операторы $(D_1)\Lambda_1(D_1^d),\ldots,(D_k)\Lambda_k(D_k^d)$. Прогнозирование выполняется в соответствии со свойствами композиции, а логическое условие α_Σ^3 является результатом применения булевых операций ко всем логическим условиям $\alpha_1^3\circ\alpha_2^3\circ\ldots\circ\alpha_k^3$, связанным этими операциями.

В результате выполненных построений, дополнив сигнатуру САА/Д операцией прогнозирования, мы получили алгебраический аппарат с таким же составом операций, как в алгебре Глушкова, и удовлетворили еще одно из требований, сформулированных во введении.

Свойства введенных Д-операторов и операции композиции позволяют расширить возможности САА/Д за счет построения производных Д-операторов и алгоритмических конструкций.

ПРОИЗВОДНЫЕ Д-ОПЕРАТОРЫ И ОПЕРАЦИИ

Построение производных алгоритмических конструкций начнем с определения границы возможностей таких построений. Для этого, исходя из определения композиции Д-операторов, где на последовательность Д-операторов не накладывается никаких ограничений, приведем список всех возможных их сочетаний:

$(D_A)A(D'_A)*(D_B)B(D'_B)$; (5)

$(D_A)A(D'_A)*(D)P(\alpha^x)$; (6)

$(D_A)A(D'_A)*(D)\Lambda(D^d)$; (7)

$(D_A)A(D'_A)*(D^d)P(\alpha^x)$; (8)

$(D_1)\Lambda_1(D_1^d)*(D_2)\Lambda_2(D_2^d)$; (9)

$(D)\Lambda(D^d)*(D)P(\alpha^x)$; (10)

$(D)\Lambda(D^d)*(D_A)A(D'_A)$; (11)

$(D)\Lambda(D^d)P(\alpha^x)$; (12)

$(D_1)P_1(\alpha_1^x)*(D_2)P_2(\alpha_2^x)$; (13)

$(D)P_1(\alpha^x)*(D^d)P_2(\beta^x)$; (14)

$(D)P(\alpha^x)*(D_A)A(D'_A)$; (15)

$(D)P(\alpha^x)*(\hat{D})\Lambda(D^d)$; (16)

$(D_1^d)P_1(\alpha_1^x)*(D_2^d)P_2(\alpha_2^x)$; (17)

$(D^d)P_1(\alpha^x)*(D)P_2(\beta^x)$; (18)

$(D^d)P(\alpha^x)*(D_A)A(D'_A)$; (19)

$(D_1^d)P(\alpha^x)*(D)\Lambda(D_2^d)$. (20)

Докажем, что не все соотношения из приведенного достаточно обширного списка могут быть использованы.

Теорема 4. Использование Д-операторов $(D)\Lambda(D^d)$ и $(D^d)P(\alpha^x)$ возможно в том и только в том случае, когда Д-оператор $(D)\Lambda(D^d)$ предшествует предикату $(D^d)P(\alpha^x)$, а предикат $(D^d)P(\alpha^x)$ следует за $(D)\Lambda(D^d)$, и такая композиция может быть использована только в операциях (1)–(3).

Доказательство построим от противного.

Предположим, что возможна композиция Д-операторов вида $(D)\Lambda(D^d)*$ $*(D)\mathcal{R}(D')$, где Д-оператор $(D)\mathcal{R}(D')$ отличен от $(D^d)P(\alpha^x)$.

В этом случае, поскольку динамические данные в соответствии с определением 1 анализируются только предикатом $(D^d)P(\alpha^x)$, а $(D)\mathcal{R}(D')$ отличен от $(D^d)P(\alpha^x)$, множество данных D^d будет согласно определению 3 утрачено без-

результатно, т.е. на следующие состояния вычислительного процесса не повлияет. Отсюда в соответствии с определением 4 получим $(D)\Lambda(D^d) = Z$, а с учетом теоремы 1 $(D)\Lambda(D^d) * (D)\mathcal{R}(D') = Z * (D)\mathcal{R}(D') = (D)\mathcal{R}(D')$, т.е. такой вариант использования этих Д-операторов невозможен.

