УДК 519.681

А.Е. Дорошенко, В.Г. Акуловский

ПОСТРОЕНИЕ И ПРЕОБРАЗОВАНИЕ ОПЕРАЦИЙ И НЕКОТОРЫХ АЛГОРИТМИЧЕСКИХ КОНСТРУКЦИЙ АЛГЕБРЫ АЛГОРИТМОВ С ДАННЫМИ

Рассмотрены свойства операций, входящих в сигнатуру алгебры алгоритмов с данными. Показана возможность создавать, в частности с помощью фиксирующих логических условий, производные операции, удобные для различных приложений и учитывающие эффективность реализации на целевом языке программирования. Показана возможность осуществлять оптимизирующие преобразования типичных алгоритмических конструкций.

Введение

Качество любой программы наиболее результативно обеспечивается на этапе разработки её алгоритма, а наиболее эффективным средством описания алгоритмов является алгебраический аппарат.

В работах [1–3] заложены основы алгебраического аппарата, в который, в результате модификации известной [4] модели ЭВМ Глушкова, “встроены” данные. Модификация состояла в оснащении указанной модели ЭВМ памятью – носителем данных. Данными D названа упорядоченная пара $<N,S>$, где N – носитель данных (фрагмент памяти), S – кортеж значений, хранимый этим носителем данных в текущий момент времени. В частном случае данные d называются простыми, когда они в текущий момент времени содержат единственное значение s. Данные, расположенные в памяти и обрабатываемые некоторым алгоритмом, представляют собой семейство множеств $D^P=\{D_1,D_2,...,D_n\}$. При этом отметим, что структура любых данных D_i не ограничена по сложности и объему.

Упомянутый формальный аппарат это – система алгоритмических алгебр (САА\Д) – $<U,L,W>$, где U – множество Д-операторов, L – множество логических условий, W – сигнатура, состоящая из логических операций W_1, принимающих значения на множестве L и операций W_2, принимающих значения на множестве Д-операторов U.

Принципиальные отличия предложенного формального аппарата от алгебры Глушкова состоят в следующем.

На входе и выходе Д-операторов, которые записываются в виде $(D)X(D')$, специфицированы обрабатываемые данные, в связи с чем и введено такое их название.

На множестве k-значных логических условий $l \in \acute{r}J_k = \{0,1,...,k-1\}$ определены известные (см., например, [5]) обобщенные операции:

- отрицание $\bar{l} = l + 1(mod\ k)$;
- дизъюнкция $l_1 \vee l_2 = max(l_1,l_2)$;
- конъюнкция $l_1 \wedge l_2 = min(l_1,l_2)$.

Множество значений, принимаемых логическими условиями дополнено логической константой «неопределенность», которая обозначена символом x. Такой, что $l=x$, если $l \notin \acute{r}J_k = \{0,1,...,k-1\}$.

Для выведенной константы логические операции выполняются следующим образом: $\bar{x} = x$, $l \vee x = 1$, $l \wedge x = x$.

В результате построены k+1-значные логические условия

$$l^k \in \acute{r}J_{k+1} = \{0,1,...,k-1,x\},$$

образующие множество L, на котором определены приведенные логические операции, образующие множество W_1.

Ключевым для САА\Д является понятие – состояние вычислительного процесса.

Определение 1. Вычислительный процесс на любом, например i-том, шаге выполнения находится в одном из состояний: $D_i^T=D_i^P$, $D_i^T=D_i^P \cup \{l_i^k\}$, где D_i^T –

текущее состояние вычислительного процесса, – текущее состояние памяти, l_i^k – логическое условие. Состояние памяти является составляющей состояния вычислительного процесса на каждом его шаге. Логические условия определены только на некоторых шагах вычислительного процессса, т. е. $\exists$, где $\{l_m^k\} \neq \varnothing$ в состоянии . Эта составляющая состояния вычислительного процесса существует только на данном шаге, т. е. для любого , где $\neq$, выполняется $\{l_m^k\} = \varnothing$.

Исходя из определения 1, введены Д-операторы.

Определение 2. Д-операторы образуют следующий базовый набор:

– $(D)O(D')$, для которого допустимо как $D \cap D' = \varnothing$, так и $D \cap D' \neq \varnothing$, переводит вычислительный процесс из любого исходного для него состояния D_i^T в состояние $D_{i+1}^T \neq D_i^T$, где $\{l_{i+1}^k\} = \varnothing$, $D_{i+1}^P \neq D_i^P$;

– $(D)P(l^k)$, который в дальнейшем будем называть предикатом, представляет собой в общем случае n-местную логическую функцию

$$P_{k+1}^n : D^o \to l^k \in E_{k+1} = \{0,1,\ldots,k-1,x\},$$

где D^o – n-арное (в общем случае) отношение, l^k – k+1-значное логическое условие, продуцируемое предикатом. $l^k = x$, если $l^k \notin E_k = \{0,1,\ldots,k-1\}$ или если предикат не определен на множестве данных D^o. Этот Д-оператор переводит вычислительный процесс из любого исходного для него состояния в состояние D_{i+1}^T такое, что

$$D_{i+1}^T = D_{i+1}^P \cup \{l_{i+1}^k\},\ \{l_{i+1}^k\} \neq \varnothing,\ D_{i+1}^P = D_i^P.$$

В работе [6] введены фиксирующие логические условия как средство фиксации состояния вычислительного процесса.

