УДК 519.681

ОСНОВЫ АЛГЕБРЫ АЛГОРИТМОВ, БАЗИРУЮЩЕЙСЯ НА ДАННЫХ

В.Г. Акуловский

Академия таможенной службы Украины.
49000, Днепропетровск, ул. Дзержинского 2\4.
Тел/Факс: (056) 745 5596, (0562) 471 791
e-mail: academy@amsu.dp.ua

Предложена модель ЭВМ, у которой информационное множество представляет собой множество данных, образующих иерархию структур данных произвольной сложности. На основе этой модели построена система алгоритмических алгебр и рассмотрены некоторые аспекты и перспективы её использования.

The model of the computer at which the information set represents set of the data forming hierarchy of structures of the data of any complexity is offered. On the base of this model the system of algorithmic algebras basing the data is constructed, some which prospects of use are considered in the report.

Введение

Известно, что абстрактная модель ЭВМ Глушкова [1, 2] представляет собой схему взаимодействия двух автоматов управляющего *У* и операционного *О*, а множество состояний операционного автомата *М* – его информационное множество. Выходные сигналы *А* управляющего автомата отождествляются с операторами, которые изменяют состояние операционного автомата, а выходные сигналы операционного автомата, представляют собой значения различных элементарных логических условий α, определенных на множестве состояний операционного автомата.

Для формализации представления алгоритмов функционирования абстрактной модели ЭВМ В.М. Глушков предложил математический аппарат – систему алгоритмических алгебр (САА) [2, 3]. Этот формальный аппарат нашел широкое применение и получил дальнейшее развитие в многочисленных работах его учеников и последователей (например, [4, 5]).

Вместе с тем, в работе [3] высказана мысль (которая не нашла своего развития) о том, что при различных дополнительных предположениях об автоматах *У* и *О* функционирование модели может интерпретироваться по-разному.

Развивая эту мысль, будем утверждать, что модификация модели ЭВМ позволяет построить алгебраический аппарат, возможности которого определяются выбранной моделью. И таким образом может быть построен формальный аппарат, отвечающий некоторым сформулированным требованиям и ориентированный на решение некоторых задач, в том числе методологических.

В качестве примеров задач такого рода упомянем следующие задачи.

В программировании известны два подхода к разработке: от данных и от управления, которые существуют независимо и, более того, конкурируют между собой. Очевидно существуют классы программных систем, при разработке которых один из упомянутых подходов наиболее эффективен. Однако легко представить такие системы, для которых выбор одного из подходов будет неоднозначен, и уместным было бы комплексное использование этих подходов.

Весьма важной и перспективной представляется задача формализации информационных связей в алгоритмах и программах, так как в случае ее решения открываются пути к контролю корректности и преобразованию алгоритмов и программ на основе анализа таких связей. В частности, вероятно это позволит разрешить некоторые аспекты проблемы распараллеливания алгоритмов и программ.

Кроме того, в ряде работ Н.П. Брусенцова [6–8], наряду с критикой современных ЭВМ, показаны преимущества трехзначной логики (по сравнению с двузначной) при решении многих задач. Более того, в этих (и других) работах утверждается и показывается, что трехзначная логика не только вполне корректна, но является даже более удобной и привычной для людей формой мышления.

Для построения алгебры алгоритмов, в рамках которой могут быть решены эти и другие подобные задачи, рассмотрим модель ЭВМ, модифицированную, в соответствии с идеей, предложенной в [3]. А именно, будем интерпретировать множество состояний операционного автомата как множество состояний обрабатываемых алгоритмом данных.

Модель ЭВМ

Будем полагать, что автомат *О* содержит множество данных *D*, которое будем трактовать как его информационное множество. Носителем этих данных является множество абстрактных атомарных (неделимых) элементов памяти (в дальнейшем элементов памяти) $\Delta = \{\delta_1, \delta_2, ..., \delta_n\}$. Каждому элементу δ_i сопоставим

конечное множество всех возможных значений $I_{\delta_i}=\bigcup_{s=1}^{k} {}_s i_{\delta_i}$ и будем говорить, что множество I_{δ_i} отображается на одноэлементное множество δ_i. Или другими словами, что каждый элемент δ_i может принимать любое значение ${}_j i_{\delta_i} \in I_{\delta_i}$. Одновременно любой элемент δ_i может принимать одно и только одно значение из множества I_{δ_i}.

Элементы, образующие множество Δ, могут быть, как одинаковыми (однотипными), так и различными (разнотипными), т. е. $\Delta=\{\Delta_1,\Delta_2,...,\Delta_n\}$, где для $\forall\delta_i^p \in \Delta_p$ и $\forall\delta_j^s \in \Delta_s$ в первом случае выполняется соотношение $I_{\delta_i^p}=I_{\delta_j^s}$, а во втором – $I_{\delta_i^p}\neq I_{\delta_j^s}$.

