МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ОДЕССКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ И. И. МЕЧНИКОВА

Институт математики, экономики и механики
Кафедра компьютерной алгебры и дискретной математики

С.В.ФЕДОРОВСКИЙ

АЛГЕБРАИЧЕСКИЕ СИСТЕМЫ В АЛГОРИТМИКЕ

Методические указания для студентов 3 курса направления подготовки
050102 «компьютерная инженерия»

ОДЕССА – 2013

АЛГЕБРАИЧЕСКИЕ СИСТЕМЫ В АЛГОРИТМИКЕ.
Методические указания для студентов 3 курса направления подготовки 050102 «компьютерная инженерия». – 65с.

Составители: **Федоровский С.В**., к.ф.-м.н., доцент кафедры компьютерной алгебры и дискретной математики ИМЭМ

Рецензенты: **Евтухов В.М.** д.ф.-м.н., профессор кафедры дифференциальных уравнений ИМЭМ

Кореновский А.А., д.ф.-м.н., профессор кафедры математического анализа ИМЭМ

Рекомендовано к печати
Ученым советом ИМЭМ Одесского национального университета имени И. И. Мечникова
протокол № 1 от 19 сентября 2013 г.
.

СОДЕРЖАНИЕ

Введение.

Настоящие методические указания предназначены для студентов третьего курса направления подготовки 6.050102 «компьютерная инженерия» ИМЭМ. Методические указания имеют главной целью ознакомление студентов с основными алгебраическими системами в алгоритмике – алгебрами Калужнина, Янова, Дэйкстры, Глушкова.

В настоящее время при компьютерной реализации конкретных алгоритмов программисты очень часто, как вспомогательное средство, используют блок-схемы алгоритмов – очень наглядное и эффективное средство. Однако для своего изображения блок-схемы требуют большой площади и потому, для громоздких алгоритмов, приходится строить их по частям и, тем более, подобную работу, в свою очередь, невозможно поручить той же ЭВМ.

В алгебрах Янова, Дэйкстры и Глушкова, схемы алгоритмов изображаются формулами в привычном понимании, напоминающими формулы обычной математики (высшей алгебры, математического анализа и т.д.) и, следовательно, схемы алгоритмов и сами алгоритмы представляются в компактной форме. Такое представление алгоритмов и их схем позволяет строить более сложные алгоритмы из более простых с помощью операции суперпозиции формул. Подобную работу можно поручить ЭВМ, что весьма плодотворно используется в настоящее время в макропрограммировании.

Кроме того, знакомство студентов с алгеброй Калужнина позволит им взглянуть на блок-схемы алгоритмов с алгебраической точки зрения и, следовательно, использовать их с большей эффективностью.

Настоящие методические указания будут полезны также студентам направления подготовки 6.040301 «прикладная математика» ИМЭМ, изучающим курс математической логики и теории алгоритмов.

1. Алгебраические системы.

1.1 Универсальные алгебры.

Определение 1.1.1 ***Универсальной алгеброй*** *(УА) называется упорядоченная пара (кортеж, набор) двух объектов* $\tilde{A} = \langle A; \Sigma_0 \rangle$, *где* ***A*** *– произвольное непустое множество (****основа УА****) и* Σ_0 *- множество* ***алгебраических*** *операций на множестве* ***A*** *(в общем случае различной арности (местности)) (****сигнатура УА****).*

Понятие универсальной алгебры, может быть, и не употреблялось ранее (на младших курсах обучения), но не является по сути своей новым, поскольку изучавшиеся, например, в курсе алгебры ***группоиды, полугруппы, моноиды, группы, кольца, поля*** являются примерами универсальных алгебр. При этом группоид, полугруппа, моноид, группа $\tilde{G} = \langle G; * \rangle$ - УА, сигнатура которой содержит одну бинарную алгебраическую операцию; кольцо, поле $\tilde{R} = \langle R; \{+, \bullet\} \rangle$ - УА, сигнатура которой содержит две бинарные алгебраические операции; булева алгебра $\tilde{B} = \langle B; \{\neg, \vee, \wedge\} \rangle$ (где $B = \{0,1\}$) – УА, сигнатура которой содержит одну унарную (одноместную) алгебраическую операцию «отрицание» и две бинарные (двухместные) алгебраические операции – дизъюнкцию и конъюнкцию.

Таким образом, мультипликативная группа (по умножению) $\mathbb{C}_n$ корней ***n***-ой степени из единицы, аддитивная группа (по сложению) $\mathbb{Z}_m$ классов вычетов по***mod m***, кольцо квадратных матриц над некоторым полем, кольцо многочленов над заданным полем, поля $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ - все это примеры универсальных алгебр.

1.2 Метаалгебры.

Нам в дальнейшем придется знакомиться с метаалгебрами в алгоритмике. Для того чтобы лучше понять идею перехода от алгебры к

ее метаалгебре, рассмотрим построение метаалгебры (высшего уровня формализации конкретной алгебры) на примере алгебры Буля (или булевой алгебры) $\tilde{B} = \langle B; \{\neg, \vee, \wedge\} \rangle$.

Как уже отмечалось выше, булева алгебра – это универсальная алгебра $\tilde{B} = \langle B; \{\neg, \vee, \wedge\} \rangle$, в которой основой алгебры является множество B = {0,1}. В математической логике константа 0 понимается как произвольное ложное высказывание, а константа 1 – как произвольное истинное высказывание. Математическая логика игнорирует содержательный смысл высказываний, а принимает во внимание только тот факт, как из элементарных (простых) высказываний строятся более сложные составные высказывания с помощью операций над высказываниями. Таким образом, булева алгебра в силу своего определения своей основой имеет множество всевозможных высказываний, представленных логическими (пропозициональными) переменными, а сигнатура содержит операции над высказываниями (переменными), как элементами основы алгебры.

Далее, в процессе развития этой алгебры вводится понятие формулы алгебры и операции сигнатуры обобщаются до операций над формулами. Такой переход от операций над элементами множества к операциям над формулами является общим для всей математики. Вспомним, например, операции над формулами, определяющими функции в обычной алгебре или математическом анализе. Если некоторые функции задаются формулами ***f*** и ***g***, то сумма этих функций ***h*** задается формулой ***h=f+g***, где сумма формул понимается в следующем смысле ***h(x)=(f+g)(x)=f(x)+g(x)***, т.е. определяется с помощью сложения элементов множества, на котором определены исходные функции. На этом этапе формализации (развития) исходной алгебры появляется некоторая неопределенность, а именно операции сигнатуры можно рассматривать и как операции над формулами основы, и, с дру-

гой стороны, как формулы, входящие в основу алгебры. Например, рассматривая булеву алгебру (алгебру логики) с уже введенным понятием формулы (множество всевозможных формул этой алгебры обозначается P_2) и учитывая операции над формулами этой алгебры, мы получаем новую алгебру $\tilde{B} = \langle P_2; \{\neg, \vee, \wedge\} \rangle$, которая по-прежнему называется алгеброй логики (булевой алгеброй) и в которой, например, выражение $x \vee y$ можно с одной стороны понимать как некоторую формулу основы P_2, и как результат применения операции дизъюнкции к элементам основы x и y с другой стороны.

На следующем этапе формализации (развития) исходной алгебры операции сигнатуры предыдущей алгебры заменяются одной единственной операцией – ***операцией суперпозиции*** формул изучаемой алгебры. Напомним определение операции суперпозиции функций (формул).

Определение 1.2.1 *Пусть нам задана некоторая универсальная алгебра* $\tilde{A} = \langle A; \Sigma_0 \rangle$ *и следующий набор функций (формул)* $f^{(n)}, g_1^{(m)}, g_2^{(m)}, ..., g_n^{(m)}$, *заданных на основе алгебры* A *(здесь нижний индекс указывает порядковый номер функционального символа, а верхний – количество переменных, от которых зависит соответствующая функция). Тогда говорят, что функция* $F^{(m)}$ *получена с помощью операции суперпозиции из исходных функций* $f^{(n)}, g_1^{(m)}, g_2^{(m)}, ..., g_n^{(m)}$ *в указанном порядке, если*

$$F^{(m)} = S^{(n+1)}(f^{(n)}, g_1^{(m)}, g_2^{(m)}, ..., g_n^{(m)}) = f^{(n)}(g_1^{(m)}, g_2^{(m)}, ..., g_n^{(m)}),$$

где $S^{(n+1)}$ *– оператор суперпозиции.*

Таким образом, если $f^{(n)} = f(x_1, x_2, ..., x_n)$, $g_1^{(m)} = g_1(y_1, y_2, ..., y_m)$,

$$g_2^{(m)} = g_2(y_1, y_2, ..., y_m), ..., \; g_n^m = g_n(y_1, y_2, ..., y_m),$$

то $F^{(m)} = F(y_1, y_2, ..., y_m) = f(g_1(y_1, y_2, ..., y_m), g_2(y_1, y_2, ..., y_m), ..., g_n(y_1, y_2, ..., y_m))$.

Иными словами, операция суперпозиции функций (формул) состоит в подстановке одних функций вместо переменных в другие функции.

Этот момент для нас важен, ибо в дальнейшем, при рассмотрении алгебраических систем в алгоритмике, мы будем пользоваться операцией суперпозиции именно в отмеченном выше смысле.

Определение 1.2.2 *Универсальная алгебра, полученная на втором этапе формализации из начальной (с единственной операцией суперпозиции) по отношению к исходной называется* ***метаалгеброй****.*

Например, $\tilde{B} = \langle P_2; СУПЕР \rangle$ является метаалгеброй для булевой алгебры $\tilde{B} = \langle B; \{\neg, \vee, \wedge\} \rangle$. Очень часто название «метаалгебра» при использовании метаалгебры не употребляется, а применяется исходное название универсальной алгебры. Так, например, булева алгебра на среднем этапе формализации применялась при изучении нормальных форм булевой алгебры, а ее метаалгебра – при изучении полных систем булевых функций, при этом специально не подчеркивалось на каком уровне формализации эта алгебра использовалась.

1.3 Модели.

Для введения понятия модели нам понадобятся некоторые вспомогательные понятия.

Определение 1.3.1 *Пусть нам задано некоторое непустое множество* ***D*** *и некоторая функция* $P: D^n \to B$*, где* $B = \{0,1\}$ *– булево множество. Тогда эта функция* ***P*** *называется* ***n - местным предикатом*** *в области* ***D****.*

Поскольку элементы булевого множества B можно интерпретировать в логическом смысле (1 – истина или истинное высказывание и 0 – ложь или ложное высказывание), то определение предиката можно сформулировать иначе.

Определение 1.3.2 ***n - местным предикатом*** *в области* ***D****, где* ***D*** *– произвольное непустое множество, называется всякая* ***n –*** *местная функция, определенная на множестве* ***D****, значениями которой служат высказывания.*

Примеры предикатов.

1) Рассмотрим произвольное уравнение, например, от двух переменных в поле $\mathbb{R}$ действительных чисел и посмотрим на него, как на некоторую функцию от двух переменных $P(x,y)$ над $\mathbb{R}$, т.е. $P(x,y):=(2x-3y^2+5=0)$. Тогда, например, $P(1,-3)=(2\cdot 1-3\cdot(-3)^2+5=0)=(-20=0)$, т.е. значением функции $P(x,y)$ на наборе значений переменных $(1,-3)$ является ложное высказывание $(-20=0)$, или $P(1,-3)=0$. Аналогично рассмотренному, получим $P(-1,1)=1$ (истинное высказывание $(0=0)$). И так далее. Таким образом, уравнение $2x-3y^2+5=0$ определяет над $\mathbb{R}$ двухместный предикат $P(x,y)$.

2) Рассмотрим произвольное неравенство, например, от трех переменных в кольце целых чисел $\mathbb{Z}$, а именно $Q(x,y,z):=(x+2y-z^3>0)$. Аналогично предыдущему примеру имеем: $Q(1,1,1)=1$ (истинное высказывание), $Q(-1,-1,1)=0$ (ложное высказывание) и т.д. Таким образом, неравенство $x+2y-z^3>0$ определяет над $\mathbb{Z}$ трехместный предикат $Q(x,y,z)$.

3) Рассмотрим на множестве натуральных чисел $\mathbb{N}$ отношение $R(x):=(x:2)$ («***x*** делится на 2» или «2 является делителем ***x***»). Тогда, например, $R(3)=0$ (ложь), а $R(2)=1$ (истина) и т.д. Следовательно, отношение $x:2$ на множестве натуральных чисел $\mathbb{N}$ определяет одноместный предикат $R(x)$.