Множество логических условий $\subset$ такое, что все его элементы $(\forall f \in L_f)$ не изменяются при выполнении любого Д-оператора $(D)X(D') \in U$. Другими словами говоря, условия множества L_f сохраняют свои значения независимо от действия Д-операторов множества U, в связи с чем называются фиксирующими, так как позволяют зафиксировать (сохранить) характеристику текущего состояния данных, некоторое событие, имевшее место в ходе вычислительного процесса, или иметь любую другую трактовку.

Рассмотрим частный случай Д-оператора $(D)O(D')$ – $(\varnothing)Y^j(f_i^k)$, в результате выполнения которого фиксирующее условие f_i^k получает значение $j \in E_k = \{0,1,\ldots,k-1\}$. Поскольку в исходном состоянии (до выполнения $(\varnothing)Y^j(f_i^k)$) фиксирующее условие $f_i{}^k$ не определено, то фактически $f^k \in E_{k+1} = \{0,1,\ldots,k-1,x\}$.

Д-оператор $(\varnothing)Y^j(f_i^k)$ переводит вычислительный процесс из любого исходного для него состояния D_i^T в состояние $D_{i+1}^T \neq D_i^T$, где $\{l_{i+1}^k\} = \varnothing$, $D_{i+1}^P \neq D_i^P$.

Отметим, что фиксирующие логические условия не являются чем-то принципиально новым в программировании, так как они, по-видимому, всегда использовались на практике под именами “флажки”, ”признаки” и т. п. В данном случае это средство только формализовано в соответствующем контексте.

На множестве Д-операторов U определены следующие операции. При этом количество их минимально.

Операция композиция (обозначается “*”) Д-операторов $(D_1)O_1(D_1')*$ $*(D_2)O_2(D_2')$ означает последовательное выполнение сначала Д-оператора $(D_1)O_1(D_1')$, а затем Д-оператора $(D_2)O_2(D_2')$.

Операции p-дизъюнкции (в данном случае p_3-дизъюнкции, когда $l^3 \in E_{2+1} = \{0,1,x\}$)

$$[(D^o)P(l^3)]((D_1)O_1(D_1') \vee (D_2)O_2(D_2') \vee$$

$$\vee (D_3)O_3(D_3')) = \begin{cases} (D_1)O_1(D_1'), \text{ если } l^3 = 1; \\ (D_2)O_2(D_2'), \text{ если } l^3 = 0; \\ (D_3)O_3(D_3'), \text{ если } l^3 = x. \end{cases}$$

В случае фиксирующих логических условий

$$[f^3]((D_1)O_1(D_1')\vee(D_2)O_2(D_2')\vee$$
$$\vee(D_3)O_3(D_3'))=\begin{cases}(D_1)O_1(D_1'), \text{ если } f^3=1;\\ (D_2)O_2(D_2'), \text{ если } f^3=0;\\ (D_3)O_3(D_3'), \text{ если } f^3=x.\end{cases}$$

Хотя сигнатура САА\Д содержит минимальное число определенных на множестве U операций, их реализация такова, что позволяет строить производные операции и удобные для решения конкретных задач алгоритмические конструкции. Кроме того, имеет место система тождественных соотношений, которая позволяет осуществлять эквивалентные оптимизирующие преобразования операций и распространенных алгоритмических конструкций.

Рассмотрению указанных возможностей САА\Д посвящена данная работа.

Производные операции САА\Д

Рассмотрение производных операций начнем с определения тождественного и неопределенного Д-операторов.

Определение 3. Д-оператор $(D)X(D')$, в результате исполнения которого будет получено соотношение $D_i^T=D_{i+1}^T$, будем называть тождественным и обозначать Z.

Определение 4. Д-оператор $(D)X(D')$, который переводит вычислительный процесс из любого произвольного состояния D_i^T в неопределенное состояние, после перехода в которое все последующие шаги вычислительного процесса не определены, будем называть неопределенным и обозначать N.

Исходя из определений 1–3, приведем следующее тождественное соотношение, свойственное операции композиция:

$$(D)X(D')*Z=Z*(D)X(D')=(D)X(D'). \qquad (1)$$

Построим производную операцию p_2-дизъюнкции, имеющую аналоги в большинстве языков программирования:

$$[(\mathrm{D}^o)P(l^3)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2')\vee$$
$$\vee(D_2)O_2(D_2'))=\begin{cases}(D_1)O_1(D_1'), \text{ если } l^3=1,\\ (D_2)O_2(D_2'), \text{ если } l^3=0,=\\ (D_2)O_2(D_2'), \text{ если } l^3=x\end{cases}$$
$$=[(D^o)P(l^2)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2')).$$

Полученная таким образом производная операция приводит к выполнению Д-оператора $(D_1)O_1(D_1')$ при истинном значении логического условия (l^2=1) и $(D_2)O_2(D_2')$ при $l^2\neq1$, трактуемом как ложное, т. е. $l^2\in E_2=\{0,1\}$.