Теперь введем понятие данных, причем будем различать элементарные и простые данные.

Для того чтобы определить элементарные данные, введем множество идентификаторов $D_Э \subseteq D$ и некоторому атомарному элементу памяти $\delta_i \in \Delta_э$ (где $\Delta_Э \subseteq \Delta$) поставим в соответствие элемент $d_j \in D_Э$ (в общем случае не единственный), который назовем идентификатором переменной или просто переменной. Переменной d_j поставим в соответствие множество допустимых значений $I_{d_j}=\bigcup_{l=1}^{p} {}_l i_{d_j}$, такое, что $I_{d_j}\subseteq I_{\delta_i}$ и будем говорить, что множество I_{d_j} отображается на одноэлементное множество d_j или, что переменная d_j принимает одно из допустимых значений ${}_l i_{d_j} \in I_{d_j}$, что будем записывать в виде ${}_l i_{d_j} \to d_j$. В том случае, когда $I_{d_j}=I_{\delta_j}$, будем говорить, что для переменной d_j допустимы любые значения или, что для переменной d_j множество допустимых значений не ограничено.

Для того чтобы определить простые данные, введем множество идентификаторов $D_P \subseteq D$ и будем рассматривать некоторые совокупности однотипных элементов памяти $\delta_{n+1},\delta_{n+2},...,\delta_{n+l}$, где каждый из элементов памяти δ_{n+i} может принимать значения из множества всех возможных значений $I_{\delta_{n+i}}=\bigcup_{s=1}^{k} {}_s i_{\delta_{n+i}}$, а совокупность элементов $\delta_{n+1},\delta_{n+2},...,\delta_{n+l}$ принимают значения из множества $I_i^{П} = \bigcup_{p=n+1}^{n+l} {}_p I_{\delta_{n+i}}$. Каждой совокупности элементов $\delta_{n+1},\delta_{n+2},...,\delta_{n+l}$ поставим в соответствие элемент $d_j^{П} \in D_{П}$ (в общем случае не единственный), который назовем идентификатором простой переменной или простой переменной. А каждому $d_j^{П}$ в соответствие поставим множество допустимых значений $I_{d_j^{П}}=\bigcup_{p=n+1}^{n+l}\bigcup_{s=1}^{m} {}_p i_s$, такое, что $I_{d_j^{П}}\subseteq I_i^{П}$. В том случае, когда $I_{d_j^{П}}=I_i^{П}$, будем говорить, что для переменной d_j допустимы любые значения или, что для переменной d_j множество допустимых значений не ограничено.

Дальнейшие рассуждения будем осуществлять в предположении, что элементарные и простые данные могут быть сгруппированы с целью построения более сложных – составных данных.

Множество семантически связанных элементарных и простых данных, образующих составные данные первого уровня группировки запишем в виде $D_j^1=\{d_1,d_2,...,d_i,...,d_n\}$ и $D_i^1=\{d_1^{П},d_2^{П},...,d_j^{П},...,d_k^{П}\}$, где D_j^1 и D_i^1 – идентификаторы составных данных первого уровня группировки, а d_i и $d_j^{П}$ - идентификаторы элементарных и простых данных, обладающие вышеописанными свойствами.

Этот процесс может быть продолжен неограниченно, что позволяет строить составные данные с произвольным уровнем (степенью) группировки, т. е. $D_p^k=\{D_1^{k-1},D_2^{k-1},...,D_s^{k-1}\}$.

С учетом вышеизложенного введем понятие структур данных.

Структуры данных на некотором уровне группировки в общем случае включают в себя семантически связанные элементарные, простые и составные данные произвольного уровня группировки, а так же структуры данных, полученные на предыдущих уровнях группировки.

Таким образом, множество состояний операционного автомата *D* – это в общем случае иерархия структур данных произвольной степени вложенности и сложности.

Определив данные и некоторые их свойства, вернемся к модели ЭВМ.

Выходные сигналы автомата *У – Д*-операторы, образующие множество *U*, поступающие на вход автомата *O* и определенные (в общем случае частично определенные) на множестве данных D.

Будем говорить, что *Д*-оператор изменяет состояние операционного автомата, изменяя (преобразуя) хранимые им данные из множества D. Если множество $D_j \subseteq D$ является областью определения, а множество $D'_j \subseteq D$ – областью значения, то некоторый *Д*-оператор может быть записан в виде $A(D_j) = D'_j$, где $A \in U$, $D_j, D'_j \subseteq D$.

В дальнейшем *Д*-оператор будем записывать в виде (более удобном) $(D)A(D')$, множество данных D, будем называть входными, а множество D' – выходными данными *Д*-оператора.