Теперь предположим, что возможна композиция Д-операторов вида $(D)\mathcal{R}(D') * (D^d)P(\alpha^x)$, где $(D)\mathcal{R}(D')$ — произвольный Д-оператор, отличный от $(D)\Lambda(D^d)$. Поскольку единственным Д-оператором, продуцирующим динамические данные, в соответствии с определением 1 является Д-оператор $(D)\Lambda(D^d)$, то в результате выполнения Д-оператора $(D)\mathcal{R}(D')$ получим в соответствии с определением 3 такое состояние вычислительного процесса, в котором $D^d = \varnothing$. В этом случае согласно теореме 2 получим предикаты $(\varnothing)P(\alpha^3)$ $((\varnothing)P(\alpha^2))$, в результате вычисления которых в соответствии с определением 2 $\alpha^3 = \mu$ $(\alpha^2 = 0)$. Из описания модели ЭВМ следует, что логические условия могут использоваться только операциями (1)–(3), однако любая операция, в которой будет использована такая композиция, выродится в случае p_3-дизъюнкции

$$[(D)\mathcal{R}(D') * (D^d)P(\alpha^3)][(D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C)) =$$
$$= [(D)\mathcal{R}(D') * (\varnothing)P(\mu)][(D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C)) =$$
$$= (D)\mathcal{R}(D') * (D_C)C(D'_C);$$

в случае p_2-дизъюнкции

$$[(D)\mathcal{R}(D') * (D^d)P(\alpha^2)][(D_A)A(D'_A) \vee (D_B)B(D'_B)) =$$
$$= [(D)\mathcal{R}(D') * (\varnothing)P(0)][(D_A)A(D'_A) \vee (D_B)B(D'_B)) =$$
$$= (D)\mathcal{R}(D') * (D_B)B(D'_B);$$

в случае p-итерации в соответствии со свойством этой операции $[(D)P(0)]\{(D_A)A(D'_A)\} = Z$, где 0 — тождественно ложное условие (см. определение операции), и с учетом теоремы 1

$$[(D)\mathcal{R}(D') * (D^d)P(\alpha^2)]\{(D_A)A(D'_A)\} = [(D)\mathcal{R}(D') * (\varnothing)P(0)]\{(D_A)A(D'_A)\} =$$
$$= (D)\mathcal{R}(D') * Z = (D)\mathcal{R}(D').$$

Таким образом, и второй рассмотренный случай композиции Д-операторов невозможен, что противоречит сделанным предположениям. Отсюда следует, что возможна только композиция $(D)\Lambda(D^d)P(\alpha^x)$, результатом выполнения которой является логическое условие. Но, как отмечалось выше, логические условия используются только в операциях (1)–(3), в противном случае в соответствии с определением 3 эти логические условия теряются. Отсюда вытекает, что единственная возможная композиция Д-операторов $(D)\Lambda(D^d)$ и $(D^d)P(\alpha^x)$ может быть использована только в операциях (1)–(3).

Теорема доказана.

Следствием доказанной теоремы является то, что соотношения (8)–(11), (14), (17) не могут использоваться в операциях (1)–(3), поэтому из дальнейшего рассмотрения исключаются.

Теорема 5. Для композиций Д-операторов вида $(D^d)P(\alpha^x) * (D)\mathcal{R}(D')$ и $(D)P(\beta^x) * (D)\mathcal{R}(D')$ (за исключением случая $(D_1)P_1(\alpha_1^x) * (D_2)P_2(\alpha_2^x)$) выполняются соотношения $(D^d)P(\alpha^x) * (D)\mathcal{R}(D') = (D)\mathcal{R}(D')$ и $(D)P(\beta^x) * (D)\mathcal{R}(D') = (D)\mathcal{R}(D')$.