Воспользовавшись тождественным Д-оператором, построим ещё одну производную операцию

$$[(D^o)P(l^2)]((D_1)O_1(D_1')\vee Z)=$$
$$=\begin{cases}(D_1)O_1(D_1'), \text{ если } l^2=1\\ Z, \text{ если } l^2=0\end{cases}=$$
$$=[(D^o)P(l)](D_1)O_1(D_1'),$$

где $l^2\in E_2=\{0,1\}$. Эта операция играет важнейшую роль в дальнейшем изложении, в котором будем называть её последовательной фильтрацией (p-фильтром). Для неё справедливы следующие тождественные соотношения:

$$[(D^o)P(l)]([(D^o)P(\bar{l})](D)O(D'))=Z; \qquad (2)$$

$$[(D^o)P(l)]([(D^o)P(l)](D)O(D'))=$$
$$=[(D^o)P(l)](D)O(D'); \qquad (3)$$

$$[(D_1^o)P_1(l_1)]([(D_2^o)P_2(l_2)](D)O(D'))=$$
$$=[(D_1^o)P_1(l_1)\wedge(D_2^o)P_2(l_2)](D)O(D').$$

Производные операции реализуемы и в случае фиксирующих логических условий в виде:

$[f^2]((D_1)O_1(D_1')\vee(D_2)O_2(D_2'))$;

$[f](D_1)O_1(D_1')$.

Воспользовавшись операциями p_2-дизъюнкции и p_3-дизъюнкции, построим операцию p_k-дизъюнкции, которая

существенно расширит возможности алгебраического аппарата. Будем исходить из того, что D_1^o – n-арное отношение и для него выполняются следующие соотношения:

$$D_1^o \supset D_2^o \supset D_3^o \supset ... \supset D_{k-1}^o;$$
$$D_1^o = D_2^o \cup D_3^o \cup ... \cup D_{k-1}^o;$$
$$\dots$$
$$D_i^o = D_{i+1}^o \cup ... \cup D_{k-1}^o;$$
$$\dots$$
$$D_{k-2}^o \supset D_{k-1}^o.$$

Предикат $(D_i^o)P_i(l_i^2)$ (см.определение 2) логическая функция $P: D_i^o \to l_i^2 \in E_2 = \{0,1\}$, а $(D_i^o)P_i(l_i^3)$ – $P: D_i^o \to l_i^3 \in E_3 = \{0,1,x\}$.

Организовав вложенность операций p_2-дизъюнкции, а в качестве последней вложенной операцией использовав p_3-дизъюнкцию получаем

$$[(D_1^o)P_1(l_1^2)]((D_1)O_1(D_1') \vee ...$$
$$... \vee [(D_{k-2}^o)P_{k-2}(l_{k-2}^2)]((D_{k-2})O_{k-2}(D_{k-2}') \vee$$
$$\vee [(D_{k-1}^o)P_{k-1}(l_{k-1}^3)]((D_{k-1})O_{k-1}(D_{k-1}') \vee$$
$$\vee (D_k)O_k(D_k') \vee (D_{k+1})O_{k+1}(D_{k+1}'))...) =$$
$$= [(D_1^o)P(l^k)]((D_1)O_1(D_1') \vee (D_2)O_2(D_2') \vee ...$$
$$... \vee (D_{k+1})O_{k+1}(D_{k+1}')) =$$
$$= \begin{cases} (D_1)O_1(D_1'), \text{ если } l^k = k-1; \\ (D_2)O_2(D_2'), \text{ если } l^k = k-2; \\ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \\ (D_k)O_k(D_k'), \text{ если } l^k = 0; \\ (D_{k+1})O_{k+1}(D_{k+1}'), \text{ если } l^k = x, \end{cases}$$

где предикат $(D_1^o)P(l^k)$ логическая функция $P: D_1^o \to l^k \in E_k = \{0,1,k-1,x\}$ такая, что $l^k = k-1$ замещает $l_1^2 = 1$, $l^k = k-2$ замещает $l_2^2 = 1$ и т. д., а $l^k = 1$, $l^k = 0$, $l^k = x$ замещают, соответственно $l_{k-1}^3 = 1$, $l_{k-1}^3 = 0$, $l_{k-1}^3 = x$.

k+1-значные логические условия позволяют разветвлять вычислительный процесс по произвольному числу направлений.

Построенная операция реализуется и в случае фиксирующих логических условий

$$[f^k]((D_1)O_1(D_1') \vee (D_2)O_2(D_2') \vee ...$$
$$... \vee (D_{k+1})O_{k+1}(D_{k+1}')) =$$
$$= \begin{cases} (D_1)O_1(D_1'), \text{ если } f^k = k-1; \\ (D_2)O_2(D_2'), \text{ если } f^k = k-2; \\ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \\ (D_k)O_k(D_k'), \text{ если } f^k = 0; \\ (D_{k+1})O_{k+1}(D_{k+1}'), \text{ если } f^k = x. \end{cases}$$

Очевидна неполнота алгебры алгоритмов, которая не содержит операцию выполняющую цикл. Для восполнения этого пробела построим такую производную операцию следующим образом.

Определение 5. Операция p-итерации

$$[((D^o)P(l)]\{(D)O(D')\} =$$
$$= [((D^o)P(l)](D)O(D'^1) *$$
$$*[((D^{1o})P(l)](D^1)O(D'^2) * ...$$
$$... *[(D^o)P(l)]\{(D^k)O(D'^{k+1}) * ...,$$

представляют собой последовательность p-фильтров, связанных операцией композиция, которая обрывается в случае, когда предикат некоторого n-го р-фильтра $(D^{no})P(l)$ продуцирует ложное логическое условие (l=0). Где D^{io}, D^i – данные, полученные после выполнения (j-1)-го элементом последовательности, D'^{j+1} – после выполнения j-го элемента последовательности.

Иначе говоря, операции p-итерации осуществляют циклическое выполнение Д-оператора $(D)O(D')$ при $l = 1$ и завершается в противном случае. Логическое условие l будем называть условием выполнения цикла, Д-оператор $(D)O(D')$ – телом цикла, а каждый элемент последовательности p-фильтров – витком (шагом) цикла.