Таким образом, в математике мы очень часто имеем дело с предикатами соответствующей местности, заданными на некоторых числовых множествах – любое уравнение, любое неравенство с неизвестными, любое отношение на числовых множествах определяют некоторые предикаты на этих множествах.

Определение 1.3.3 ***Моделью*** *называется упорядоченная пара (кортеж, набор) двух объектов* $\tilde{M}=\langle M;\Sigma_\pi\rangle$, *где* ***M*** *– произвольное непустое*

*множество (**основа модели**) и Σ_π - **сигнатура модели**, т.е. множество предикатов на множестве M (в общем случае различной местности).*

Модели, например, используются при интерпретации формул логики предикатов, при формализации постановки задач.

1.4 Многоосновные алгебраические системы.

Рассмотренные ранее понятия универсальной алгебры и модели можно обобщить, рассматривая вместо одной основы в этих алгебраических системах произвольное конечное их количество.

Определение 1.4.1 ***Многоосновной универсальной алгеброй*** *(многоосновной УА) называется набор объектов* $\tilde{A} = \langle A_1, A_2, ..., A_n; \Sigma_0 \rangle$, *где* $A_1, A_2, ..., A_n$ *– произвольные непустые множества (**основы многоосновной УА**) и* Σ_0 *- множество **полиморфных** операций на указанных основах (в общем случае различной арности (местности)) - **сигнатура многоосновной УА.***

Отличие полиморфных операций от алгебраических состоит в том, что аргументы такой операции могут выбираться в нескольких основах, а результат (значение) может принадлежать любой из основ (может быть не имеющей никакого отношения к тем основам, откуда выбирались аргументы). С подобными универсальными алгебрами мы уже встречались в курсе линейной алгебры, рассматривая линейные пространства над некоторым полем. Вспомним определение линейного пространства.

Определение 1.4.2 *Непустое множество **L** элементов произвольной природы называется линейным пространством над полем **ℙ**, если:*

*1. На **L** определена «внутренняя» алгебраическая операция⊕ сложения элементов относительно которой **L** образует абелеву группу;*

*2. На **L** определена «внешняя» операция⊙ умножения элементов **L** на числа поля **ℙ**, относительно которой выполняются следующие требования:*

а) (∀x∈ L)(∀α∈ℙ)[α⊙x∈ L] (аналог «алгебраичности» операции);

б) (∀x∈ L))(∀α,β∈ℙ)[α⊙(β⊙x)=(αβ)⊙x=β⊙(α⊙x)] (аналог ассоциативного закона);

в) (∀x∈ L)[1⊙x=x], где 1 – единица поля ℙ (унитарный закон);

3. Операции ⊕ и⊙ связаны двумя дистрибутивными законами:

а) (∀x∈ L))(∀α,β∈ℙ)[(α+β)⊙x=α⊙x⊕β⊙x];

б) (∀x,y∈ L)(∀α∈ℙ)[α⊙(x⊕y)=(α⊙x)⊕(α⊙y)].

Здесь мы специально подчеркнули различия в начертании операций сложения чисел и сложения векторов (элементов ***L***), а также произведения чисел и умножения вектора на число, хотя в литературе их начертания не различаются (какая из операций используется подразумевается из контекста). Например, закон 3 б) имеет вид

$$(\forall x,y\in L)(\forall\alpha\in\mathbb{P})[\alpha(x+y)=\alpha x+\alpha y].$$

Исходя из определения линейного пространства (ЛП) ясно, что с формальной точки зрения ЛП является двухосновной универсальной алгеброй $\tilde{L}=\langle L,\mathbb{P};\{\oplus,\odot,+,\cdot\}\rangle$ в сигнатуре которой содержатся алгебраическая на ***L*** операция ⊕, алгебраические на ***ℙ*** операции + и ·, полиморфная на ***L*** и ***ℙ*** операция ⊙.

Аналогично предыдущему можно обобщить понятие модели.

Определение 1.4.3 ***Многоосновной моделью*** *называется набор объектов $\tilde{M}=\langle M_1,M_2,...,M_n;\Sigma_\pi\rangle$, где $M_1,M_2,...,M_n$ – произвольные непустые множества (**основы модели**) и Σ_π - множество полиморфных предикатов на основах (в общем случае различной местности) - **сигнатура многоосновной модели**.*

С полиморфными предикатами мы уже встречались на младших

курсах, хотя такого названия и не употребляли. Например, в определении предела последовательности

$$\lim_{n\to+\infty} x_n = A \overset{def}{\Leftrightarrow} (\underset{\varepsilon\in R\wedge\varepsilon>0}{\forall\varepsilon})(\underset{M\in N\wedge M=M(\varepsilon)}{\exists M})(\underset{n\in N\wedge n\geq M}{\forall n})\left[|x_n - A| < \varepsilon\right]$$

(здесь R – поле действительных чисел, а N – множество натуральных чисел), используется полиморфный предикат $[|x_n - A| < \varepsilon]$, т.к. его аргументы ***n*** и ***ε*** принадлежат разным множествам.

Определение 1.4.4 ***Многоосновной алгебраической системой*** *(МАС) называется набор объектов* $\tilde{A} = \langle A_1, A_2, ..., A_n; \Sigma_\pi \cup \Sigma_\pi \rangle$, *где* $A_1, A_2, ..., A_n$ *– произвольные непустые множества (**основы МАС**),* Σ_0 *- множество полиморфных операций на основах, а* Σ_π *- множество полиморфных предикатов на основах (в общем случае различной местности) (**сигнатура МАС**).*

Как видим, сигнатура многоосновной алгебраической системы содержит как операции на основах, так и предикаты (в общем случае полиморфные). Таким образом, МАС – наиболее широкое понятие алгебраической системы такое, что все остальные – это частные случаи МАС. Например, многоосновная универсальная алгебра – это МАС, сигнатура предикатов которой пустая, а универсальная алгебра – МАС с одной основой и пустой сигнатурой предикатов.

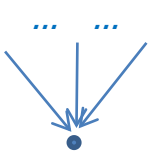
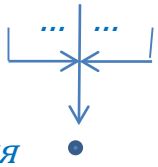
2. Алгебра Калужнина (АК).

2.1 Граф-схемы (ГС).

Определение 2.1.1. *Обыкновенный (без кратных ребер и петель) ориентированный граф (орграф) G = (V,E) называется операторной граф-схемой, если он обладает следующими особенностями:*

1. Соотнесенный с ним неориентированный граф является связным.

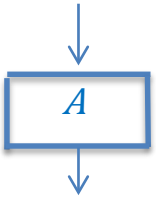
2. При наличии нескольких входных ребер у некоторой вершины одно из этих ребер объявляется главным (объявляется входным ребром этой вершины), а остальные служат для «передачи управления» на это ребро, т.е.

вместо  *применяется конструкция* 

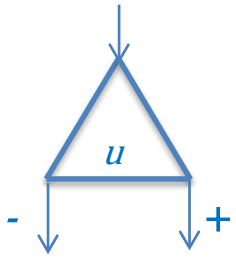
Таким образом, каждая вершина графа G *может иметь не более одного входного ребра.*

3. Множество вершин V *графа* G *имеет две специфические вершины «Вход» и «Выход». Вершина «Вход» имеет только одно выходное ребро и не имеет входных ребер. Вершина «Выход», наоборот, не имеет выходных ребер и имеет только одно входное ребро. При геометрическом задании графа* G *эти вершины изображаются кружочками или овалами, внутри которых стоит название вершины «Вход» или «Выход». Множество остальных вершин графа разбито на два непересекающиеся подмножества – подмножество «операторных» и подмножество «условных» (логических) вершин (подмножество «условных» вершин может быть пустым):*

а) «Операторные» вершины характерны тем, что каждая из них имеет одно входное и одно выходное ребра. Такие вершины при геометрическом задании графа G *изображаются прямоугольниками, внутри которых расположены обозначения вершин. Операторные вершины обозначаются большими буквами начала латинского алфавита или такими же индексированными переменными. Например,*



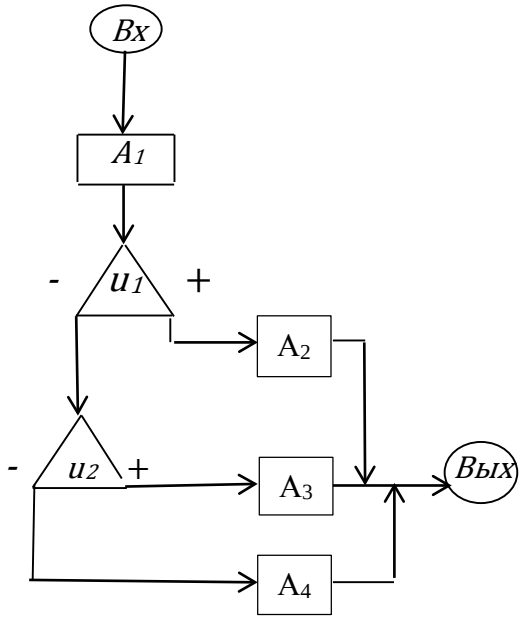
б) «Условные» вершины характерны тем, что каждая из них имеет одно входное и два выходных ребра (слово «условная» здесь не означает «мнимая» или «несуществующая», а понимается в смысле «реализующая некоторое «условие» или «предикат»). Эти ребра снабжены метками «+» и « - ». При геометрическом задании графа ***G*** *«условные» вершины изображаются равнобедренными или равносторонними треугольниками, стоящими на основании. Внутри треугольников расположены обозначения вершин. «Условные» вершины обозначаются маленькими буквами конца латинского алфавита x, y, z, u, v или такими же индексированными переменными (символами пропозициональных логических переменных). Например,*



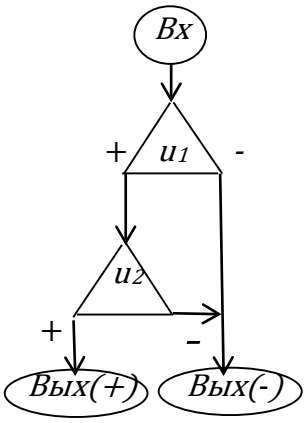
Замечание 2.1.1. *Неориентированный граф, соотнесенный с орграфом* ***G****, получается из* ***G****, если у всех ребер графа* ***G*** *устранить направления и затем в полученном неориентированном графе устранить лишние вхождения кратных ребер, т.е. вместо двух ребер вида (a, b) и (b, a) оставить одно из них.*

По аналогии с операторными граф-схемами можно рассматривать и логические ГС. Отличие последних от операторных состоит в том, что логическая ГС не имеет операторных вершин и имеет два выхода, снабженных метками «+» и « - » (положительный и отрицательный).