Д-оператор определим следующим образом.

Определение 1. *Д*-оператор $(D_A)A(D'_A)$ преобразует множество входных данных $D_A \subseteq D$, которое имело значения из множества значений ${}_D I_l \subseteq_D I$, в множество выходных данных $D'_A \subseteq D$, которое принимает множество значений ${}_{D'} I_p \subseteq_{D'} I$.

Теперь введем понятие неопределенного оператора.

Определение 2. *Д*-оператор $(D_A)A(D'_A)$ будем называть неопределенным, если при $\forall_{D_A} I_l$ вычислительный процесс переходит в неопределенное состояние. Такой оператор будем обозначать N.

Заметим, что под неопределенным состоянием вычислительного процесса будем понимать утрату управления, нарушение последовательности вычислений и остановку (зацикливания) этого процесса.

Множество переменных D'_A (со своими значениями), специфицированных на выходе оператора $(D_A)A(D'_A)$, до его выполнения обозначим D''_A и определим понятие тождественного оператора.

Определение 3. Оператор $(D_A)A(D'_A)$ будем называть тождественным, если для множества данных D'_A, выполняется соотношение $D''_A = D'_A$, т. е., если оператор не изменяет данные и, таким образом, не влияет на состояние операционного автомата. Такой оператор будем обозначать *E*.

Из этого определения следует, что *Д*-операторы вида $(\varnothing)A(\varnothing)$ и $(D_A)A(\varnothing)$ – тождественные.

Далее будем полагать, что на множестве *D* определены предикаты, образующие два множествами: множеством двузначных – P_2 и множеством трехзначных – P_3 предикатов, таких, что $p_2(D) = \alpha_2 \in E_2 = \{0,1\}$, где $p_2(D) \in P_2$, $p_3(D) = \alpha_3 \in E_3 = \{0,1,\mu\}$, где $p_3(D) \in P_3$, а $\alpha_3 = \mu$ когда $\alpha_3 \notin \{0,1\}$. Операционный автомат, вычисляя эти предикаты, формирует на своем выходе элементарные логические условия, которые поступают на вход управляющего автомата. Заметим, что вычисление предикатов не ведет к изменению информационного множества *D* автомата *O*.

Завершая рассмотрение модели ЭВМ, сформулируем некоторые полученные результаты.

В рамках предлагаемой модели в качестве информационного множества автомата *O* выступают данные (множество *D*), которые образуют структуры любой степени сложности. Введенные *Д*-операторы и предикаты определенны на множестве *D*. Причем предикаты продуцируют как двузначные, так и трехзначные логические условия. Логические условия и *Д*-операторы являются входными и выходными сигналами, которыми обмениваются автоматы, образующие рассматриваемую модель ЭВМ. При этом *Д*-операторы изменяют данные и таким образом, изменяют состояние информационного множества автомата *O*, а логические условия характеризуют состояние этих данных, т. е. состояние упомянутого информационного множества.

Основы алгебры алгоритмов

Пусть <U, W, Ω> – система алгоритмических алгебр, базирующаяся на данных (САА\Д), где *U* – множество операторов; *W* – множество логических условий, Ω – сигнатура операций, состоящая из логических операций – Ω_1, принимающих значения на множестве *W* и операций – Ω_2, принимающих значения на множестве операторов *U*.

На множестве *W* определены известные (см., например, [3]) операции дизъюнкция, конъюнкция и отрицание.

На множестве *U* введем следующие операции, причем таким образом, чтобы они были всюду определены.

Операцию *p*-дизъюнкции будем рассматривать в следующих вариантах, зависящих от вида используемого предиката.

$$[p_3(\tilde{D})]((D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C) = \begin{cases} (D_A)A(D'_A), \text{ если } \alpha_3 = 1; \\ (D_B)B(D'_B), \text{ если } \alpha_3 = 0; \\ (D_C)C(D'_C), \text{ если } \alpha_3 = \mu; \end{cases},$$

где $(D_A)A(D'_A),(D_B)B(D'_B),(D_C)C(D'_C) \in U$, $\tilde{D}, D_A, D'_A, D_B, D'_B, D_C, D'_C \subseteq D$, $p_3(\tilde{D}) \in P_3$, $p_3(\tilde{D}) = \alpha_3$, $\alpha_3 \in W$.

Результатом выполнения этой конструкции является выполнение одного из трех возможных операторов, который выбирается в соответствии со значением логического условия α_3, принимающего истинные значения трехзначной логики E_3. В качестве предиката в операторе могут выступать совокупности предикатов, тогда продуцируемые ими логические условия связываются логическими операциями множества Ω_1.

$$[p_2(\tilde{D})]((D_A)A(D'_A) \vee (D_B)B(D'_B)) = \begin{cases} (D_A)A(D'_A), \text{ если } \alpha_2 = 1; \\ (D_B)B(D'_B), \text{ в противном случае,} \end{cases}$$

где $p_2(\tilde{D}) \in P_2$, $\alpha_2 \in W$.