Докажем наличие ограничений на организацию циклов.

Теорема 1. Необходимым условием для завершения операций

p-итерации $[(D^o)P(l)]\{(D)O(D')\}$ является выполнение соотношений $D^o \cap D' \neq \varnothing$ и $D \cap D' \neq \varnothing$.

Доказательство построим от противного.

Из определений 1 и 2 следует, что *p*-фильтр последовательно переводит вычислительный процесс из исходного состояния D_i^T в состояние D_{i+1}^T и D_{i+2}^T. При этом предикат не изменяет состояния множества данных D^o, а Д-оператор изменяет состояние памяти таким образом, что допустимы соотношения $D \cap D' = \varnothing$ и $D \cap D' \neq \varnothing$.

Предположим, что имеет место соотношение $D^o \cap D' = \varnothing$ и на первом шаге предикат $(D^o)P(l)$ продуцирует логическое условие $l = 1$. В результате выполнения $(D)O(D')$ множество данных D^o не изменяется, так как $D^o \cap D' = \varnothing$, что приведет к выполнению второго и последующих элементов последовательности *p*-фильтров при неизменном D^o (см. определение 4), т. е. получим $(D^o)P(l) \equiv 1$ и, таким образом, последовательность *p*-фильтров становится бесконечной. То есть, происходит "зацикливание" вычислительного процесса.

Предположим, что $D^o \cap D' \neq \varnothing$, а $D \cap D' = \varnothing$, и на первом витке цикла $l = 1$. В результате выполнение тела цикла $(D)O(D')$, при условии $D^o \cap D' \neq \varnothing$, множество данных D^o изменится, и получаем множество данных D^{1o}. При условии получения $(D^{1o})P(l) = 0$ выполнение цикла завершится после первого шага. Если этого не произойдет ($(D^{1o})P(l) = 1$), то, в связи с $D \cap D' = \varnothing$, на втором и всех последующих шагах цикла множество данных D^i останется неизменным, что приведет к неизменности данных D'^{i+1} и D^{io}. Откуда следует $(D^{io})P(l) \equiv 1$ (i>1), т. е. зацикливание, аналогичное вышерассмотренному случаю. Таким образом, при втором предположении операция либо зацикливается, либо вырождается в *p*-фильтр.

Поскольку сделанные предположения противоречат условию теоремы, то соотношения $D^o \cap D' \neq \varnothing$ и $D \cap D' \neq \varnothing$ являются условиями, необходимыми для завершения операции *p*-итерации.

Теорема доказана.

Следствие 1. В операциях *p*-итерации условие выполнения цикла зависит от результатов выполнения его тела.

Исходя из определения операции и теоремы 1, приведем свойственные *p*-итерации тождественные соотношения:

$$- [0]\{(D)O(D')\} = Z;$$
$$- [1]\{(D)O(D')\} = N^*,$$

где 0 – тождественно ложное, 1 – тождественно истинное условие, Z – тождественный, N – неопределенный Д- оператор;

$- [(D^o)p(l)]\{Z\} = N$ или Z, в случае, когда на первом витке цикла $(D^o)p(l) = 0$;

$$- [(D^o)P(l)]\{[(D^o)P(\bar{l})]\{(D)O(D')\}\} = Z;$$
$$- \begin{array}{l}[(D^o)P(l)]\{(D)O(D')\} = \\ = [(D^o)P(l)]\{(D)O(D')\}\bar{l},\end{array} \quad (4)$$

т. е. по завершению цикла множество данных D^o получает такие значения, при которых $(D^o)P(l) = 0$.

Рассмотрим другие, представляющие интерес производных операций САА\Д.

К их числу относится в частности известная [7] операция – переключатель. Построим несколько вариантов этой операции, обладающих разной "мощностью". Для начала, воспользовавшись

---

* Следует обратить внимание на тот факт, что существует достаточно широкий класс программных систем (в частности, системы управления, реализуемые, как правило, на программируемых контроллерах), для которых бесконечный цикл, не только приемлем, но и необходим. Поэтому будем полагать, что существует операция цикла, для которой выполняется $[1]\langle(D)O(D')\rangle \neq N$, и которую в данной работе не рассматриваем.

вложенными операциями p_2-дизъюнкции и p-фильтром, получим переключатель

$$[(D_1^o)P_1(l^2)]((D_1)O_1(D_1')\vee$$
$$\vee[(D_2^o)P_2(l^2)]((D_2)O_2(D_2')\vee ...$$
$$...\vee[(D_k^o)P_k(l)]((D_k)O_k(D_k'))...) =$$
$$=\begin{pmatrix}(D_1^o)P_1(l^2),...,(D_k^o)P_k(\mathrm{l})\\(D_1)O_1(D_1'),...,(D_k)O_k(D_k')\end{pmatrix},$$

выполняющий i-й Д-оператор, которому соответствует истинное значение логического условия, продуцируемого предикатом $(D_i^o)P_i(l^2)$.

Воспользовавшись p_2-дизъюнкцией, в качестве последней вложенной операции, получаем переключатель

$$[(D_1^o)P_1(l^2)]((D_1)O_1(D_1')\vee ...$$
$$...\vee[(D_{k\text{-}1}^o)P_{k\text{-}1}(l^2)]((D_{k\text{-}1})O_{k\text{-}1}(D_{k\text{-}1}')\vee$$
$$\vee(D_k)O_k(D_k'))...) =$$
$$=\begin{pmatrix}(D_1^o)P_1(l^2),...,(D_k^o)P_k(l^2)\\(D_1)O_1(D_1'),...,(D_{k\text{-}1})O_{k\text{-}1}(D_{k\text{-}1}')\\(D_k)O_k(D_k')\end{pmatrix},$$

который отличается от предыдущего тем, что Д-оператор $(D_k)O_k(D_k')$ выполняется в случае, когда все предикаты продуцируют ложные логические условия.