Пример 2.1.1 (операторной ГС) *:*

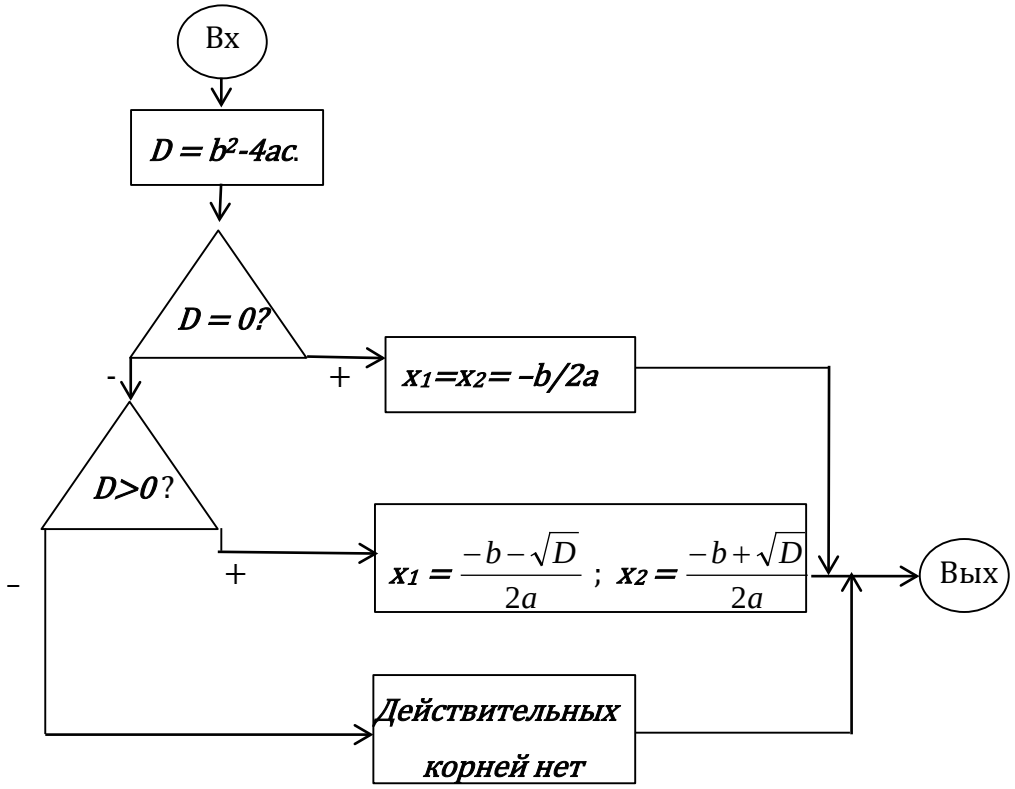


Пример 2.1.2 (логической ГС)*:*



Из определения ГС ясно, что ГС представляет собой абсолютно формальную конструкцию, не несущую в себе никакого содержательного смысла. Содержательный смысл ГС приобретает в ***интерпретациях***. Интерпретация ГС по сути ничем не отличается от интерпретации математических формул или (что ближе) формул логики предикатов и состоит в наполнении конкретным содержанием каждого символа ГС. Например, зададим следующую интерпретацию ГС примера 2.1.1. Рассмотрим произвольное квадратное уравнение $ax^2+bx+c=0$ над полем действительных чисел $\mathbb{R}$. В качестве оператора A_1 выберем оператор вычисления выражения $D = b^2-4ac$. В качестве оператора A_2 выберем оператор вычисления выражения $(-b/2a)$ и печати сообще-

ния $x_1=x_2=-b/2a$. В качестве оператора A_3 выберем оператор вычисления выражений $\frac{-b-\sqrt{D}}{2a}$ и $\frac{-b+\sqrt{D}}{2a}$ и печати сообщения $x_1 = \frac{-b-\sqrt{D}}{2a}$ и $x_2 = \frac{-b+\sqrt{D}}{2a}$. В качестве оператора A_4 выберем оператор печати сообщения «*действительных корней нет*». Вместо логической переменной u_1 выберем предикат $D = 0$, а вместо u_2 – предикат $D>0$. В заданной интерпретации исходная ГС представляет собой ни что иное, как алгоритм решения квадратного уравнения $ax^2+bx+c=0$ и имеет вид:



Таким образом, интерпретированные ГС (здесь и в общем случае) задают некоторые алгоритмы, а неинтерпретированные – *схемы алгоритмов*. Ясно также, что ГС представляют собой реализацию формализованного понятия блок-схем алгоритмов.

Уточним понятие интерпретации ГС.

Определение 2.1.2. *Под интерпретацией операторной ГС понимается задание пары объектов* $I = \langle \tilde{A}, \varphi \rangle$, *где* $\tilde{A} = \langle D, \Sigma_0 \cup \Sigma_\pi \rangle$ *- некоторая одноосновная алгебраическая система, сигнатура которой содержит как операции (операторы), так и предикаты;* φ *- интерпретирующая функция, которая каждой операторной переменной ГС сопоставляет конкретный оператор из* Σ_0 *и каждой логической переменной сопоставляет конкретный предикат из* Σ_π. *Под интерпретацией логической ГС понимается задание пары объектов ,* $I = \langle \tilde{M}, \varphi \rangle$, *где -* $\tilde{M} = \langle D, \Sigma_\pi \rangle$ *- некоторая модель;* φ *- интерпретирующая функция, которая каждой логической переменной сопоставляет конкретный предикат из* Σ_π.

Еще раз подчеркнем, что интерпретация конкретной операторной ГС приводит к некоторому конкретному алгоритму преобразования входных слов в некотором алфавите, а интерпретация логических ГС приводит к заданию сложных предикатов, построенных из более простых с помощью логических операций (как правило, операций булевой тройки – отрицания, конъюнкции и дизъюнкции).

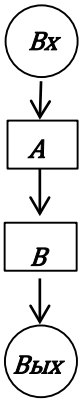
Процесс обработки (преобразования) слов в некотором алфавите («данных») алгоритмом, заданным с помощью интерпретированной ГС, следующий. Входное слово подается на вершину «Вход» и дальнейшее продвижение слов по ГС происходит в соответствии с направлением ребер ГС. При этом если некоторое слово $\boldsymbol{m_i}$ поступает по входному ребру в операторную вершину, в которой находится в результате интерпретации ГС некоторый конкретный оператор $\mathfrak{R}$, то по выходному ребру продолжает движение по ГС результат обработки $\boldsymbol{m_i}$ оператором $\mathfrak{R}$, т.е. $\mathfrak{R}(\boldsymbol{m_i})$, если же слово $\boldsymbol{m_i}$ поступает на вход «условной» вершины, в которой в силу интерпретации находится конкретный предикат $\mathfrak{U}$, то $\boldsymbol{m_i}$ проходит эту вершину, не претерпевая никаких изменений, но свой дальнейший путь это слово продолжает по

ребру с меткой «+», если значение предиката $\mathfrak{U}(m_i)$ равно *1* (истина), и по ребру с меткой « - », если значение предиката $\mathfrak{U}(m_i)$ равно *0* (ложь). Алгоритм прекращает свою работу, если на определенном такте некоторое слово m поступает на вход вершины «Выход». Это слово и является результатом обработки исходного входного слова алгоритмом, заданным с помощью интерпретированной ГС.

2.2 Основные программистские конструкции в терминах граф-схем.

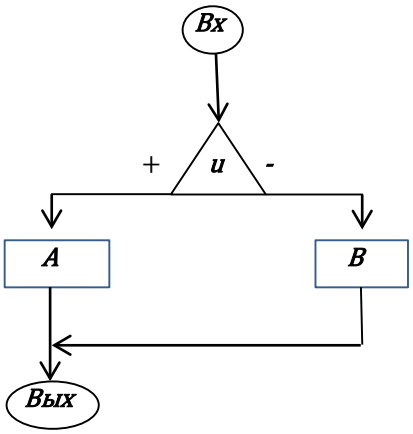
При написании конкретных программ, как правило, используются стандартные конструкции – композиции, альтернативы и циклы. Рассмотрим их представления с помощью неинтерпретированных граф-схем.

1. ***Композиция*** (операторов) $A*B$ - последовательное выполнение операторов – сначала A, затем B (т.е., в отличие от композиции отображений в алгебре или суперпозиции функций в анализе, выполняется в порядке не справа налево, а наоборот, т.е. в порядке следования). ГС композиции $A*B$ операторов A и B следующая:



2. ***Альтернатива*** подразумевает разветвление алгоритма на две ветви по некоторому условию. Скажем, пусть нам дано некоторое условие u и два оператора A и B. И пусть еще известно, что в случае истинности условия u выполняется оператор A, а если условие u ложное – вы-

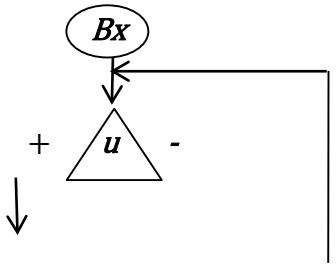
полняется оператор ***B***. Тогда ГС альтернативы, определяемой ***u, A, B*** следующая:

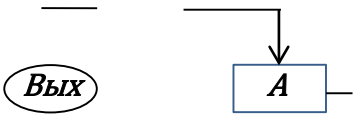


Условимся в дальнейшем положительную ветвь альтернативы считать главной, а отрицательную ветвь замыкать на положительную.

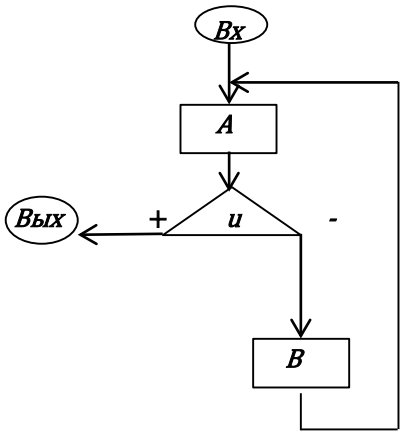
3. ***Циклы.*** Не будем останавливаться на описании понятия цикла (считаем, что представление о циклах к третьему году обучения имеется, например, циклы в графах). Напомним только, что существуют три разновидности циклов – циклы с предусловием, циклы с условием в середине, циклы с постусловием.

А) ***Цикл с условием в начале цикла*** характерен тем, что при его применении сначала проверяется условие цикла (некоторый предикат) ***u***, а затем, при ***отрицательном*** условии (ложном значении предиката) выполняется тело цикла (некоторый оператор) ***A***. В случае, когда условие цикла становится ***положительным*** (при истинном значении предиката) происходит выход из цикла. Неинтерпретированная ГС такого цикла следующая:

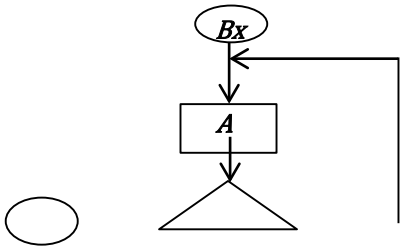




Б) ***Цикл с условием в середине цикла*** характерен тем, что тело этого цикла состоит из двух операторов A и B. При выполнении такого цикла сначала ко входному слову m_i применяется оператор A (предоператор), затем на полученном слове $A(m_i)$ проверяется условие цикла u (вычисляется значение предиката) и в случае его невыполнения (значение предиката 0) продолжается выполнение тела цикла – к слову $A(m_i)$ применяется оператор B (постоператор) после чего полученное слово $B(A(m_i))$ снова подается на вход цикла. Выход из цикла происходит в том случае, когда условие цикла выполняется (предикат примет значение 1 (истина)). Неинтерпретированная ГС такого цикла следующая:



В) ***Цикл с условием в конце цикла***. В таком цикле ко входному слову сначала применяется оператор тела цикла, а затем на полученном слове проверяется условие цикла. Если оно не выполняется, то полученное на этом этапе слово снова подается на вход цикла, если же выполняется – происходит выход из цикла. Неинтерпретированная ГС такого цикла следующая:



Если ввести в рассмотрение тождественный оператор E такой, что $\forall m_i : E(m_i)=m_i$, то ясно, что цикл с условием в середине цикла является наиболее общим. Цикл с условием в начале цикла получается из цикла с условием в середине в случае, когда предоператор тела цикла равен тождественному оператору. Цикл с условием в конце цикла получается из цикла с условием в середине цикла в случае, когда постоператор тела цикла равен тождественному оператору.

2.3 Классические алгоритмы сортировки массивов данных.

Пусть нам задано некоторое непустое множество A (алфавит), элементы которого (буквы) a_i будем называть «данными», а слова в этом алфавите (конечные последовательности букв) будем называть массивами данных. Пусть, кроме того, известно, что A – частично упорядоченное множество (ЧУМ) с линейным порядком «≤»(любые два элемента этого множества связаны отношением частичного порядка). Пусть еще нам заданы символы H,$У$, K не принадлежащие A, но в отношении которых известно, что $(\forall a \in A)[H < a < K]$. Символ $У$ не связан отношением «≤» с элементами множества A. Эти символы будем называть «маркерами» и использовать при записи массивов данных. При этом маркер H (маркер начала массива) располагается в самом начале массива (непосредственно перед первым элементом массива), маркер K (маркер конца массива) – в самом конце массива (непосредственно после последнего элемента массива) и маркер $У$ (указатель) может передвигаться по массиву и располагаться в любом месте массива (но обязательно после маркера начала массива и перед маркером конца). Массивы данных, записанные с помощью указанных выше маркеров, будем называть ***размеченными массивами данных*** (в дальнейшем просто массивами). Массив $M = HУa_1a_2...a_mK$ называется упорядоченным (отсортированным), если $\underset{i\in 1,m-1}{\forall} i : a_i \le a_{i+1}$.

Преобразование произвольного массива в массив упорядоченный можно выполнить с помощью многих алгоритмов сортировки массивов. Рассмотрим три классических алгоритма сортировки массивов и изображение этих алгоритмов с помощью интерпретированных ГС, а также неинтерпретированные схемы этих алгоритмов. Условимся, что на вход алгоритма сортировки массив поступает в виде $M=НУa_1a_2...a_mК$ (указатель расположен непосредственно за маркером начала массива).