Результатом выполнения этой конструкции является выполнение одного из двух возможных операторов, который выбирается в соответствии со значением логического условия α_2, принимающего истинные значения двузначной логики E_2. При этом первый из операторов выполняется при истинном значении логического условия, а второй – во всех остальных случаях. В качестве предиката в операторе могут выступать совокупности предикатов, тогда продуцируемые ими логические условия связываются логическими операциями множества Ω_1.

Частным случаем *p*-дизъюнкции является операция *p*-фильтрация (последовательная фильтрация), которая вводится следующим образом:

$$[p_2(\tilde{D})]((D_A)A(D'_A 0 \vee E) = [p_2(\tilde{D})]((D_A)A(D'_A)) = (D_A)A(D'_A), \text{ если } \alpha_2 = 1,$$

где $p_2(\tilde{D}) \in P_2$, $\alpha_2 \in W$.

Результатом выполнения этой конструкции является выполнение одного оператора при условии истинности логического условия α_2.

Следующая операция, которую мы рассмотрим – *p*-итерация и введем следующим образом.

$_{p_2(\tilde{D})}\{(D_A)A(D'_A)\}$ – операция *p*-итерации состоит в вычислении предиката $P_2(\tilde{D})$ и проверке полученного логического условия α_2. Если α_2 истинно, выполняется 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 оператора, осуществляется до тех пор, пока условие $\alpha_2 = 1$ в противном случае операция *p*-итерации завершается.

Операция композиции (обозначается "*"), определим следующим образом.

Композиция операторов $(D_A)A(D'_A)*(D_B)B(D'_B)$ означает последовательное выполнение сначала оператора $(D_A)A(D'_A)$ и затем оператора $(D_B)B(D'_B)$.

Эта операция обладает следующими свойствами:

$$E*(D_A)A(D'_A) = (D_A)A(D'_A)*E = (D_A)A(D'_A);$$

$$N*(D_A)A(D'_A) = (D_A)A(D'_A)*N = N,$$

где *N* – неопределенный, а *E* – тождественный операторы.

Теперь определим регулярную схему *Д*-операторов.

Определение 4. Представление любого *Д*-оператора из *U* через образующие элементы системы <*U, W,* Ω> называется регулярной схемой этого *Д*-оператора (РСД).

В рамках РСД операция композиции обладает следующими свойствами:

$$E*ОП = ОП*E = ОП;$$

$$N*ОП = ОП*N = N;$$

где *ОП* – произвольная операция из Ω_2, *N* – неопределенный, а *E* – тождественный операторы.

Исходя из определения 4 и приведенных свойств, будем утверждать следующее.

Утверждение. Любая операция множества Ω_2 может рассматриваться как некоторый *Д*-оператор.

Доказательство. Некоторый оператор можно представить в виде

$$(D_A)A(D'_A) = (D_B)B(D'_B)*ОП,,$$

где *ОП* произвольная операция. В случае если оператор $(D_B)B(D'_B)$ тождественный, то, в соответствии со свойством операции композиция, это выражение будет выглядеть, как $(D_A)A(D'_A) = E*ОП = ОП.$

Следствие. Любую операцию множества Ω_2 можно записывать (оформлять) в виде *Д*-оператора.

На основании доказанного утверждения введем понятие композиционной схемы *Д*-операторов.

Определение 5. Регулярная схема *Д*-оператора, в которой используется единственная операция – композиция, а остальные операции множества Ω_2 записаны в виде *Д*-операторов, называется композиционной схемой (КС) этого *Д*-оператора.

Таким образом, мы заложили основы алгебраического аппарата, который позволяет представлять алгоритмы, специфицируя при этом обрабатываемые данные. В частности, определили *Д*-операторы, основные операции алгебры, регулярные схемы *Д*-операторов и их композиционные схемы.

Далее рассмотрим вопросы, связанные с формализацией информационных связей в алгоритмах.

Информационные связи в композиционных схемах и их преобразование

В рамках композиционных схем будем решать вопросы о взаимосвязи между операторами, образующими эти схемы. Заметим, что формализации информационных связей, контролю корректности и преобразованию алгоритмов посвящены работы [9–11], и в данном случае мы воспользуемся результатами полученными в этих работах.

Пусть КС представляет собой следующее выражение:

$$(D)A(D') = (D_1)A_1(D_1')*(D_2)A_2(D_2')*...*(D_i)A_i(D_i')*...*(D_n)A_n(D_n').$$

Определим понятие информационных связей в КС.