Мощность переключателя возрастет в случае использования вложенных p_3-дизъюнкций

$$[(D_1)P_1(l^3)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2')\vee$$
$$\vee[(D_2^o)P_2(l^3)]((D_3)O_3(D_3')\vee(D_4)O_4(D_4')\vee ...$$
$$...\vee[(D_k^o)P_k(l^3)]((D_{2k-1})O_{2k-1}(D_{2k-1}')\vee$$
$$\vee(D_{2k})O_{2k}(D_{2k}')\vee(D_{2k+1})O_{2k+1}(D_{2k+1}'))...) =$$
$$=\begin{pmatrix}(D_1^o)P_1(l^2),...,(D_k^o)P_k(l^2)\\(D_1)O_1(D_1'),...,(D_{2k-1})O_{2k-1}(D_{2k-1}')\\(D_2)O_2(D_2'),...,(D_{2k})O_{2k}(D_{2k}')\\(D_{2k+1})O_{2k+1}(D_{2k+1}').\end{pmatrix}.$$

В данном случае, при истинном значении логического условия продуцируемого предикатом $(D_i^o)P_i(l^2)$, выполняется Д-оператор $(D_{2i-1})O_{2i-1}(D_{2i-1}')$, а при ложном – $(D_{2i})O_{2i}(D_{2i}')$. В случае неопределенного логического условия выполняется следующий предикат. Если логическое условие не определено после выполнения всех предикатов, то выполняется Д-оператор $(D_{2k+1})O_{2k+1}(D_{2k+1}')$.

Очевидно, что в переключателях могут быть задействованы фиксирующие логические условия, а процесс построения переключателей может быть продолжен.

Широко используемым в программировании, в связи с удобством решения широкого класса задач, является цикл *for*. Такой цикл реализуется с помощью следующей алгоритмической конструкции

$$(D_p)U(D_p')*[(D^o)P(\mathrm{l})]\{(D_l)O(D_l')*$$
$$*(D_p)S(D_p')\},$$

где Д-оператор $(D_p)U(D_p')$ устанавливает исходные значения параметров цикла D_p, а $(D_p)S(D_p')$ – изменяет их с заданным шагом. При условии выполнения соотношений $(D_1\cup D_1')\cap(D_\mathrm{p}\cup D_p')=\varnothing$ будет получен аналог операции *for*, который оформим традиционно в виде производной операции p-итерации

$$[(D_p)U(D_p');(D^o)P(\mathrm{l});$$
$$;(D_p)S(D_p')]\{(D_l)O(D_l')\}.$$

Построенная операция при заданных ограничениях позволяет обеспечить независимость условия выполнения цикла от его тела, в качестве которого рассматривается Д-оператор $(D_1)O(D_1')$ (см. следствие 1). Условием, необходимым для завершения цикла, является $D_p'\cap D^o\neq\varnothing$, что легко доказать по аналогии с теоремой 1.

К производным операциям относится обратная р-итерация (цикл с постусловием), в котором условие выполнения проверяется после выполнения тела цикла и который получим в результате следующего построения:

$$\{(D)O(D')\}[(D^o)P(\mathrm{l})] =$$
$$= (D)O(D')*[(D^o)P(\mathrm{l})]\{(D)O(D')\}.$$

Условием, необходимым для завершения цикла, является $D' \cap D^o \neq \varnothing$, что легко доказать по аналогии с теоремой 1.

Исходя из определения операций и условия завершения цикла, приведем тождественные соотношения, свойственные *p*-итерации с постусловием:

$\{(D)O(D')\}[0] = (D)O(D')$;
$\{(D)O(D')\}[1] = N$;
$\{Z\}[(D^o)p(l)] = N$ или Z, когда на первом витке цикла $(D^o)p(l) = 0$.

Ещё одна производная конструкция – обобщенный цикл [7]

$$\{(D_1)O_1(D_1')[(D)P(l)](D_2)O_2(D_2')\},$$

т. е. цикл с контролем условия выполнения, осуществляемом в середине цикла. Такая операция легко реализуема на алгоритмическом уровне в виде

$$\{(D_1)O_1(D_1')[(D)P(l)](D_2)O_2(D_2')\} =$$
$$= (D_1)O_1(D_1')*$$
$$*[(D)P(l)]\{(D_2)O_2(D_2')*(D_1)O_1(D_1')\}.$$

Однако, поскольку в большинстве языков программирования отсутствует такой оператор, то при записи приведенной реализации на языке программмирования проявится тот недостаток, что Д-оператор $(D_1)O_1(D_1')$ записывается дважды. Если это Д-оператор достаточно объемен, а операция используется достаточно часто, то это может привести к такому увеличению программы, что возникшие издержки сделают применение этой конструкции недопустимой.