2.3.1. Алгоритм «Пузырек». *На каждом такте работы алгоритма сравниваются между собой элементы, стоящие рядом с указателем (правее и левее указателя или соседние с указателем). Если между соседними элементами нарушения порядка нет, то указатель смещается вправо на один элемент массива и снова сравниваются соседние с указателем элементы массива и т.д. Если же указатель обнаруживает соседнюю пару элементов с нарушением порядка (левый элемент больше правого), то происходит перестановка этих элементов местами (транспонирование элементов), указатель смещается направо на один элемент, снова сравниваются соседние элементы и т.д. Если при движении слева направо по массиву указатель достигает маркера конца массива, то указатель сбрасывается (устанавливается) в начало массива и на этом заканчивается очередной шаг работы алгоритма. Далее процесс упорядочения массива продолжается (выполняется следующий шаг работы алгоритма). Алгоритм заканчивает свою работу, если массив становится упорядоченным.*

Из описания алгоритма становится понятным название алгоритма. На каждом шаге работы алгоритма (на одном прогоне указателя от начала до конца массива) обнаруживается (как бы «всплывает» как пузырек) наибольший из еще неупорядоченных элементов массива и

передвигается («плывет») направо по массиву до установки на свое место.

Формализуем алгоритм. Выделим необходимые для реализации алгоритма элементарные операторы и предикаты (задавая тем самым алгебраическую систему, сигнатура которой используется при интерпретации схемы алгоритма). Пусть ***MASS*** - множество размеченных массивов данных в заданном алфавите ***A***. Все действия по упорядочению массива имеют смысл только в том случае, если массив неотсортированный. Поэтому необходим предикат, проверяющий упорядоченность массива. Пусть ***U(M)*** («упорядоченный массив?») - такой предикат на ***MASS***:

$$U(M)=\begin{cases}1, \textit{если } M-\textit{упорядоченный};\\ 0, \textit{если } M \textit{ неупорядоченный}.\end{cases}$$

Далее, при перемещении указателя по массиву слева направо, сравниваются между собой соседние с указателем элементы массива. Для такой проверки упорядоченности соседних с указателем элементов необходим предикат $(l>r,У)(M)$ («левый больше правого относительно указателя?»):

$$(l>r,У)(M)=\begin{cases}1, \textit{если } M=Ha_1a_2...a_iУa_{i+1}...a_mK\wedge(a_i>a_{i+1});\\ 0, \textit{если } M=Ha_1a_2...a_iУa_{i+1}...a_mK\wedge(a_i\le a_{i+1}).\end{cases}$$

Наконец, при достижении указателем конца массива, он должен устанавливаться в начало массива. Таким образом, необходима проверка того, достиг ли указатель конца массива или нет. Пусть такую проверку выполняет предикат ***d(У,K) (M)*** («достиг указатель конца массива?»):

$$d(У,K)(M)=\begin{cases}1, \quad \textit{если } M=Ha_1a_2...a_ia_{i+1}...a_mУK;\\ 0, \textit{если } \exists i: M=Ha_1a_2...a_iУa_{i+1}...a_mK.\end{cases}$$

Выделим теперь элементарные операторы, необходимые для реализации алгоритма. Поскольку при достижении указателем конца массива указатель должен устанавливаться в начало массива, то такое

действие – это некоторое преобразование размеченного массива (изменение его записи). Пусть это преобразование выполняет оператор ***Уст(У,Н)(М)***, такой, что:

если $M = Ha_1a_2...a_iYa_{i+1}...a_mK$, то $Уст(У,Н)(М) = НУa_1a_2...a_ia_{i+1}...a_mK$.

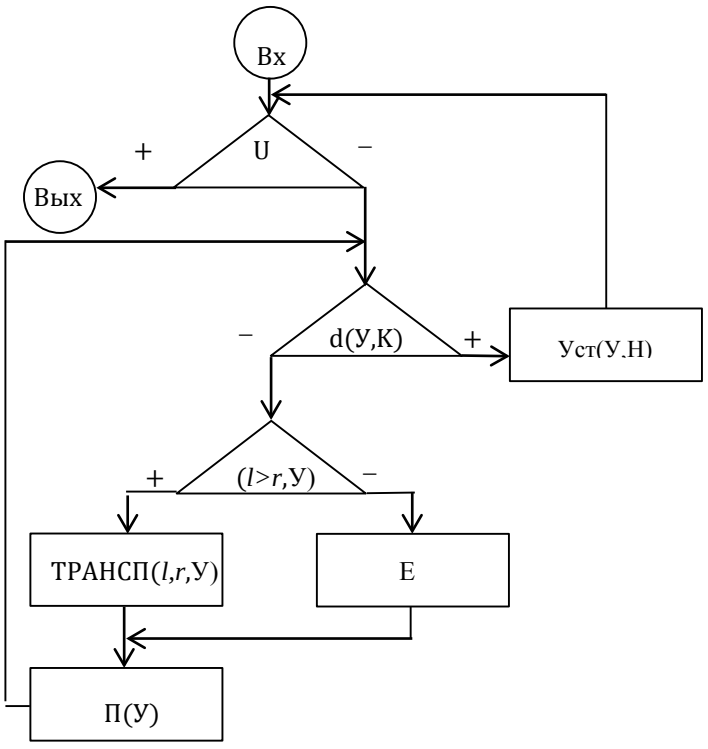
Далее, поскольку при обнаружении неупорядоченной пары соседних элементов массива эти элементы транспонируются, то это преобразование массива должен выполнять некоторый оператор $ТРАНСП(l,r,У)(M)$, такой, что:

если $M = Ha_1a_2...a_iYa_{i+1}...a_mK$, то $ТРАНСП(l,r,У)(M) = Ha_1a_2...a_{i+1}Ya_i...a_mK$.

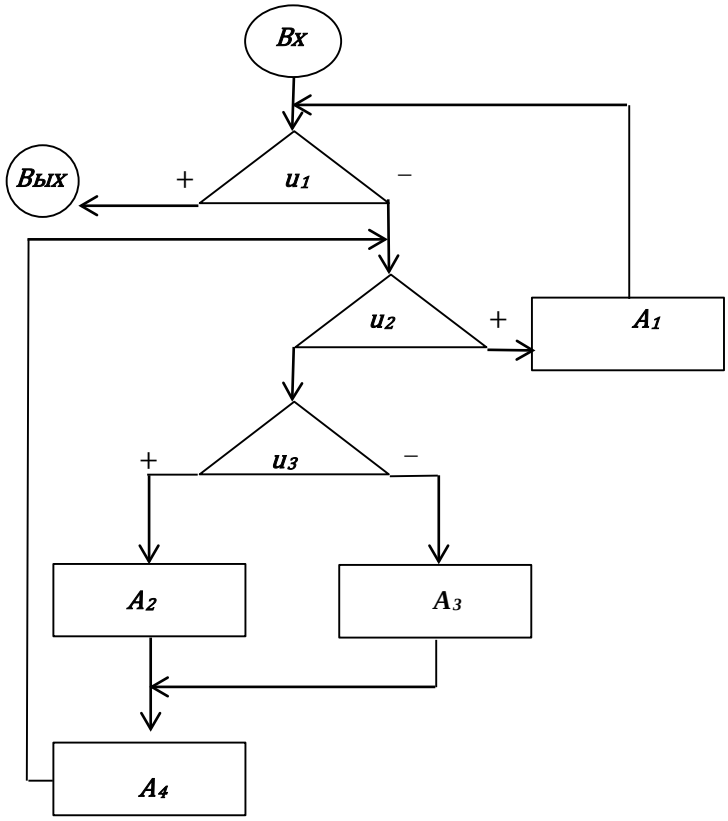
Наконец, нужен оператор, обеспечивающий передвижение указателя слева направо по массиву. Пусть подобное преобразование массива выполняет оператор ***П(У)(М)***:

если $M = Ha_1a_2...a_iYa_{i+1}...a_mK$, то $П(У)(М) = Ha_1a_2...a_ia_{i+1}Ya_{i+2}...a_mK$.

Искомая интерпретированная ГС, реализующая алгоритм «Пузырек» сортировки массивов данных определяется неоднозначно. Если использовать в ГС стандартные программистские конструкции, то одним из вариантов такой ГС алгоритма «Пузырек» является:



В этой ГС присутствует тождественный оператор ***E***. Необходимость его введения вызывается соглашением использования в ГС стандартных программистских конструкций. Если это ограничение снять, то операторную вершину ***E*** из ГС можно безболезненно удалить. Получим еще одну ГС, реализующую алгоритм «Пузырек». Неинтерпретированная ГС, результатом интерпретации которой является алгоритм «Пузырек»:



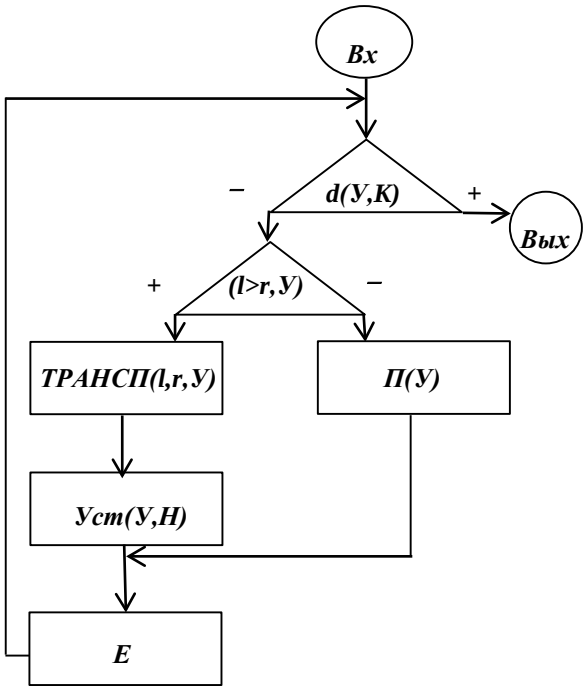
Алгебраическая система, использование которой при интерпретации последней ГС приводит к алгоритму «Пузырек» сортировки массивов данных, суть

$$AC = \langle MASS; \{Уст(У,H), ТРАНСП(l,r,У), П(У), E\} \cup \{U, d(У,K), (l > r, У)\} \rangle .$$

2.3.2. Алгоритм «Раствор». *На каждом такте работы алгоритма сравниваются между собой элементы, стоящие рядом с указателем (правее и левее указателя или соседние с указателем). Если между соседними элементами нарушения порядка нет, то указатель смещается*

вправо на один элемент массива и снова сравниваются соседние с указателем элементы массива и т.д. Если же указатель обнаруживает соседнюю пару элементов с нарушением порядка (левый элемент больше правого), то происходит перестановка этих элементов местами (транспонирование элементов) и указатель сбрасывается в начало массива. Далее процесс упорядочения массива продолжается (выполняется следующий шаг работы алгоритма). Алгоритм заканчивает свою работу, если массив становится упорядоченным.

Из описания алгоритма ясно, что можно воспользоваться элементарными операторами и предикатами, рассмотренными в предыдущем примере. Учтем еще, что алгоритм заканчивает свою работу (массив становится упорядоченным), если указатель достигает конца массива. Поэтому для реализации этого алгоритма сортировки (интерпретации соответствующей ГС) можно воспользоваться рассмотренной в предыдущем примере алгебраической системой (предикат, проверяющий упорядоченность массива можно не использовать). Одной из ГС, использующей стандартные программистские конструкции и реализующей алгоритм «Раствор» является следующая ГС:



Заметим, что в этой ГС, в отличие от предыдущей интерпретированной ГС, безболезненно удалить вершину с тождественным оператором ***E*** нельзя, ибо получим конструкцию, недопустимую в ГС (нельзя передавать управление на ребра, в свою очередь передающие управление).