Определение 6. Операторы $(D_i)A_i(D_i')$ и $(D_j)A_j(D_j')$ (где i<j), входящие в КС, связаны, если для них выполняется соотношение: $D_i' \cap D_j \neq \varnothing$, т. е. некоторое подмножество выходных данных оператора $(D_i)A_i(D_i')$ поступает на вход оператора $(D_j)A_j(D_j')$. Будем говорить, что множество данных ${}_i\vec{D}_j = D_i' \cap D_j$ (${}_i\vec{D}_j \subseteq D_i'$ и ${}_i\vec{D}_j \subseteq D_i$) связывает операторы A_i и A_j (на что указывают используемые индексы) и эти данные будем называть связывающими.

Из определения 6 следует, что данные могут передаваться от любого оператора к одному или нескольким следующим за ним операторам, т. е. передаются слева направо и, что именно связывающие данные образуют информационные связи в КС. При этом в общем случае, некоторый оператор может быть связан со всеми следующими за ним и всеми предшествующими ему операторами.

Используя определение 6, специфицируем в КС информационные связи. Для этого детализуем входные и выходные данные операторов и дополним систему обозначений следующим образом. Данные на входе *j*-го оператора, связывающие его с *k*-м, будем обозначать ${}_k\vec{D}_j$ (где *k* – "адрес источника", *j* – "адрес приемника" данных), а на выходе, связывающие его с *p*-м – ${}_j\vec{D}_p$ (где *j* – "адрес источника", *p* – "адрес приемника" данных).

Перепишем КС с учётом введенных обозначений:

$$(D)\mathrm{A}(D') = (D_1)A_1(D_1', {}_1\vec{D}_2, {}_1\vec{D}_3, ..., {}_1\vec{D}_j, ..., {}_1\vec{D}_n)*(D_2, {}_1\vec{D}_2,)A_2(D_2', {}_2\vec{D}_3, ..., {}_2\vec{D}_j, ..., {}_2\vec{D}_n)*...$$
$$...*(D_j, {}_1\vec{D}_j, .., {}_l\vec{D}_j, ,..., {}_{j-1}\vec{D}_j)A_j(D_j', {}_j\vec{D}_{j+1}, ..., {}_j\vec{D}_p, ..., {}_j\vec{D}_n)*...*(D_n, {}_1\vec{D}_n, .., {}_j\vec{D}_n, ..., {}_{n-1}\vec{D}_n)A_n(D_n').$$

В построенной КС не только специфицированы данные и связи между операторами, но и все "источники" и "приемники" данных поставлены в однозначное соответствие, т. е. любому ${}_l\vec{D}_p^1$ соответствует ${}_l\vec{D}_p^1$. Поскольку в КС, как правило, не все операторы связаны друг с другом любое из множеств ${}_l\vec{D}_p^1$ может быть пустым и, соответственно, пустым будет и множество ${}_l\vec{D}_p^1$.

На основе специфицированных связей будем решать задачу оптимизирующего преобразования алгоритмов.

Рассмотрим информационные связи между операторами в КС и определим некоторые связанные с этим понятия.

Определение 7. Множество $S_j^л = {}_1\vec{D}_j, {}_2\vec{D}_j, .., {}_l\vec{D}_j, ,..., {}_{j-1}\vec{D}_j$ назовем множеством левых связей *j*-го оператора, а множество $S_j^{пр} = {}_j\vec{D}_{j+1}, {}_j\vec{D}_{j+2}, ..., {}_j\vec{D}_p, ..., {}_j\vec{D}_n$ – множеством его правых связей. В связи с этим, будем различать три типа операторов, которые назовем:

- связанные, у которых $S_j^л \neq \varnothing$ и $S_j^{пр} \neq \varnothing$;

- связанные слева (справа), у которых $S_j^л \neq \varnothing$, а $S_j^{пр} = \varnothing$ ($S_j^{пр} \neq \varnothing$, а $S_j^л = \varnothing$);

- несвязанные, у которых $S_j^л = \varnothing$ и $S_j^{пр} = \varnothing$.

Множество $S_j^o = {}_1\vec{D}_{j+1}, ..., {}_1\vec{D}_n, {}_2\vec{D}_{j+1}, ..., {}_2\vec{D}_n, ..., {}_{j-1}\vec{D}_{j+1}, ..., {}_{j-1}\vec{D}_n$ назовем множеством связей, охватывающих (огибающих) *j*-й оператор. Количество огибающих связей оператора A_j, определим как: $K_j^o = \left|S_j^o\right|$. Любое из множеств ${}_j\vec{D}_l$ (${}_r\vec{D}_s$), входящих в множества $S_j^л$, $S_j^{пр}, S_j^o$, может быть пустым и в этом случае оно из рассмотрения исключается. Из чего следует, что множества $S_j^л$, $S_j^{пр}, S_j^o$ так же могут быть пустыми.