Доказательство. В соответствии с определениями 1 и 3 состояние вычислительного процесса после выполнения Д-операторов $(D^d)P(\alpha^x)$ и $(D)P(\beta^x)$, вне зависимости от его состояния D_i^T до выполнения этого Д-оператора, имеет вид

$D^T_{i+1} = \{\{D^s_i\}, \{\alpha^x\}\}$ и $D^T_{i+1} = \{\{D^s_i\}, \{\beta^x\}\}$. При этом ни один из возможных Д-операторов, следующих за предикатом, логического условия не использует и относительно выполнения этого Д-оператора состояние вычислительного процесса перед его выполнением $D^T_{i+1} = \{\{D^s_i\}\}$. В результате логические условия, полученные на $i+1$-м шаге, после выполнения Д-оператора $(D)\mathcal{R}(D')$ в соответствии с определением 3 будут безвозвратно и безрезультатно утрачены и не окажут влияния на вычислительный процесс на любом из последующих шагов. Отсюда согласно определению 4 выполняется $(D^d)P(\alpha^x) = Z$ и $(D)P(\beta^x) = Z$, а согласно теореме 1 $(D^d)P(\alpha^x)*(D)\mathcal{R}(D') = Z*(D)\mathcal{R}(D') = (D)\mathcal{R}(D')$ и $(D_1)P_1(\alpha^x_1)*$ $*(D)\mathcal{R}(D') = Z*(D)\mathcal{R}(D') = (D)\mathcal{R}(D')$.

Теорема доказана.

Следствием доказанной теоремы является исключение из дальнейшего рассмотрения соотношений (12), (13), (15)–(17) .

Таким образом, в качестве допустимых к использованию остались следующие соотношения:

$(D_A)A(D'_A)*(D_B)B(D'_B)$; (21) $(D)\Lambda(D^d)P(\alpha^x)$; (24)

$(D_A)A(D'_A)*(D)P(\alpha^x)$; (22) $(D_1)P_1(\alpha^x_1)*(D_2)P_2(\alpha^x_2)$. (25)

$(D_A)A(D'_A)*(D)\Lambda(D^d)$; (23)

Соотношения (24) и (25) использовались выше, а соотношения (21) и (23) не могут быть использованы в рассматриваемых операциях, так как в результате их выполнения не продуцируются логические условия, необходимые для выполнения этих операций.

Таким образом, относительно использования композиций Д-операторов в операциях (1)–(3) нерассмотренной осталась только конструкция (22). Для того чтобы воспользоваться возможностями такой композиции, определим понятие связанных Д-операторов и понятие контекста выполнения Д-оператора.

Определение 7. Д-операторы $(D_1)\mathcal{R}_1(D'_1)*(D_2)\mathcal{R}_2(D'_2)$ назовем связанными, если для них выполняется соотношение $D'_1 \cap D_2 \neq \varnothing$.

Определение 8. Информационным контекстом результата выполнения Д-оператора $(D_A)A(D'_A)$ в композиции $(D_A)A(D'_A)*(D)P(\alpha^x)$ назовем множество данных $D^k = D/D'_A$, не измененных Д-оператором $(D_A)A(D'_A)$ (определение 1), но анализируемых предикатом.

Из определения 8 следует, что при $D^k = \varnothing$ предикат анализирует только результат выполнения Д-оператора $(D_A)A(D'_A)$ без учета контекста. При этом в случае $D = D'_A$ он анализирует весь результат, а при $D \subset D'_A$ — его часть.

Из определений 7 и 8 вытекает, что если в композиции $(D_A)A(D'_A)*$ $*(D)P(\alpha^x)$ Д-операторы не связаны, то анализируется только контекст результата выполнения Д-оператора $(D_A)A(D'_A)$.

Рассматриваемую композицию Д-операторов в случае, когда предикат анализирует результат выполнения Д-оператора $(D_A)A(D'_A)$ без учета контекста, иногда удобно использовать в операциях (1)–(3). Тогда операция p_3-дизъюнкции будет записана в виде

$$[(D_A)A(D'_A)*(D)P(\alpha^3)]((D_G)G(D'_G)\vee(D_B)B(D'_B)\vee(D_C)C(D'_C)),$$

а в случае, когда $D = D'_A$, записав эту композицию в виде $(D_A)A(D'_A)P(\alpha^3)$, получим

$$[(D_A)A(D'_A)P(\alpha^3)]((D_G)G(D'_G)\vee(D_B)B(D'_B)\vee(D_C)C(D'_C)).$$

Для операции p_2-дизъюнкции запись аналогична, а операция p-итерации имеет вид

$$[(D_A)A(D'_A)*(D)P(\alpha^2)]\{(D_B)B(D'_B)\},$$

а в случае, когда $D = D'_A$, имеем $[(D_A)A(D'_A)P(\alpha^2)]\{(D_B)B(D'_B)\}$.