Учитывая это, реализуем данную алгоритмическую конструкцию с помощью фиксирующего логического условия:

$$\{(D_1)O_1(D_1')[(D)P(l)](D_2)O_2(D_2')\} =$$
$$= (\varnothing)Y^1(f)*[f]\{(D_1)O_1(D_1')*$$
$$*[(D)P(l^2)]((D_2)O_2(D_2') \vee (\varnothing)Y^0(f))\}.$$

Данная реализация выполняется следующим образом. Для входа в цикл Д-оператор $(\varnothing)Y^1(f)$ устанавливает исходное истинное значение условия f. В цикле выполняется Д-оператор $(D_1)O_1(D_1')$, и при истинном значении условия l операция *p*-дизъюнкции выполняет Д-оператор $(D_2)O_2(D_2')$. Этот процесс продолжается до тех пор, пока *l*=1. Когда l примет значение ложь, Д-оператор $(D_2)O_2(D_2')$ не выполнится, а будет установлено $f = 0$. В результате чего, выполнение цикла завершится. Таким образом, цикл выполняется до тех пор, пока логическое условие, проверяемое в середине цикла, имеет истинное значение.

Из рассмотренного случая видно, что фиксирующие условия являются удобным средством для построения новых алгоритмических конструкций. Продемонстрируем эту возможность на примерах операторов, характерных для многих языков программирования.

Начнем с оператора *continue,* полагая, что нам необходимо реализовать следующую алгоритмическую конструкцию:

$$[(D)P(l)]\{(D_1)O_1(D_1')*$$
$$*[(D_1')P(l_1)]((D_2)O_2(D_2')*continue))*$$
$$*(D_3)O_3(D_3')\},$$

в которой, при истинном значении логического условия l_1, выполняется Д-оператор $(D_2)O_2(D_2')$, а $(D_3)O_3(D_3')$ не выполняется, так как цикл переходит к следующей итерации в результате выполнения оператора continue.

Поскольку оператор continue не является операцией САА\Д, воспользуемся фиксирующим логическим условием для построения конструкции, которая решает поставленную задачу.

$$[(D)P(l)]\{(D_1)O_1(D_1')*$$
$$*[(D_1')P(l^2)]((D_2)O_2(D_2')*(\varnothing)Y^0(f)) \vee$$
$$\vee (\varnothing)Y^1(f))*[f](D_3)O_3(D_3')\},$$

где при истинном значении логического условия l^2 фиксирующее условие f получает ложное значение, что и позволяет перейти к следующей итерации, пропуская Д-оператор $(D_3)O_3(D_3')$ с помощью операции р-фильтрации. При ложном значении l^2 фиксирующее условие f становится истинным, в результате чего Д-оператор $(D_3)O_3(D_3')$ выполняется с помощью той же операции.

Рассмотрим оператор *break*, необходимость использования которого подвергается сомнению, но, тем не менее, он присутствует во всех наиболее распространенных языках программирования. Возьмем аналогичную задачу

$$[(D)P(l^2)]\{(D_1)O_1(D_1')*$$
$$*[(D_1')P(l)]((D_2)O_2(D_2')*break)*$$
$$*(D_3)O_3(D_3')\},$$

в которой, после выполнения Д-оператора $(D_2)O_2(D_2')$, выполнение цикла прерывается.

Снова воспользуемся фиксирующими логическими условиями для построения конструкции, решающей поставленную задачу

$$(\varnothing)Y^1(f)*[(D)P(l)\wedge f]\{(D_1)O_1(D_1')*$$
$$*[(D_1')P(l_1)]((D_2)O_2(D_2')*(\varnothing)Y^0(f))*$$
$$*[f](D_3)O_3(D_3')\},$$

в которой фиксирующее условие f устанавливается в исходное истинное состояние для входа в цикл. Цикл выполняется по условию $l\wedge f$ при $f=1$, если условие l_1 не принимает истинного значения. Когда $l_1=1$, в результате выполнения р-дизъюнкции выполняется Д-оператор $(D_2)O_2(D_2')$, и фиксирующее условие f принимает ложное значение. В результате чего, Д-оператор $(D_3)O_3(D_3')$ не выполняется, а процесс циклирования прерывается по условию $l\wedge f$ при $f=0$.

При выполнении композиции предикатов

$$(D_1^o)P_1(l_1^k)\circ(D_2^o)P_2(l_2^k)\circ\ldots\circ(D_n^o)P_n(l_n^k),$$

где $\circ$ – одна из логических операций, формируется логическое условие $l_R^k=l_1^k\circ l_2^k\circ\ldots\circ l_n^k$ как результат применения логических операций ко всем условиям, продуцируемым предикатами.

Учитывая это, операции, например, p_2-дизъюнкции и p-итерации записываются в виде

$$[(D_1^o)P_1(l_1^2)\circ\ldots\circ(D_n^o)P_n(l_n^2)]((D_1)O_1(D_1')\vee$$
$$\vee(D_2)O_2(D_2')),$$
$$[(D_1^o)P_1(l_1^i)\circ\ldots\circ(D_n^o)P_n(l_n^i)]\{(D)O(D')\}$$

и очевидно, что все рассмотренные операции САА\Д могут выполняться с учетом логических условий неограниченной сложности.

Приведенные производные операции демонстрируют возможности создавать в рамках САА\Д операции, удобные для различных приложений. При этом показана возможность, не только строить такие конструкции, но и строить их с учетом эффективности реализации на целевом языке программирования.

Эквивалентные преобразования операций САА\Д и некоторых алгоритмических конструкций

Рассмотрение этого раздела начнем с определения понятия эквивалентности алгоритмических конструкций. При этом под алгоритмической конструкцией будем понимать любые допустимые сочетания Д-операторов и операций САА\Д.