Теперь введем и определим понятие длины информационных связей.

Определение 8. Длину произвольной связи между операторами A_i и A_j определим для случая i<j, как ${}_i d_j = j - i$, а для случая j<i, как ${}_i d_j = i - j$.

Исходя из определений 7 и 8, общую (суммарную) длину левых (правых) связей оператора A_k запишем в виде $L_k^л = \sum_{p=1}^{k-1} {}_p d_k$ ($L_k^{пр} = \sum_{p=k+1}^{n} {}_k d_p$), а общую длину огибающих k-й оператор связей в виде $L_k^o = \sum_{l=1}^{k-1} \sum_{r=k+1}^{n} {}_l d_r$. Во всех приведенных случаях любой из элементов ${}_i d_j$, входящих в приведенные выражения (в соответствии с определением 7), может быть равен нулю.

Учитывая тот факт, что, наряду с вышерассмотренными, существуют и другие виды связей, которые в [9] названы обратными, приведем определения локально независимых операторов.

Определение 9. Локально независимыми назовем любые два несвязанных оператора $(D)A(D')*(\hat{D})B(\hat{D}')$, если для них выполняются следующие ограничения: $D \cap \hat{D} = \varnothing$, $D \cap \hat{D}' = \varnothing$, $D' \cap \hat{D}' = \varnothing$, т. е. эти операторы не имеют никаких информационных связей между собой. В результате этого, для них выполняется тождественное соотношение: $(D)A(D')*(\hat{D})B(\hat{D}') = (\hat{D})B(\hat{D}')*(D)A(D')$, т. е. операторы могут перемещаться в рамках КС.

Далее, обобщая определение 9, введем понятие области независимости.

Определение 10. Областью независимости слева от оператора $(D_i)A_i(D_i')$, входящего в КС, назовем непрерывную последовательность операторов $(D_m)A_m(D_m')*...*(D_{i-1})A_{i-1}(D_{i-1}')$, где каждый из операторов, входящих в эту последовательность, не связан с данным и предшествует ему, и для каждого из которых выполняются соотношения из определения 9. В рассматриваемом (более общем) случае эти соотношения представляют собой следующее:

$$(\bigcup_{p=m}^{i-1} D_p) \cap D_i = \varnothing,\ (\bigcup_{p=m}^{i-1} D_p) \cap D_i' = \varnothing,\ (\bigcup_{p=m}^{i-1} D_p') \cap D_i' = \varnothing,$$

(где $m \le 1 - m$). Область независимости справа от оператора $(D_i)A_i(D_i')$ определяется аналогично и формально записывается в виде следующих соотношений:

$$D_i \cap (\bigcup_{p=i+1}^{r} D_p) = \varnothing,\quad D_i \cap (\bigcup_{p=i+1}^{r} D_p') \neq \varnothing,\quad D_i' \cap (\bigcup_{p=i+1}^{r} D_p') = \varnothing,$$

(где $r \ge i+1$). Будем говорить, что оператор $(D_i)A_i(D_i')$ обладает областью независимости, если для него выполняются приведенные соотношения.

Исходя из определений 9 и 10, будем утверждать, что каждый оператор в своей области независимости не имеет никаких информационных связей и может быть перемещен в рамках этой области.

На основании вышеизложенного докажем, что перемещение операторов в рамках их зоны независимости может обеспечить сокращение длины информационных связей в КС.

Теорема. В любой КС, содержащей операторы, обладающие областями независимости, длина информационных связей сократится в результате перемещения этих операторов в рамках их областей независимости при выполнении следующих условий. Если некоторый оператор A_i в результате перемещения и перенумерации операторов становится j-м (A_j), то в зависимости от его типа (см. определение 7), длина информационных связей в КС сократится при условии выполнения следующих соотношений:

1) для несвязанных операторов, при перемещении в произвольном направлении, $L_j^o < \hat{L}_i^o$, при этом перемещение 1-го и n-го (последнего) операторов к сокращению связей не приведут;

2) для связанных слева (справа) операторов, в случае перемещения их влево (вправо), $L_j^o < \hat{L}_i^o + \Delta_j^л$ ($L_j^o < \hat{L}_i^o + \Delta_j^{пр}$), а для 1-го и n-го соответственно $L_j^o < \Delta_1^{пр}$ и $L_j^o < \Delta_n^л$;

3) для связанных операторов в случае перемещения влево (вправо) $L_j^o + \overline{\Delta}_j^{пр} < \hat{L}_i^o + \Delta_j^л$ ($L_j^o + \overline{\Delta}_j^л < \hat{L}_i^o + \Delta_j^{пр}$).