Следует отметить, что в данном случае условие цикла может быть независимым от тела цикла, тогда ограничение $D_A \cap D'_B \neq \varnothing$, наложенное на операцию при ее определении, может быть снято. В таком варианте получим операцию, эквивалентную программной конструкции типа for.

Теперь перейдем к построению производных Д-операторов и покажем, что из выражений (19)–(21) могут быть получены производные Д-операторы. Для проведения доказательства определим следующие новые понятия.

Определение 9. Полным результатом выполнения некоторого Д-оператора $(D)\mathcal{R}(D')$ назовем состояние вычислительного процесса D^T_{i+1}, полученное после его выполнения. Наблюдаемым результатом некоторого Д-оператора $(D)\mathcal{R}(D')$ назовем множество данных D'.

Теорема 6. Д-операторы, связанные операцией композиции $(D_1)\mathcal{R}_1(D'_1)*$ $*(D_2)\mathcal{R}_2(D'_2)$, могут быть объединены в один, эквивалентный этой композиции Д-оператор $(D_3)\mathcal{R}_3(D'_3)$, при условии, что специфицированные на его входе данные обладают свойством $D_3 = D_1 \cup D_2$.

Доказательство. При выполнении композиции Д-операторов $(D_1)\mathcal{R}_1(D'_1)*$ $*(D_2)\mathcal{R}_2(D'_2)$ вычислительный процесс переходит из исходного состояния D^T_i последовательно в состояние D^T_{i+1} после выполнения Д-оператора $(D_1)\mathcal{R}_1(D'_1)$ и в состояние D^T_{i+2} после выполнения Д-оператора $(D_2)\mathcal{R}_2(D'_2)$. Другими словами, композиция $(D_1)\mathcal{R}_1(D'_1)*(D_2)\mathcal{R}_2(D'_2)$ переводит вычислительный процесс из исходного состояния D^T_i в результирующее состояние D^T_{i+2}. На входах Д-операторов $(D_1)\mathcal{R}_1(D'_1)$ и $(D_2)\mathcal{R}_2(D'_2)$ специфицированы входные данные соответственно D_1 и D_2, которые подлежат обработке этими Д-операторами. При этом согласно определениям 1 и 3 для исходного состояния D^T_i выполняется $D_1 \subseteq$ $\subseteq D^s_i \subseteq D^T_i$ и $D_2 \subseteq D^s_i \subseteq D^T_i$. Объединив множества входных данных $D_1 \cup D_2 =$ $= D_3$ и исключив из рассмотрения состояния вычислительного процесса D^T_{i+1}, получим $(D_1)\mathcal{R}_1(D'_1)*(D_2)\mathcal{R}_2(D'_2) = (D_3)\mathcal{R}_1{*}\mathcal{R}_2(D'_3)$. Обозначив последовательность действий по обработке данных, выполняемую парой Д-операторов $\mathcal{R}_1 * \mathcal{R}_2$ как $\mathcal{R}_3$, получим Д-оператор $(D_3)\mathcal{R}_3(D'_3)$, функционально полностью эквивалентный (в смысле определения 6) исходной паре Д-операторов. Причем на входе полученного Д-оператора специфицированы и, таким образом, ему доступны все подлежащие обработке данные.

В силу полученной по построению функциональной эквивалентности результирующего Д-оператора $(D_3)\mathcal{R}_3(D'_3)$ исходной паре Д-операторов $(D_1)\mathcal{R}_1(D'_1)*(D_2)\mathcal{R}_2(D'_2)$ и в связи с тем, что на его входе специфицированы все подлежащие обработке данные $D_3 = D_1 \cup D_2$, в результате выполнения Д-оператора $\mathcal{R}_3$ будет получен как наблюдаемый, так и полный результаты (см. определение 9) такие, что $D'_3 = D'_1 \cup D'_2$ и $D^T_{i+1} = D^T_{i+2}$.