2.3.3. Алгоритм «Челнок». *Алгоритм передвигает указатель слева направо по массиву до обнаружения первой неупорядоченной пары элементов, соседних с указателем. Если таковая обнаружена – происходит транспозиция этих элементов и указатель меняет направление движения – он начинает двигаться влево по массиву, переставляя местами неупорядоченные элементы, до обнаружения первой упорядоченной пары элементов. Если таковая обнаружена, то происходит изменение направления движения указателя – он начинает двигаться направо по массиву до обнаружения первой неупорядоченной пары элементов. И так далее. Алгоритм заканчивает свою работу, если массив становится упорядоченным.*

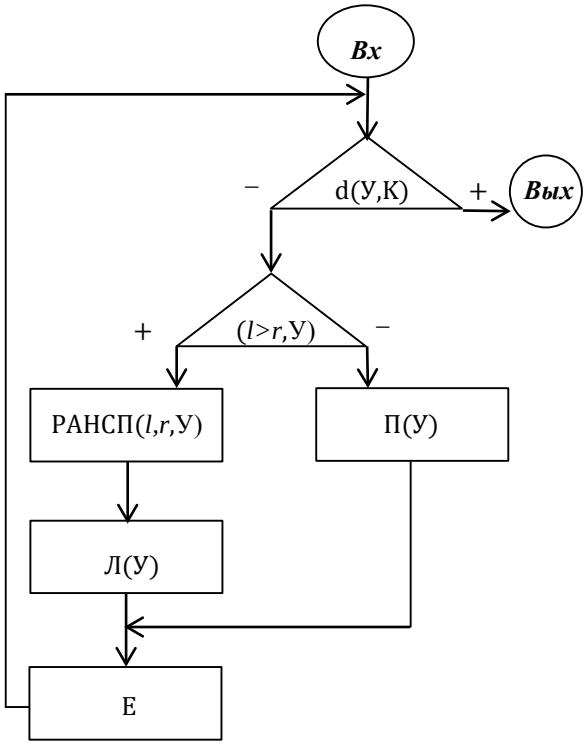
Из описания алгоритма ясно, что «Челнок» заканчивает работу, если указатель достигает конца массива. Для реализации этого алгоритма с помощью интерпретированных ГС нам понадобится (в чем несложно убедиться) следующая алгебраическая система:

$$AC = \langle MASS; \{ ТРАНСП(l, r, У), П(У), Л(У), E \} \cup \{ d(У, K), (l > r, У) \} \rangle,$$

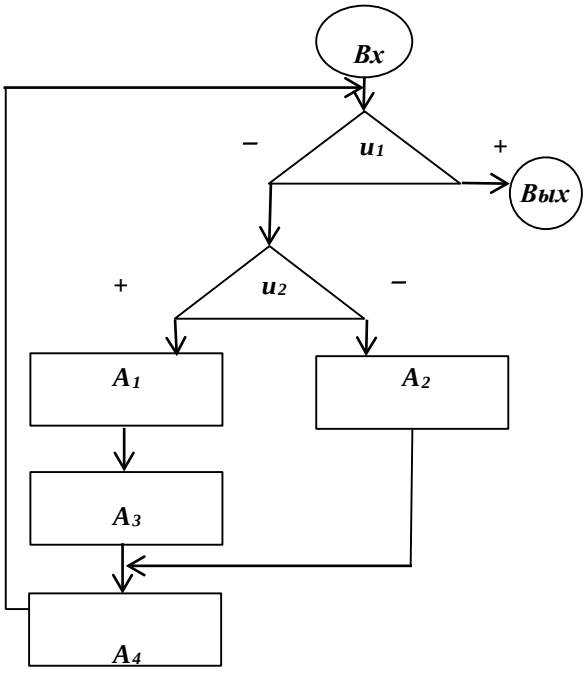
где ***Л(У)(М)*** – новый оператор (сдвига указателя на один элемент массива влево по массиву):

если $M = Ha_1a_2...a_iУa_{i+1}...a_mK$, то $Л(У)(M) = Ha_1a_2...a_{i-1}Уa_ia_{i+1}...a_mK$.

Одной из интерпретированных ГС, использующей стандартные программистские конструкции и реализующей алгоритм «Челнок» является следующая ГС:



Рассмотрим неинтерпретированные ГС, интерпретации которых приводят к алгоритмам «Раствор» и «Челнок» (схемы этих алгоритмов) и убеждаемся, что эти схемы абсолютно одинаковые (с точностью до обозначения операторных и логических переменных). Эту общую схему передает неинтерпретированная ГС:



Последняя неинтерпретированная ГС и различные алгоритмы, порождаемые этой граф-схемой в интерпретациях, свидетельствует о том, что неинтерпретированные ГС являются аналогами формул обычной математики (алгебры или анализа). Это дает возможность строить формальную теорию (метаалгебру), в которой формулами являются неинтерпретированные ГС.

2.4 Алгебра Калужнина.

Определение 2.4.1. *Алгебра Калужнина – формальная двухосновная универсальная метаалгебра, в которой основами служат множество* ***АГС*** *операторных граф-схем (операторных формул этой алгебры) и множество булевых функций* P_2*, представленных логическими ГС (логических формул алгебры), а сигнатура содержит единственную операцию – операцию* ***СУПЕР*** *суперпозиции, которая на каждой из основ* ***АГС*** *и* P_2 *является алгебраической, а на основах* ***АГС*** *и* P_2*, рассматриваемых совместно, – полиморфной, т.е.*

$$АК = (АГС, P_2; СУПЕР).$$

Операция суперпозиции ГС по сути своей ничем не отличается от описанной ранее (см. определение 1.2.1) суперпозиции формул произвольной универсальной алгебры. Пусть операторная ГС G получена из ГС G_1 в результате замены операторной переменной A на операторную ГС G_2 (обозначение $G = G_1(A \rightarrow G_2)$). Для того, чтобы практически выполнить такую суперпозицию ГС необходимо:

1) В ГС G_1 зафиксировать ***все*** операторные вершины, помеченные операторной переменной A и удалить эти вершины из ГС G_1, сохранив ребра, инцидентные этим вершинам;

2) В ГС G_2 удалить вершины ***Вход*** и ***Выход***, сохранив инцидентные им ребра;

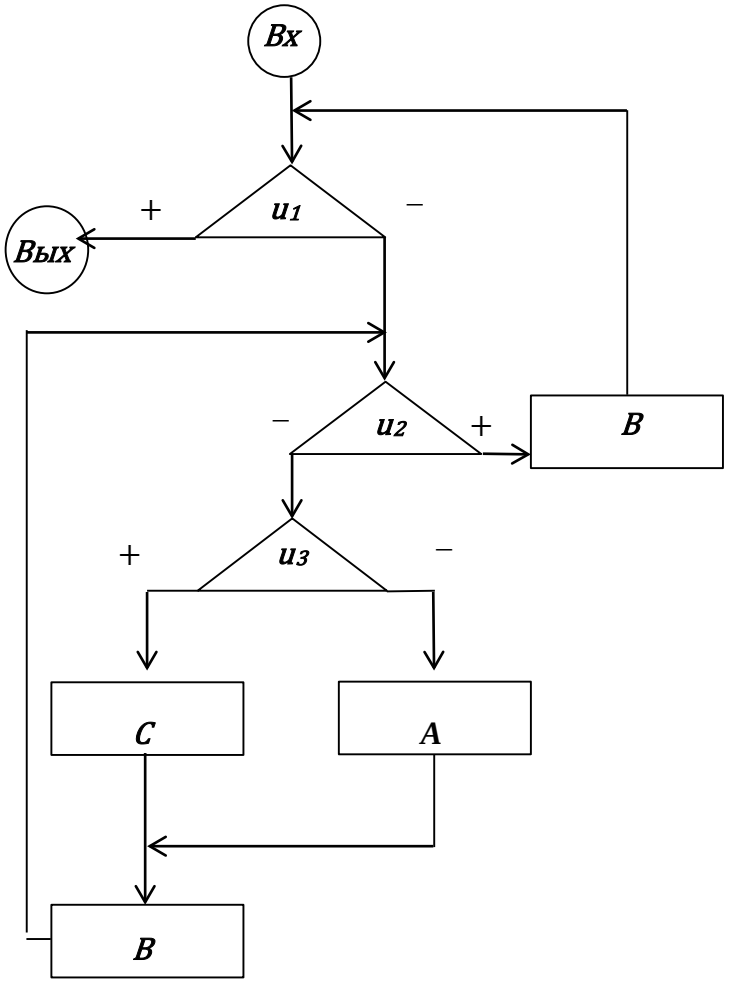
3) Подставить конструкцию (определенную в 2) на места ***всех*** пустующих операторных вершин полученной в 1) неполной ГС, «скле-

ив» (отождествив) между собой соответственно входные и выходные ребра.

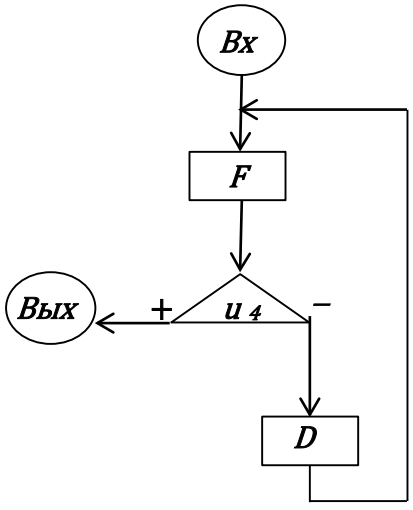
Пример 2.4.1. Пусть нам заданы ГС G_1 и G_2. Построить ГС

$$G = G_1(B \longrightarrow G_2).$$

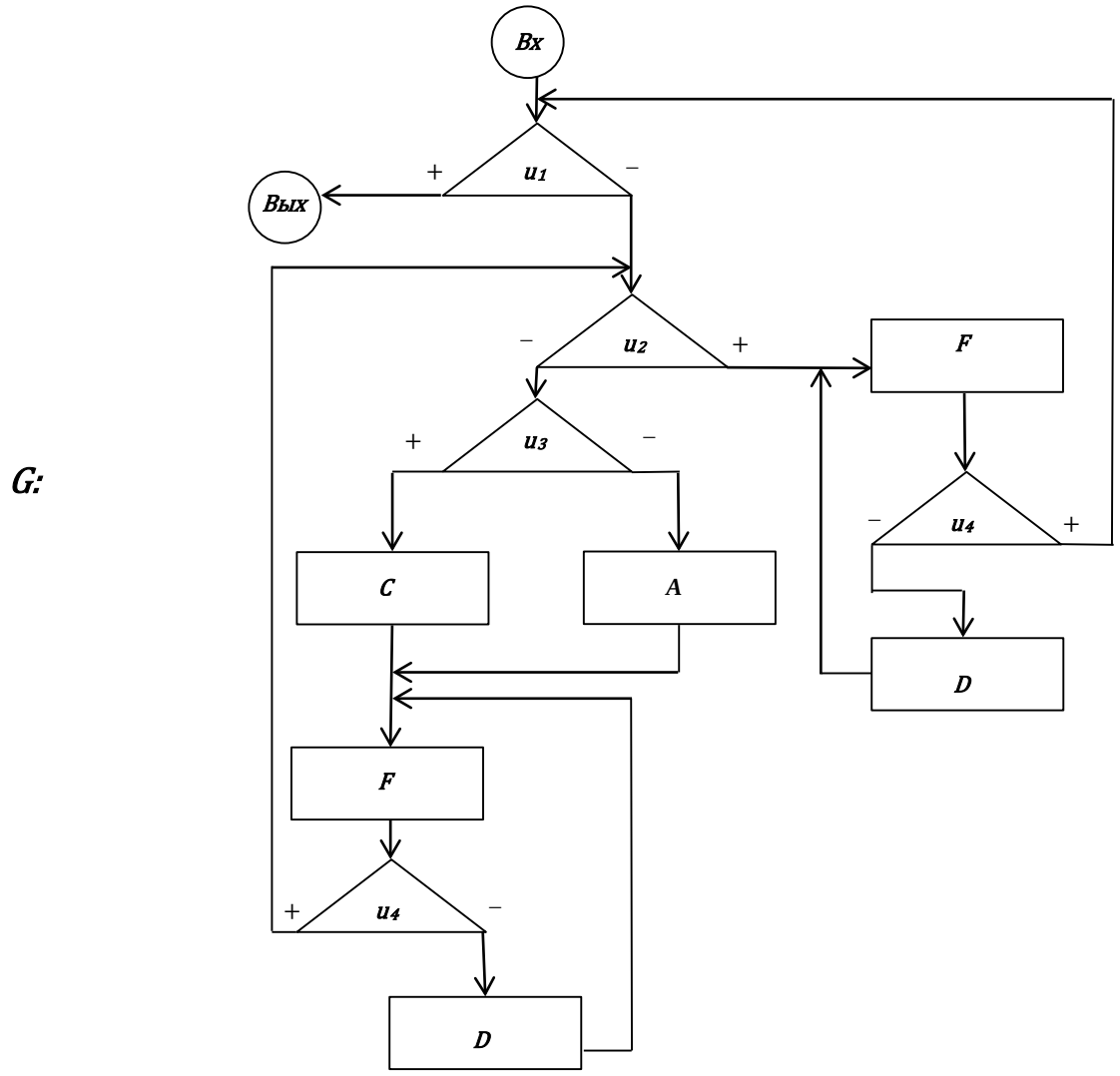
G_1:



G_2:



Тогда ГС для $G = G_1(B \rightarrow G_2)$ следующая:



Суперпозиция логических ГС строится аналогичным образом с той лишь разницей, что она строится по логическим переменным и логическим ГС и, следовательно, при «склеивании» ребер необходимо учитывать метки ребер (склеивать между собой ребра с метками « - » и, аналогично, ребра с метками «+»).