Определение 6. Две любые алгоритмические конструкции, в частности Д-операторы и операции САА\Д, эквивалентны, если при равенстве исходных для них состояний вычислительного процесса $D_i^T=D_j^T$ в результате их выполнения будут получены состояния вычислительного процесса D_k^T и D_m^T, такие, что $D_k^P=D_m^P$.

Рассмотрим возможность разложения операции p_2-дизъюнкции на последовательные фильтры. Учитывая это, будем утверждать следующее.

Утверждение 1. Операция p_2-дизъюнкции

$$[(D^o)P(l^2)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2'))$$

может быть разложена на последовательные фильтры, т. е. представлена в виде эквивалентной ей композиции таких фильтров, при условии выполнения следующих ограничений $D^o\cap D_2'=\varnothing$ или $D^o\cap D_1'=\varnothing$.

Доказательство. Для обеспечения эквивалентности в выражении

$$\begin{aligned}[(D^o)P(l^2)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2'))=\\ =[(D^o)P(l)](D_1)O_1(D_1')*\\ *[(D^o)P(\bar{l})](D_2)O_2(D_2')\end{aligned}$$

необходимо обеспечить только выполнение одного и того же Д-оператора в левой и правой его частях, так как предикат на состояние D^P не влияет.

При $l^2=1$ и $l=1$, в обеих частях соотношения выполняется Д-оператор $(D_1)O_1(D_1')$, однако, в правой части состояние множества данных D^o проверяется дважды. В случае $D^o\cap D_1'\neq\varnothing$, это приведет к изменению состояния множества данных D^o. В результате чего, может быть получено $\bar{l}=1$, что приведет к выполнению $(D_2)O_2(D_2')$ и нарушению эквивалентности.

Специфика рассматриваемой операции такова, что при наличии указанного ограничения разложение может быть выполнено

$$\begin{aligned}[(D^o)P(l^2)]((D_1)O_1(D_1')\vee(D_2)O_2(D_2'))=\\ =[(D^o)P(\bar{l})](D_2)O_2(D_2')*\\ *[(D^o)P(l)](D_1)O_1(D_1'),\end{aligned}$$

однако аналогичные рассуждения приведут к нарушению эквивалентности при выполнении соотношения $D^o\cap D_2'\neq\varnothing$.

Таким образом, эквивалентность разложения p_2-дизъюнкции в первом случае обеспечивается ограничением $D^o\cap D_1'=\varnothing$, а во втором – $D^o\cap D_2'=\varnothing$.

Утверждение доказано.

Исходя из утверждения, можно сделать следующие выводы.

В том случае, когда на условие l влияет один из двух Д-операторов, указанное ограничение можно снять за счет выбора порядка следования этих Д-операторов.

Отметим, что аналогично могут быть доказаны случаи разложения p-дизъюнкций с большей значностью логических условий. Однако, имеющие место ограничения делают разложение p-дизъюнкций с большой значностью логических условий нецелесообразным.

Из доказанного утверждения следует, что в случае фиксирующих логических условий, которые по определению не зависят от выполнения Д-операторов, на преобразование операции p_2-дизъюнкции

$$\begin{aligned}[f^2]((D_1)O_1(D_1')\vee(D_2)O_2(D_2'))=\\ =[f](D_1)O_1(D_1')*[\bar{f}](D_2)O_2(D_2')\end{aligned}$$

никаких ограничений не накладывается.

Теперь покажем возможности оптимизирующих преобразований вложенных p-дизъюнкций, при выполнении приведенных в утверждении 1 ограничений, и p-итераций.

Рассмотрим следующие алгоритмические конструкции:

$$\begin{aligned}[(D^o)P(l^2)]([(D^o)P(l^2)]((D_1)O_1(D_1')\vee\\ \vee(D_2)O_2(D_2'))\vee(D_3)O_3(D_3')).\end{aligned}$$

Разложим p-дизъюнкцию на последовательные фильтры

$$\begin{aligned}[(D^o)P(l)]([(D^o)P(l)](D_1)O_1(D_1'))*\\ *[(D^o)P(\mathrm{l})]([(D^o)P(\bar{\mathrm{l}})](D_2)O_2(D_2'))*\\ *[(D^o)P(\bar{\mathrm{l}})]((D_3)O_3(D_3')),\end{aligned}$$

а полученное выражение преобразуем в соответствии с соотношениями (2), (3). В результате получаем выражение

$$\begin{aligned}[(D^o)P(l)]((D_1)O_1(D_1')*Z))*\\ *[(D^o)P(\bar{l})](D_3)O_3(D_3')),\end{aligned}$$

исключив из которого, в соответствии с (1), тождественный Д-оператор, и свернув фильтры в *p*-дизъюнкцию получим следующий оптимизированный результат $[(D^o)P(l)]((D_1)O_1(D_1')\vee(D_3)O_3(D_3'))$.

$$[(D_1^o)P_1(l^2)]([(D_2^o)P_2(l_1^{\,2})]((D_1)O_1(D_1')\vee \vee(D_2)O_2(D_2'))\vee(D_2)O_2(D_2')). \quad (5)$$

$$[(D_1^o)P_1(l^2)]((D_1)O_1(D_1')\vee \vee[(D_2^o)P_2(l_1^{\,2})]((D_1)O_1(D_1')\vee \vee(D_2)O_2(D_2'))). \quad (6)$$