Во всех приведенных случаях: L_j^o – длина связей, огибающих оператор A_j (в новом месте расположения), а $\hat{L}_i^o$ – длина этих связей в исходном месте расположения, после удаления оператора A_i, $\Delta_j^{пр}(\Delta_j^л)$ – величина, на которую сократится общая длина правых (левых) связей, $\overline{\Delta}_j^{пр}(\overline{\Delta}_j^л)$ – прирост общей длины правых (левых) связей в результате перемещения оператора в его новое место расположения.

Доказательство. При перемещении операторов в их области независимости на изменение длины связей в КС влияют три (и только три) фактора: сокращение общей длины огибающих связей в месте удаления оператора и её увеличение в месте вставки оператора, сокращение левых (правых) связей при перемещении оператора влево (вправо), удлинение левых (правых) связей при перемещении оператора вправо (влево). Никаким другим изменениям информационные связи не подвергаются.

Рассмотрим влияние этих факторов, причем рассмотрение первого из них разобъем на два этапа.

Первый этап. Во всех случаях, когда некоторый оператор A_i извлекается из последовательности операторов, образующих его область независимости, а операторы, входящие в КС, перенумеровываются, длина каждой отдельной, огибающей этот оператор, связи сократится на единицу. Это следует из определения 8 и того факта, что все индексы в диапазоне от i+1 до n уменьшатся на единицу. Суммарную длину огибающих связей, полученную после извлечения i-го оператора, запишем в виде $\hat{L}_i^o = L_i^o - K_i^o$ (где L_i^o – исходная длина этих связей, а K_i^o – их количество).

Второй этап. Во всех случаях, когда извлеченный ранее оператор A_i вставляется в последовательность операторов, образующих его область независимости, и после перенумерации получает номер j (A_j), каждая отдельная исходная связь, которая после вставки оператора A_j будет его охватывать, увеличится на единицу. Это следует из определения 8 и того факта, что все индексы в диапазоне от j+1 до n–1 увеличатся на единицу. Суммарную длину огибающих связей, полученную после вставки j-го оператора, запишем в виде $L_j^o = \tilde{L}_j^o + K_j^o$ (где $\tilde{L}_j^o$ – исходная длина связей, а K_j^o – их количество).

Исходя из изложенного, сделаем вывод, что влияние первого фактора на общую длину связей в РС (увеличение или сокращение) зависит от количества огибающих связей в исходном и результирующих местах расположения оператора, т. е. от выбора места, в которое будет перемещен оператор A_i.

Второй фактор. В результате перемещения оператора влево (вправо) длина его левых (правых) связей со всеми предшествующими (последующими) операторами сократится на величину перемещения, что докажем, основываясь на определении 10, следующим образом.

При перемещении оператора A_i влево на j-е место (вправо на p-е место), где j<i (p>i) длина некоторой левой (правой) связи между операторами A_j (A_p) и A_k (A_l), где k<j (l>p), будет определяться соотношением ${}_k d_j = {}_k d_i - {}_i d_j$ (${}_p d_l = {}_i d_l - {}_i d_p$). Общая длина левых (правых) связей оператора A_j (A_p) в этом случае будет определяться соотношением $L_j^л = L_i^л - \sum_{m=i}^{j-1} {}_m d_j$ ($L_p^{np} = L_i^{np} - \sum_{m=i}^{p-1} {}_m d_p$) и она, очевидно, меньше левых (правых) связей исходного оператора. Исходя из этого, величину, на которую сократится суммарная длина левых (правых) связей, запишем в виде $\Delta^л = L_i^л - L_j^л$ ($\Delta^{np} = L_i^{np} - L_p^{np}$).

Третий фактор. При перемещении оператора A_i влево на j-е место (вправо на p-е место), где j<i (p>i) длина связи между операторами A_j (A_p) и A_r (A_s), где r>i (s<i) будет определяться соотношением ${}_j d_r = {}_i d_r + {}_j d_i$ (${}_p d_s = {}_i d_s + {}_p d_i$). То есть, в обоих случаях связи удлиняются на величину перемещения. Общая длина левых (правых) связей оператора A_j (A_p) в этом случае будет определяться соотношением $L_j^л = L_i^л + \sum_{m=i}^{j-1} {}_m d_j$ ($L_p^{np} = L_i^{np} + \sum_{m=i}^{p-1} {}_m d_p$) и она, очевидно, больше длины левых (правых) связей исходного оператора. Исходя из этого, прирост общей длины левых (правых) связей запишем в виде $\overline{\Delta}^л = L_j^л - L_i^л$ ($\overline{\Delta}^{np} = L_p^{np} - L_i^{np}$).

Первый и последний операторы РС в соответствии с определениями 9,10 обладают следующими свойствами. У первого A_1 (последнего A_n) оператора РС множества $S_1^л$ и S_1^o (S_n^{np} и S_n^o) пусты и, соответственно, $L_1^л = 0$, $L_1^o = 0$ ($L_n^{np} = 0$, $L_n^o = 0$).