Понятно, что нельзя вместо операторной переменной вставлять логическую ГС и наоборот – вместо логической переменной вставлять операторную ГС (т.е. осуществлять перекрестную суперпозицию).

Таким образом, с помощью операции суперпозиции, из более простых ГС можно строить более сложные и, тем самым, реализовывать в

интерпретациях более сложные алгоритмы с более сложными условиями. Именно с этой целью и разработана алгебра Калужнина.

Как видим, алгебра Калужнина очень наглядна и проста в приложениях. Основным недостатком этой алгебры является невозможность поручить построение суперпозиций и интерпретаций неориентированных ГС электронно-вычислительным машинам. Кроме того, в АК нужно много места для изображения схем алгоритмов и самих алгоритмов с помощью граф-схем.

3. Алгебра Янова (АЯ).

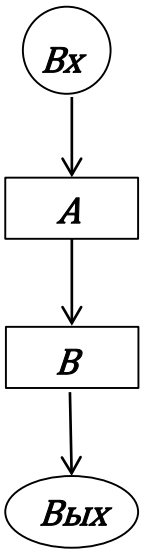
3.1. Основные программистские конструкции в алгебре Янова.

Введение основных понятий алгебры Янова проведем, используя уже известные понятия алгебры Калужнина. Граф-схемы, как формулы алгебры Калужнина, содержат кроме двух специфических вершин («Вход» и «Выход») непересекающиеся подмножества операторных и логических вершин, которые в интерпретациях превращаются в конкретные операторы и предикаты. Аналогично, в алгебре Янова рассматриваются операторные переменные и логические переменные. Сохраним их обозначения в АЯ такими же, что и в АК – операторные переменные будем обозначать большими буквами начала латинского алфавита, а логические – малыми буквами конца латинского алфавита. Последовательное выполнение операторов в интерпретированной ГС в АК определялось направлением ребер. В АЯ последовательность выполнения операторов будем обозначать обычным символом композиции отображений - ∗. Запись $A*B$ будем понимать как последовательное выполнение операторов (в интерпретациях) в порядке их следования слева направо, т.е. $(A*B)(m) = B(A(m))$. В интерпретированных ГС алгебры Калужнина в логических вершинах выполнялась проверка конкретных предикатов. Подобную проверку в алгебре Яно-

ва выполняют «условные» операторы $П(u)↓_m$ (оператор условного перехода). Если на вход условного оператора поступает некоторое входное слово a, то на нем вычисляется значение предиката (в интерпретациях) u и если $u(a)=1$ (истина), то к a применяется первый оператор подформулы, следующей за символом $m:$, если же $u(a)=0$ (ложь), то к a применяется оператор, следующий за $П(u)↓_m$ в формуле (в этом случае символ $*$ не используется). Символ $m:$ не является ни логической, ни операторной переменной. Это ***маркер*** (метка), отмечающий начало некоторой подформулы. Оператор условного перехода $П(u)↓_m$ в случае, когда условие – константа 1 (в интерпретациях тождественно истинный предикат), т.е. $П(1)↓_m$, называется оператором безусловного перехода и обозначается $П↓_m$. Если на вход безусловного оператора поступает некоторое слово a, то к a применяется первый оператор подформулы, следующей за символом $m:$ в формуле. Сказанное выше станет понятнее из следующих примеров. Выразим формулами алгебры Янова основные программистские конструкции – композицию, альтернативу и все варианты циклов.

1. Композиция :

В алгебре Калужнина :

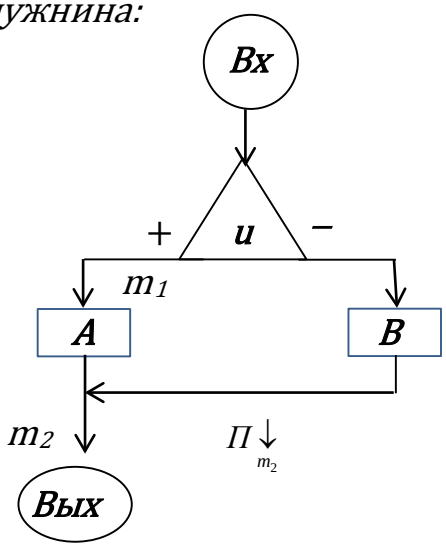


В алгебре Янова:

$A*B$

2. Альтернатива .

В алгебре Калужнина:



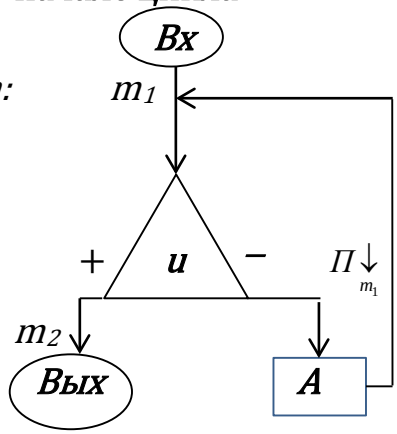
В алгебре Янова:

$$П(u)\downarrow_{m_1} B * П\downarrow_{m_2} m_1 : A * m_2 : E$$

В формуле $П(u)\downarrow_{m_1} B * П\downarrow_{m_2} m_1 : A * m_2 : E$ и далее оператор E - тождественный (единичный) оператор, т.е. ∀A: A∗E = E∗A = A.

3. Цикл с условием в начале цикла .

В алгебре Калужнина:

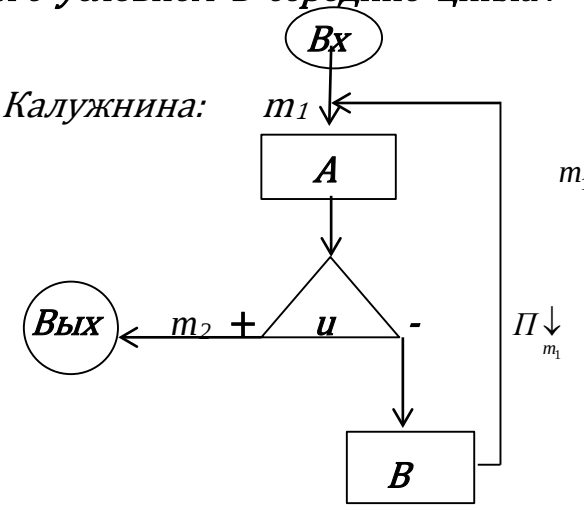


алгебре Янова:

$$m_1 : П(u)\downarrow_{m_2} A * П\downarrow_{m_1} m_2 : E$$

4. Цикл с условием в середине цикла .

В алгебре Калужнина:

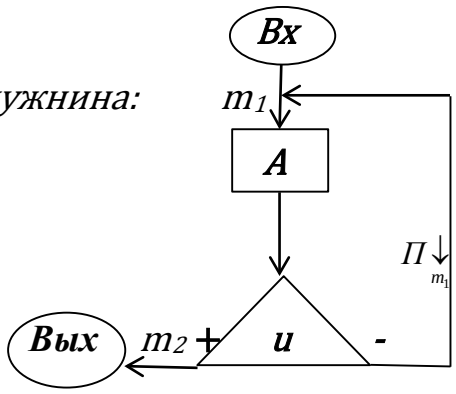


В алгебре Янова:

$$m_1 : A * П(u)\downarrow_{m_2} B * П\downarrow_{m_1} m_2 : E$$

5. Цикл с условием в конце цикла

В алгебре Калужнина:



В алгебре Янова:

$m_1 : A * \Pi(u) \downarrow_{m_2} \Pi \downarrow_{m_1} m_2 : E$

или $m_1 : A * \Pi(\bar{u}) \downarrow_{m} E$

Последовательности символов $A*B$, $\Pi(u) \downarrow_{m_1} B * \Pi \downarrow_{m_2} m_1 : A * m_2 : E$,

$m_1 : \Pi(u) \downarrow_{m_2} A * \Pi \downarrow_{m_1} m_2 : E$, $m_1 : A * \Pi(u) \downarrow_{m_2} B * \Pi \downarrow_{m_1} m_2 : E$, $m_1 : A * \Pi(u) \downarrow_{m_2} \Pi \downarrow_{m_1} m_2 : E$,

$m : A * \Pi(\bar{u}) \downarrow_{m} E$ - это примеры формул алгебры Янова.

Понятие интерпретации формулы алгебры Янова остается таким же, как и в алгебре Калужнина.

Определение 3.1.1. *Под интерпретацией формулы алгебры Янова понимается задание пары объектов* $I = \langle \tilde{A}, \varphi \rangle$, *где* $\tilde{A} = \langle D, \Sigma_0 \cup \Sigma_\pi \rangle$ *- некоторая одноосновная алгебраическая система, сигнатура которой содержит как операции (операторы), так и предикаты;* φ *- интерпретирующая функция, которая каждой операторной переменной формулы сопоставляет конкретный оператор из* Σ_0 *и каждой логической переменной сопоставляет конкретный предикат из* Σ_π.

Преобразование входных слов (входной информации) с помощью интерпретированных формул алгебры Янова, например с помощью $m_1 : A * \Pi(u) \downarrow_{m_2} B * \Pi \downarrow_{m_1} m_2 : E$, следующее. Входное слово ***a*** подается на вход алгоритма, порожденного интерпретированной формулой АЯ. К нему применяется первый в записи формулы оператор. В нашем случае

таким оператором является оператор A (расположенный самым первым в записи формулы символ m_1 является меткой (маркером) и не влияет на порядок выполнения операторов). Далее, на полученном результате $A(a)$ вычисляется значение предиката u. Если $u(A(a)) = 0$, то к $A(a)$ применяется оператор B, т.е. вычисляется $B(A(a))$. Затем к полученному результату $B(A(a))$ применяется оператор безусловного перехода $П\downarrow_{m_1}$, который отсылает $B(A(a))$ на вход оператора, следующего за меткой m_1, т.е. в нашем случае к $B(A(a))$ снова применяется оператор A, и т.д. Подобные преобразования будут происходить до тех пор, пока предикат u будет принимать ложные значения. Если же на каком-то этапе предикат примет значение 1 (истина), то слово b, на котором значение предиката стало истинным, поступит на вход оператора, следующего в формуле за меткой m_2, т.е. в нашем случае на вход тождественного оператора E. Этот оператор является последним в записи формулы, а потому результат $E(b) = b$ и есть итоговый результат преобразования исходного слова a алгоритмом, порожденным интерпретированной исходной формулой АЯ.

Из последнего примера ясно, что рассмотренный алгоритм является циклом с условием в середине. Но это можно было заметить только после анализа работы алгоритма. Таким образом, в отличие от ГС, в формулах алгебры Янова сразу нельзя определить структуру порождаемых ними алгоритмов. Поэтому в дальнейшем формулы АЯ будем называть ***неструктурными схемами*** (НС).

3.2. Преобразования формул АК в формулы АЯ и наоборот.

В предыдущем разделе формулы АЯ, передающие схемы основных программистских конструкций, были получены с помощью анализа

содержательного смысла этих схем в АК. В общем случае такой подход становится очень громоздким и не всегда возможным. А потому установим некоторые правила, с помощью которых мы сможем легко переходить от записи схем алгоритмов (а значит и самих алгоритмов) в виде ГС к их записи в виде неструктурных схем. Для этого введем в рассмотрение понятие размеченных граф-схем.