Поскольку результат исполнения построенного Д-оператора полностью совпадает с результатом выполнения исходной композиции Д-операторов, эти две конструкции эквивалентны и теорема, таким образом, доказана.

На основе этой теоремы построим для соотношений (21)–(25) производные Д-операторы:

$$(D_A)A(D'_A)*(D_B)B(D'_B) = (D_{\mathcal{R}})\mathcal{R}_1(D'_{\mathcal{R}}), \tag{26}$$

где $D_A \cup D_B = D_{\mathcal{R}}$, $D'_A \cup D'_B = D'_{\mathcal{R}}$;

$$(D_A)A(D'_A)*(D)P(\alpha^x) = (D_{\mathcal{R}})\mathcal{R}_2(D'_{\mathcal{R}}, \alpha^x), \text{ где } D_A = D_{\mathcal{R}},\ D'_A = D'_{\mathcal{R}}; \tag{27}$$

$$(D)\Lambda(D^d)P(\sigma^x) = (D_{\mathcal{R}})\mathcal{R}_3(\sigma^x), \text{ где } D = D_{\mathcal{R}}; \tag{28}$$

$$(D_A)A(D'_A)*(D_G)\Lambda(D^d) = (D_{\mathcal{R}})\mathcal{R}_4(D'_{\mathcal{R}}, D^d), \tag{29}$$

где $D_A = D_{\mathcal{R}}$, $D'_A = D'_{\mathcal{R}}$, $D'_A \cap D_G = \varnothing$.

Внутри полученных Д-операторов инкапсулирована некоторая часть функциональности исходных, а некоторые данные стали локальными, не наблюдаемыми вне производных Д-операторов. Это наглядно видно в случае (28), где производный Д-оператор внешне (по форме записи) не отличим от предиката и обладает существенно более широкой функциональностью.

Полученные Д-операторы, продуцирующие логические условия, могут использоваться в операциях (1)–(3). Продемонстрируем это на примере Д-оператора (27) и p_3-дизъюнкции

$$[(D_{\mathcal{R}})\mathcal{R}_2(D'_{\mathcal{R}}, \alpha^x)]((D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C)),$$

где выбору одного из возможных Д-операторов предшествуют обработка данных и получение результатов этой обработки.

Д-оператор, полученный в (29), может быть использован в сочетании с предикатом, и в этом случае p_3-дизъюнкция будет записана в виде

$$[(D_{\mathcal{R}})\mathcal{R}_4(D'_{\mathcal{R}}, D^d)*(D^d)P(\alpha^x)]((D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C)),$$

где выбору одного из возможных Д-операторов предшествуют обработка данных, получение ее результатов и прогнозирование результатов выполнения некоторого Д-оператора $(D_G)G(D'_G)$, которому эквивалентен Д-оператор $(D_G)\Lambda(D^d)$. При этом процедура прогнозирования инкапсулирована внутри производного Д-оператора.

Процесс построения производных Д-операторов может быть продолжен; в частности, Д-оператор (26) может быть объединен с предикатом, в результате получим

$$(D_{\mathcal{R}})\mathcal{R}_4(D'_{\mathcal{R}}, D^d)*(D^d)P(\alpha^x) = (D_{\mathcal{R}})\mathcal{R}_5(D'_{\mathcal{R}}, \alpha^x).$$

Полученный Д-оператор, по форме записи не отличимый от оператора в (27), отличается существенно более широкой функциональностью.

На основании изложенного сделаем следующий промежуточный вывод.

Построенные производные Д-операторы представляют собой более общий случай относительно исходных, так как позволяют абстрагироваться от некоторой функциональности (особенностей выполнения) и некоторых данных. Процесс построения производных Д-операторов может быть продолжен неограниченно, что позволит получать все более крупные алгоритмические компоненты, в частности и в случае (26). Объединяться могут любые Д-операторы, однако наиболее практичным представляется объединение связанных (см. определение 7) Д-операторов. Полученный результат представляется достаточно перспективным относительно решения задачи построения алгоритмов и удовлетворяет последнему требованию из введения.