Таким образом, первый фактор влияет на длину связей во всех случаях, но в случае независимых операторов он является единственным, что и отражено в соотношении 1. Причем перемещение первого и последнего операторов, в связи с приведенными свойствами $L_1^o = 0$ и $L_n^o = 0$ (то есть ни одна связь их не охватывает), к сокращению длины связей не приведет.

Наряду с первым фактором, для связанных слева (справа) операторов в соотношении 2, учтено влияние второго фактора, а для связанных – влияние второго и третьего в соотношении 3. При этом первый и последний операторы не могут быть связанными в соответствии со свойствами $L_1^л = 0$ и $L_n^{пр} = 0$ и, в качестве таковых, не рассматриваются. В случае связанных слева (справа) операторов из свойства $L_1^o = 0$ и ($L_n^o = 0$) с очевидностью следует свойство $\hat{L}_1^o = 0$ ($\hat{L}_n^o = 0$), что и ведет к трансформации соотношения 2 к соответствующим частным случаям.

Поскольку в приведенных соотношениях влияние всех соответствующих факторов учтено для каждого типа операторов, то, очевидно, выбор нового места расположения операторов (если оно существует) с учетом этих соотношений с неизбежностью приведет к сокращению суммарной длины информационных связей в КС. Теорема, таким образом, доказана.

Заметим, что хотя возможность перемещение операторов ограничена областями независимости и, таким образом, исключено их взаимное влияние, на такое перемещение следует наложить ещё одно достаточно сильное ограничение. Это ограничение связано с необходимостью обеспечения требуемой алгоритмом последовательности действий, например, последовательности ввода-вывода данных, выполняемых операторами. Однако, поскольку и пока это ограничение не формализовано, контроль за его удовлетворением оставляем за разработчиком.

Заключение

На базе предложенной модели ЭВМ, у которой информационное множество представляет собой множество данных, образующих иерархию структур данных произвольной сложности, построена САА\Д и рассмотрены некоторые её аспекты. Особенности этого формального аппарата позволяют рассчитывать на решение задач приведенных во введении.

В частности, в рамках КС процесс декомпозиции операторов может осуществляться согласовано с детализацией данных, которые, как отмечалось, представляют собой иерархию некоторых структур данных. Такой подход позволяет учитывать как специфику обрабатываемых данных, так и специфику управления процессом их обработки. Можно сказать, что в рамках предлагаемого формального аппарата, может быть реализована парадигма программирования, сочетающая положительные черты проектирования алгоритмов и программ от данных с проектированием от управления.

При этом, выразительность и компактность записи алгоритмов в известной степени обеспечивается за счет использования трехзначной логики.

Формализация информационных связей в рамках композиционных схем позволяет осуществлять преобразование алгоритмов.

Перспективным направлением развития рассмотренного алгебраического аппарата является формализация потоков данных в алгоритмах и решение задач, связанных с их распараллеливанием на основе такой формализации. Кроме того, актуальной остается задача построения методологии программирования, основанная на использовании предложенной САА\Д.

1. *Глушков В.М.* Теория автоматов и формальные преобразования микропрограмм // Кибернетика. – 1965.– № 5. — С. 1–10.
2. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – К.: Наук. думка, 1978.– 319 с.
3. *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.
4. *Цейтлин Г.Е.* Введение в алгоритмику. – К.: “Сфера”, 1998. – 310 с.
5. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – К.: Академперіодика, 2007.– 634 с.
6. *Брусенцов Н. П.* Трехзначная диалектическая логика // Программные системы и инструменты. Тематический сборник № 2. – М.: Факультет ВМиК МГУ, 2001. – С. 36–44.
7. *Брусенцов Н. П., Владимирова Ю. С.* Троичное конструктное кодирование булевых выражений //Программные системы и инструменты. Тематический сборник № 3 – М.: Факультет ВМиК МГУ, 2002 – С. 6–10.
8. *Брусенцов Н. П.* Микрокомпьютеры. – М.: Наука, 1985. – С. 141–170.
9. *Акуловский В.Г.* Некоторые подходы к контролю и преобразованию алгоритмов на основе анализа специфицируемых данных // Проблеми програмування. – 2008. – № 4.–С. 84–93.
10. *Акуловский В.Г.* Некоторые аспекты преобразования алгоритмов на основе формализации информационных связей. // Кибернетика и системный анализ. – 2009.–№ 6.–С. 50–54.
11. *Акуловский В.Г.* Некоторые аспекты формализации данных и декомпозиция *Д*-операторов алгебры алгоритмов // Проблеми програмування. – 2009. – № 4. – С. 3 –10.