Определение 3.2.1. *Граф-схема называется размеченной, если по крайней мере одно из ее ребер снабжено какой-либо из двух меток* m_i *или* $П\downarrow_{m_i}$. *При этом разметка ГС подчиняется следующим правилам:*

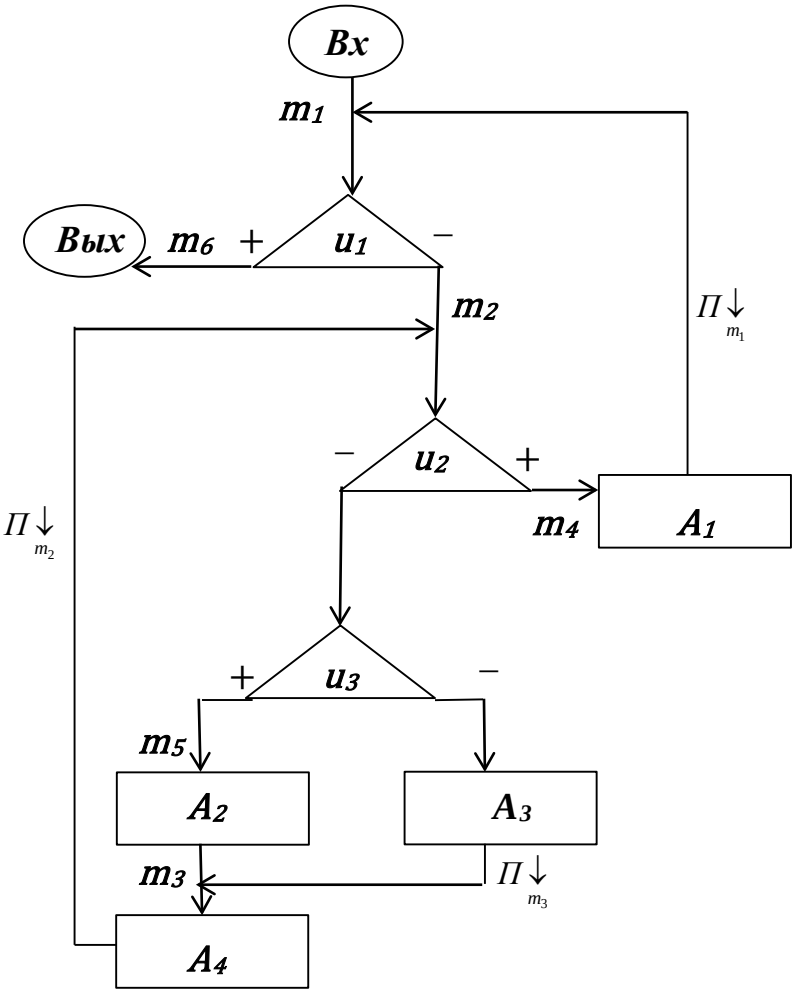
1. Ребра, на которые передается управление, снабжены метками m_i;

2. Плюсовые ребра логических вершин, ***если они не служат для передачи управления*** *на уже отмеченные метками* m_i *ребра, снабжены метками* m_i;

3. Ребра, передающие управление, но ***не плюсовые ребра логических вершин****, используемые для передачи управления, снабжены метками* $П\downarrow_{m_i}$, *где метка* m_i *совпадает с меткой ребра, на которое передается управление;*

4. Ребра могут быть отмеченными только по правилам 1÷3.

Применим правила разметки к неинтерпретированной ГС алгоритма «Пузырек» сортировки массивов данных:



На первом этапе, в соответствии с правилами разметки, в ГС появляются метки m_1, m_2, m_3. На втором – m_4, m_5, m_6. На третьем - $П\downarrow_{m_1}$, $П\downarrow_{m_2}$, $П\downarrow_{m_1}$. Разметка закончена.

Установим теперь соответствие между элементами формул алгебры Калужнина (т.е. элементами ГС) и элементами формул алгебры Янова (т.е. неструктурных схем).

В алгебре Калужнина:	*В алгебре Янова:*
начальное (первое) ребро без метки	пустой символ (нет соответствующей буквы)
отрицательное ребро логической вершины, если оно не имеет метки	пустой символ (нет соответствующей буквы)
ребро без метки	*

ребро с меткой m_i	m_i: (если этот символ стоит после ↓ в НС или является начальным в НС) или $*m_i$: (если этот символ стоит после операторной переменной в НС)
ребро с меткой $П\downarrow_{m_i}$	оператор безусловного перехода $П\downarrow_{m_i}$ (если этот символ стоит после ↓ в НС) или $*П\downarrow_{m_i}$ (если этот символ стоит после операторной переменной в НС)
операторная вершина A	операторная переменная A
«условная» (логическая) вершина u с плюсовым ребром без метки, передающим управление на ребро с меткой m_i	оператор условного перехода $П(u)\downarrow_{m_i}$
«условная» (логическая) вершина u с плюсовым ребром, помеченным меткой m_i	оператор условного перехода $П(u)\downarrow_{m_i}$

Теперь, используя соответствие между элементами формул АК и элементами формул АЯ, легко переходить от представления схем алгоритмов формулами одной алгебры к представлению этих схем формулами другой алгебры. Например, представим схему алгоритма «Пузырек» неструктурной схемой, зная его представление граф-схемой. Для этого будем продвигаться по ГС от вершины «Вход» к вершине «Выход» следуя направлениям ребер и записывать результаты такого продвижения символами алгебры Янова. Учтем еще, что

линейный порядок выполнения операторов в неструктурной схеме сохраняется при отрицательных условиях проходимых при этом операторов условного перехода (т.е. при прохождении условных вершин ГС по отрицательным ребрам). Отметим схематически первый отрезок такого пути по ГС «Пузырек» (начальный путь): Вх$\longrightarrow m_1 \longrightarrow u_1 \longrightarrow m_2 \longrightarrow u_2 \longrightarrow u_3 \longrightarrow A_3 \longrightarrow П\downarrow_{m_3}$. Нам встретилось на пути ребро с меткой $П\downarrow_{m_3}$. Оператор безусловного перехода $П\downarrow_{m_3}$ «рвет» линейный порядок выполнения операторов в НС и, в данном случае, отсылает к продолжению выполнения операторов, содержащихся в подсхеме, следующей за меткой m_3 в НС. Таким образом, часть неструктурной схемы (подсхема), соответствующая пройденному пути в ГС, имеет вид $m_1 : П(u_1)\downarrow_{m_6} m_2 : П(u_2)\downarrow_{m_4} П(u_3)\downarrow_{m_5} A_3 * П\downarrow_{m_3}$. Как видим, в полученной НС имеются четыре отсылки на подсхемы с начальными метками m_6, m_4, m_5, m_3. Поскольку ребро с меткой m_6 – выходное ребро всей ГС, то подсхему НС, для которой это ребро является начальным ребром, опишем в последнюю очередь. Остальные подсхемы с начальными метками m_4, m_5, m_3 неравноценны для описания. Ребро m_4 расположено ближе к началу ГС, а потому опишем подсхему с этим начальным ребром. Соответствующий путь: $m_4 \longrightarrow A_1 \longrightarrow П\downarrow_{m_1}$ (промежуточный путь) и, значит, соответствующая ему НС - $m_4 : A_1 * П\downarrow_{m_1}$. Пристыкуем последнюю НС справа к ранее полученной. Имеем:

$$m_1 : П(u_1)\downarrow_{m_6} m_2 : П(u_2)\downarrow_{m_4} П(u_3)\downarrow_{m_5} A_3 * П\downarrow_{m_3} m_4 : A_1 * П\downarrow_{m_1}.$$

Опишем теперь путь и соответствующую ему НС с начальной меткой m_5 (ребро с этой меткой расположено ближе к началу ГС, чем ребро с меткой m_3): $m_5 \longrightarrow A_2 \longrightarrow m_3 \longrightarrow A_4 \longrightarrow П\downarrow_{m_2}$ (промежуточный путь). Соответствующая НС: $m_5 : A_2 * m_3 : A_4 * П\downarrow_{m_2}$. Пристыкуем полученную НС к предыдущей справа, получим:

$$m_1 : \Pi(u_1) \underset{m_6}{\downarrow} m_2 : \Pi(u_2) \underset{m_4}{\downarrow} \Pi(u_3) \underset{m_5}{\downarrow} A_3 * \Pi \underset{m_3}{\downarrow} m_4 : A_1 * \Pi \underset{m_1}{\downarrow} m_5 : A_2 * m_3 : A_4 * \Pi \underset{m_2}{\downarrow}.$$

Как видим, описаны все подсхемы с начальными метками – адресами отсылки, кроме последней метки m_6. Путь: $m_6 \longrightarrow Вых$ (заключительный путь). Поскольку m_6 выходное ребро ГС, а отдельно метку m_6 в НС использовать нельзя (она должна быть начальной меткой некоторой подсхемы), то воспользуемся тождественным оператором E и тогда соответствующая заключительному пути НС: $m_6 : E$. Пристыкуем последнюю НС к предыдущей справа и окончательно получим:

$$\text{НС} = m_1 : \Pi(u_1) \underset{m_6}{\downarrow} m_2 : \Pi(u_2) \underset{m_4}{\downarrow} \Pi(u_3) \underset{m_5}{\downarrow} A_3 * \Pi \underset{m_3}{\downarrow} m_4 : A_1 * \Pi \underset{m_1}{\downarrow} m_5 : A_2 * m_3 : A_4 * \Pi \underset{m_2}{\downarrow} m_6 : E$$

.Ясно, что НС по заданной ГС определяется неоднозначно, а с точностью до порядка следования подсхем, соответствующих «промежуточным» путям в ГС. Положение подсхем, отвечающих начальному и заключительному пути в ГС, фиксировано, т.е. первая располагается в начале НС, а вторая – в конце НС.

Задавая интерпретацию полученной НС с помощью алгебраической системы

$$AC = \langle MASS; \{Уст(У, H), ТРАНСП(l, r, У), \Pi(У), E\} \cup \{U, d(У, K), (l > r, У)\} \rangle,$$

а именно, реализуя интерпретирующую функцию таблицей

u_1	U
u_2	$d(У, K)$
u_3	$(l > r, У)$
A_1	$Уст(У, H)$
A_2	$ТРАНСП(l, r, У)$
A_3	E
A_4	$\Pi(У)$

получим представление алгоритма «Пузырек» интерпретированной НС (формулой):

$$\text{«Пузырек»} = m_1 : П(U)\downarrow_{m_6} m_2 : П(d(У,К))\downarrow_{m_4} П((l > r, У)\downarrow_{m_5} E * П\downarrow_{m_3}$$

$$m_4 : Уст\,(У,Н) * П\downarrow_{m_1} m_5 : ТРАНСП(l,r,У) * m_3 : П(У) * П\downarrow_{m_2} m_6 : E$$

Видим, что представление схем алгоритмов неструктурными схемами алгебры Янова гораздо компактнее, чем граф-схемами алгебры Калужнина. Кроме того, построение суперпозиций и интерпретаций НС можно поручить ЭВМ (запрограммировать), а это уже существенное преимущество. Недостатком НС является тот факт, что по НС нельзя увидеть структуру алгоритма и, например, заранее определить, сколько альтернатив, сколько и каких циклов (подпрограмм) понадобится для программной реализации алгоритма.

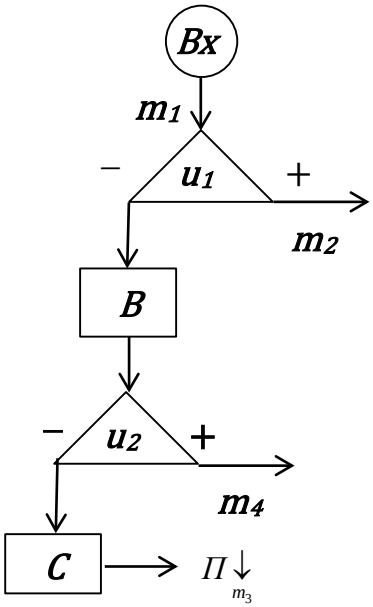
Обратный переход от неструктурных схем к граф-схемам осуществляется аналогично описанному выше переходу от ГС к НС, но в обратном порядке. Например, пусть нам задана

$$\text{НС} = m_1 : П(u_1)\downarrow_{m_2} B * П(u_2)\downarrow_{m_4} C * П\downarrow_{m_3} m_4 : A * m_3 : D * П\downarrow_{m_1} m_2 : C * П(\overline{u_3})\downarrow_{m_2} B\,.$$

Построим соответствующую ГС. Просматривая НС слева направо, выделим начальный путь в ГС, используя соответствие символов АЯ и АК:

$$\text{Вх} \longrightarrow m_1 \longrightarrow u_1 \longrightarrow B \longrightarrow u_2 \longrightarrow C \longrightarrow П\downarrow_{m_3}$$