Будем утверждать, что выражение (5) преобразуется к виду

$$[(D_1^o)P_1(l)\wedge(D_2^o)P_2(l_1)]((D_1)O_1(D_1')\vee \vee(D_2)O_2(D_2')),$$

а выражение (6) – к виду

$$[(D_1^o)P_1(l^2)\vee(D_2^o)P_2(l_1^{\,2})]((D_1)O_1(D_1')\vee \vee(D_2)O_2(D_2')).$$

Доказательство в данных случаях очевидно, так как эквивалентность этих преобразований определяется следующими фактами. В соотношении 2 Д-оператор $(D_1)O_1(D_1')$ выполняется только при истинности условий l^2 и $l_1^{\,2}$, а Д-оператор $(D_2)O_2(D_2')$ во всех остальных случаях. В соотношении 3 Д-оператор $(D_2)O_2(D_2')$ выполняется только при ложных значениях условий l^2 и $l_1^{\,2}$, а Д-оператор $(D_1)O_1(D_1')$ во всех остальных случаях.

$$[(D_1^o)P_1(l)]\{[(D_2^o)P_2(l_1)]\{(D_1)O_1(D_1')\}\}.$$

Преобразуется к виду

$$(D_1^o)P_1(l)([(D_2^o)\mathrm{P}_2(l_1)]\{(D_1)O_1(D_1')\}).$$

Из полученного соотношения следует, что предикат $(D_1^o)P_1(l)$ выполняется однократно. Докажем справедливость этого от противного полагая, что по завершению вложенного цикла условие l истинно и применяя последовательно соотношения (4), (2), (3) получаем:

$$(D_1^o)P_1(l)([(D_2^o)P_2(l_1)]\{(D_1)O_1(D_1')\}) =$$
$$= (D_1^o)P_1(l)([(D_2^o)P_2(l_1)]\{(D_1)O_1(D_1')\}\bar{l}_1) =$$
$$= (D_1^o)P_1(l)([0]\{(D_1)O_1(D_1')\}) =$$
$$(D_1^o)P_1(l)(Z) = N.$$

Поскольку при сделанном предположении получили неопределенный Д-оператор, это предположение неверно и преобразование эквивалентно.

$$[(D^o)P(l^2)]([(D_1^o)P_1(l_1)]\{(D_1)O_1(D_1')\}\vee \vee[(D_1^o)P_1(l_1)]\{(D_3)O_3(D_3')\}).$$

преобразуется к виду

$$[(D^o)P(l)]([(D_1^o)P_1(l_1)]\{(D_1)O_1(D_1')\})* *[(D_1^o)P_1(l_1)]\{(D_3)O_3(D_3')\}.$$

В данном случае доказательством служит тождественное соотношение (2), в соответствии с которым, при $(D^o)P(l^2)=1$, после выполнения в цикле Д-оператора $(D_1)O_1(D_1')$ будет получено такое состояние D_1^o, что $(D_1^o)P_1(l_1)=0$. В результате – $[(D_1^o)P_1(l_1)]\{(D_3)O_3(D_3')\}=Z$. В случае, кода $(D^o)P(l^2)=0$ будет выполняться цикл с телом $(D_3)O_3(D_3')$, а с телом $(D_2)O_2(D_2')$ будет пропущен.

Данный раздел иллюстрирует возможности преобразования за счет использования соотношений, характеризующих общие свойства операций и алгоритмических конструкций.

Заключение

Приведенные производные операции демонстрируют возможности создавать в рамках САА\Д удобные для различных приложений операции. При этом показана возможность, не только строить такие операции, но и строить их с учетом эффективности реализации на целевом языке программирования. Эта возможность, в частности с привлечением фиксирующих логических условий, представляется практически неисчерпаемой.

Возможности оптимизирующих пре образований, показанные в работе, представляющие самостоятельный интерес, могут быть существенно расширены за счет привлечения тождественных соотношений, характерных для различных предметных областей.

Обобщение полученных результатов и их апробация на конкретных примерах является перспективным направлением дальнейших исследований.

1. *Акуловский В.Г.* Основы алгебры алгоритмов, базирующейся на данных // Проблеми програмування. – 2010. – № 2-3. – С. 89–96.
2. *Акуловский В.Г.* Некоторые аспекты формализации данных и декомпозиция Д-операторов // Проблеми програмування. – 2009. – № 4 – С. 3–10.
3. *Дорошенко А.Е., Акуловский В.Г.* Алгебра алгоритмов с данными и прогнозирование вычислительного процесса // Проблеми програмування. – 2011. – № 3. – С. 3–10.
4. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – Киев.: Наук. думка, 1978. – 319 с.
5. *Яблонский С.В.* Введение в дискретную математику: учебн. Пособие для вузов. – М.: Высш.шк., 2002. – 384 с.
6. *Акуловский В.Г.* Расширенная алгебра алгоритмов // Проблеми програмування. – 2007. – № 3. – С. 3–15.
7. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е. и др.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев.: Академперіодика, 2007.– 634 с.

Получено 11.07.2011

Об авторах:

Дорошенко Анатолий Ефимович,
доктор физико-математических наук,
профессор.

*Акуловский Валерий Григорьевич***,**
кандидат технических наук, доцент.

Место работы авторов:

Институт программных систем
НАН Украины, 03187, Киев-187,
проспект Академика Глушкова, 40
Тел.: (044) 526 3559.
e-mail: dor@isofts.kiev.ua

Академия таможенной службы Украины.
49000, г. Днепропетровск,
ул. Дзержинского 2\4.
Тел/Факс: (056) 745 55 96,
e-mail: academy@amsu.dp.ua.