Завершая данный раздел, построим производную операцию как частный случай p_2-дизъюнкции, которую назовем p-фильтрацией (последовательная фильтрация) и запишем ее в виде

$$[(D)P(\alpha^2)]((D_A)A(D'_A) \vee Z) = \begin{cases} (D_A)A(D'_A), & \text{если } \alpha^2 = 1; \\ Z, & \text{если } \alpha^2 \neq 1. \end{cases}$$

Результатом выполнения данной операции является Д-оператор $(D_A)A(D'_A)$ при $\alpha_2 = 1$, и эта операция порождает тождественный Д-оператор в противном случае. Все ранее рассмотренные допустимые композиции Д-операторов и производные Д-операторы могут быть использованы в этом случае и записываются аналогично.

Наконец, определим способ описания алгоритмов в рамках построенного алгебраического аппарата.

Определение 10. Представление любого Д-оператора из U через образующие элементы системы $\langle U, l, \Omega \rangle$ называется регулярной схемой этого Д-оператора (РСД).

ЗАКЛЮЧЕНИЕ

На основе модели ЭВМ построена система алгоритмических алгебр, базирующаяся на данных $\langle U, l, \Omega \rangle$, где U — множество Д-операторов, l — множество

логических условий, Ω — сигнатура операций, состоящая из логических операций Ω_1, принимающих значения на множестве l, и операций Ω_2, принимающих значения на множестве Д-операторов U.

Д-операторы из множества U введены таким образом, что для них специфицируются входные и выходные данные. В результате обеспечена возможность определять не только последовательность действий, задаваемую алгоритмом, но и данные, обрабатываемые каждым Д-оператором. Следовательно, в рамках одной РСД описываются оба аспекта разработки алгоритма.

Предикаты, продуцирующие двухзначные и трехзначные логические условия, введены таким образом, что эти логические условия всегда определены. В результате операции САА/Д определены при любых значениях (в том числе некорректных) данных, получаемых при их обработке.

Показана возможность на основе базовых Д-операторов и операций строить производные алгоритмические конструкции, расширяющие, с одной стороны, возможности исходных, а с другой — строить Д-операторы и операции, функциональность которых интегрирует их функциональные возможности, т.е. формализован подход к построению алгоритмов «снизу–вверх».

Таким образом, заложены основы САА/Д, которая отвечает сформулированным во введении требованиям, и получены предпосылки к созданию парадигмы, сочетающие особенности двух упомянутых во введении подходов к программированию. При этом очевидно, что предложенный формальный аппарат нуждается в дальнейшей разработке. В частности, использование трехзначных логических условий для повышения надежности алгоритмов не позволило реализовать их преимущества в случаях, рассматриваемых, например, в работах [6, 7], где утверждается, что трехзначная логика в большей степени, чем двухзначная, свойственна человеческому мышлению и во многих случаях обеспечивает компактность и выразительность алгоритмов (и программ), в которых она используется. Для того чтобы обеспечить решение обеих задач, необходимо повысить значность используемой логики, что является важным направлением дальнейших исследований.

Важным направлением в развитии САА/Д является также изучение возможности преобразований, создаваемых с помощью формального аппарата алгоритмов. Кроме того, построение САА/Д порождает задачу разработки методологии проектирования алгоритмов, в частности задачу формализации декомпозиции и модуляризации алгоритмов, а также задачу формализации структур и потоков данных в алгоритмах. Эти и многие другие задачи определяют дальнейшие шаги в указанном направлении.

СПИСОК ЛИТЕРАТУРЫ

1. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. — К.: Наук. думка, 1978.—319 с.
2. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. — К.: Академперіодика, 2007. — 634 с.
3. Акуловский В.Г. Расширенная алгебра алгоритмов // Проблеми програмування. — 2007. — № 3 — С. 3–15.
4. Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. — М.: Финансы и статистика, 1989. — 208 с.
5. Цейтлин Г.Е. Введение в алгоритмику. — К.: Сфера, 1998. — 310 с.
6. Брусенцов Н.П. Трехзначная диалектическая логика // Программные системы и инструменты: Темат. сб. — М.: МГУ, 2001. — № 2. — С. 36–44.
7. Брусенцов Н.П., Владимирова Ю.С. Троичное конструктное кодирование булевых выражений // Там же. — 2002. — № 3. — С. 6–10.

Поступила 27.05.2010