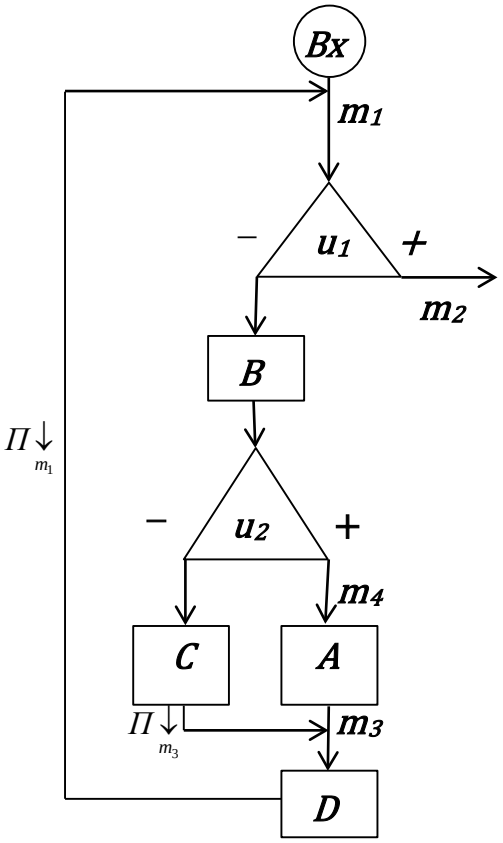
и нарисуем отвечающую ему часть ГС:



Пока неясно, плюсовые ребра передают управление на ребра с метками m_2 и m_4 или сами имеют эти метки. Начальный путь закончился оператором безусловного перехода $П\downarrow_{m_3}$. В НС после этого оператора расположена подсхема с начальной меткой m_4. Опишем соответствующий путь в ГС:

$$m_4 \longrightarrow A \longrightarrow m_3 \longrightarrow D \longrightarrow П\downarrow_{m_1}$$

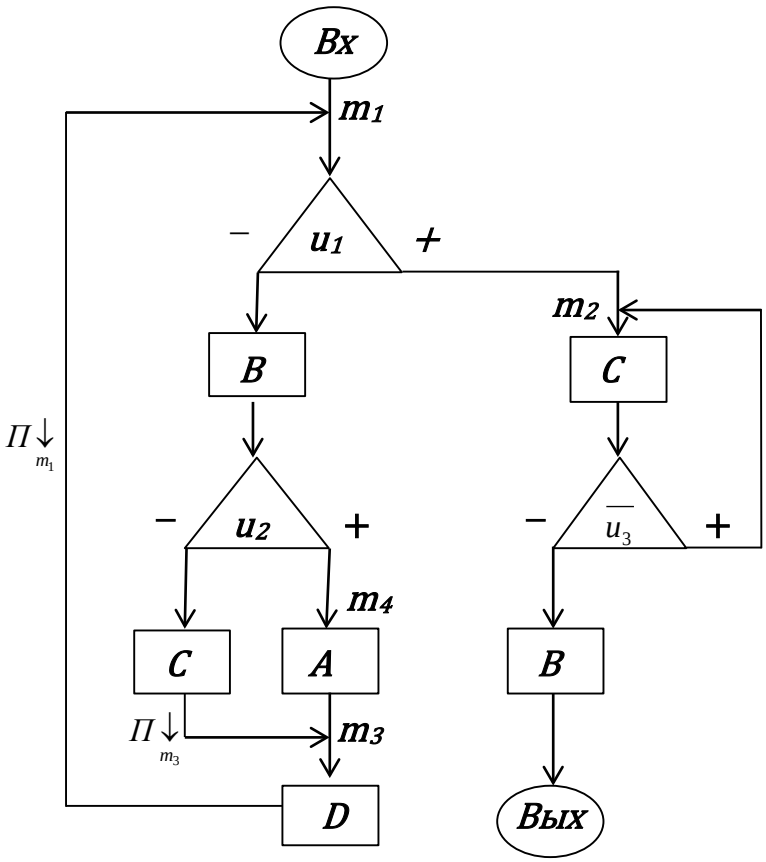
(промежуточный) и добавим в предыдущую ГС соответствующую этому пути часть ГС. Получим:



Осталось рассмотреть последний (заключительный) путь с начальным ребром m_2:

$$m_2 \longrightarrow C \longrightarrow \overline{u_3} \longrightarrow B$$

и присоединить к последней ГС соответствующую этому пути часть ГС. Получим окончательно:



Если снять разметку в последней ГС, то получим искомую ГС алгебры Калужнина, соответствующую исходной НС алгебры Янова.

Упражнение 3.2.1. *Используя ранее рассмотренные граф-схемы алгоритмов «Раствор» и «Челнок» найти их представления неструктурными схемами в алгебре Янова, указать неинтерпретированную НС, интерпретации которой приводят к указанным алгоритмам.*

3.3. Алгебра Янова.

Определение 3.3.1 *Алгебра Янова – формальная двухосновная универсальная метаалгебра, в которой основами служат множество* ***АНС*** *операторных формул этой алгебры, представленных неинтерпретированными неструктурными схемами и множество булевых функций* P_2*, представленных формулами алгебры логики (использующими, как правило, операции отрицания, дизъюнкции и конъюнкции), а сигнатура содержит единственную операцию –* ***СУПЕР*** *суперпозиции, которая на каждой из основ* ***АНС*** *и* P_2 *в отдельности яв-*

*ляется алгебраической, а на основах **АНС** и **P_2**, рассматриваемых совместно, – полиморфной, т.е.* **АЯ = (АНС, P_2; СУПЕР).**

Поскольку основы этой алгебры – множества формул в привычном понимании, то операция суперпозиции в этой алгебре – операция суперпозиции ***формул*** (см. определение 1.2.1). Эта операция является алгебраической на каждой из основ алгебры и, одновременно, полиморфной если ее рассматривать сразу на двух основах. Полиморфное ее проявление состоит в подстановке в произвольную НС вместо некоторой логической переменной произвольной булевой функции, выраженной логической формулой над булевой тройкой операций – отрицания, дизъюнкции и конъюнкции, т.е. в выполнении суперпозиции вида $G = G_1(u_i \longrightarrow f_r)$, где G_1 – некоторая НС из ***АНС***, u_i – некоторая логическая переменная в G_1, а f_r – некоторая формула из ***P_2***. Подобное применение операции суперпозиции позволяет сложные условия операторов условного перехода НС строить из более простых с помощью логических операций.

<u>Пример 3.3.1.</u>

В примере 2.4.1 была рассмотрена суперпозиция граф-схем $G = G_1(B \longrightarrow G_2)$. Рассмотрим эту суперпозицию на языке неструктурных схем. Имеем:

$$G_1 = m_1 : \Pi(u_1) \underset{m_6}{\downarrow} m_2 : \Pi(u_2) \underset{m_4}{\downarrow} \Pi(u_3) \underset{m_5}{\downarrow} A * \Pi \underset{m_2}{\downarrow} m_5 : C * m_3 : B * \Pi \underset{m_2}{\downarrow} m_4 : B * \Pi \underset{m_1}{\downarrow} m_6 : E,$$

$$G_2 = m_7 : F * \Pi(u_4) \underset{m_8}{\downarrow} D * \Pi \underset{m_7}{\downarrow} m_8 : E.$$

Тогда

$$G = G_1(B \longrightarrow G_2) = m_1 : \Pi(u_1) \underset{m_6}{\downarrow} m_2 : \Pi(u_2) \underset{m_4}{\downarrow} \Pi(u_3) \underset{m_5}{\downarrow} A * \Pi \underset{m_2}{\downarrow} m_5 : C *$$

$$* m_3 : E * m_7 : F * \Pi(u_4) \underset{m_8}{\downarrow} D * \Pi \underset{m_7}{\downarrow} m_8 : E * \Pi \underset{m_2}{\downarrow} m_4 : E * m_9 : F * \Pi(u_4) \underset{m_{10}}{\downarrow} D *$$

$$* \Pi \underset{m_9}{\downarrow} m_{10} : E * \Pi \underset{m_1}{\downarrow} m_6 : E.$$

Заметим, что непосредственная подстановка G_2 вместо B в схему G_1 (после соответствующего переобозначения меток) приводит к недопустимым в НС конструкциям $m_3{:}m_7{:}\,F$ и $m_4{:}m_9{:}\,F$. Поэтому в итоговой НС они заменены на $m_3{:}E{*}m_7{:}\,F$ и $m_4{:}E{*}m_9{:}\,F$, где E – тождественный оператор. Поскольку для любого A выполняется $E{*}A = A{*}E = A$, то последнюю НС можно упростить, отождествив метку m_7 с меткой m_3 в первом случае и m_9 с m_4 во втором. Получим

$$G = G_1(B \longrightarrow G_2) = G_1 = m_1 : \Pi(u_1)\underset{m_6}{\downarrow} m_2 : \Pi(u_2)\underset{m_4}{\downarrow} \Pi(u_3)\underset{m_5}{\downarrow} A * \Pi\underset{m_2}{\downarrow} m_5 : C *$$

$$* m_3 : F * \Pi(u_4)\underset{m_8}{\downarrow} D * \Pi\underset{m_3}{\downarrow} m_8 : E * \Pi\underset{m_2}{\downarrow} m_4 : F * \Pi(u_4)\underset{m_{10}}{\downarrow} D * \Pi\underset{m_4}{\downarrow} m_{10} : E * \Pi\underset{m_1}{\downarrow} m_6 : E$$

Убрать остальные вхождения тождественного оператора в последней НС нельзя, ибо снова получим конструкции (слова), недопустимые в АЯ: $m_8 : *\Pi\underset{m_2}{\downarrow}$ или $m_6 :$.

Пример 3.3.2.

Пусть $G_1 = m_7 : F * \Pi(u_4)\underset{m_8}{\downarrow} D * \Pi\underset{m_7}{\downarrow} m_8 : E$ и $f = u_1u_2 \vee u_3$.

Тогда $G = G_1(u_4 \longrightarrow f) = m_7 : F * \Pi(u_1u_2 \vee u_3)\underset{m_8}{\downarrow} D * \Pi\underset{m_7}{\downarrow} m_8 : E$.

Алгебра Янова и Алгебра Калужнина эквивалентны в смысле своих изобразительных возможностей – любой алгоритм, схема которого выражается некоторой ГС, представим неинтерпретированной неструктурной схемой в алгебре Янова и наоборот. Наш вывод основывается на ранее рассмотренных взаимных переходах от ГС к НС и наоборот. При этом и в первой и во второй алгебре формулы ***необязательно*** основываются на стандартных программистских конструкциях.

4. Алгебра Дэйкстры (АД).

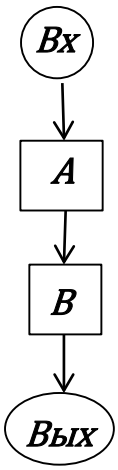
4.1 Основные программистские конструкции в алгебре Дэйкстры.

Введение основных понятий алгебры Дэйкстры проведем аналогично введению основных понятий алгебры Янова, используя уже извест

ные понятия алгебры Калужнина. Выразим формулами алгебры Дэйкстры основные программистские конструкции – композицию, альтернативу и все варианты циклов.

1. Композиция :

В алгебре Калужнина :

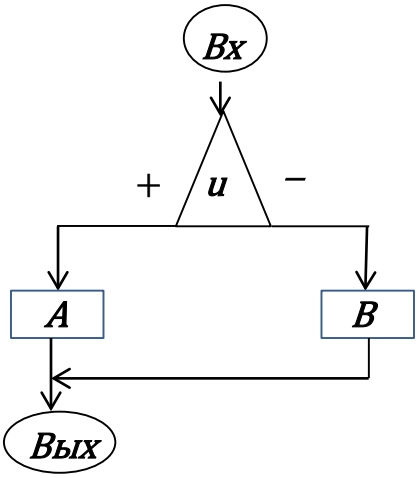


В алгебре Дэйкстры:

$A*B$

2. Альтернатива .

В алгебре Калужнина:



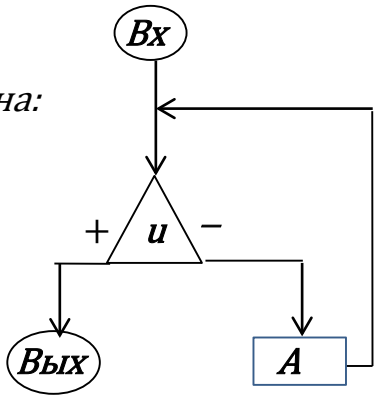
В алгебре Дэйкстры:

$([u]A, B)$

В формуле $([u]A, B)$ алгебры Дэйкстры, выражающей альтернативу, u - условие альтернативы, A - «положительная» ветвь альтернативы, B - «отрицательная» ветвь альтернативы.

***3. Цикл с условием в начале цикла*.**

В алгебре Калужнина:

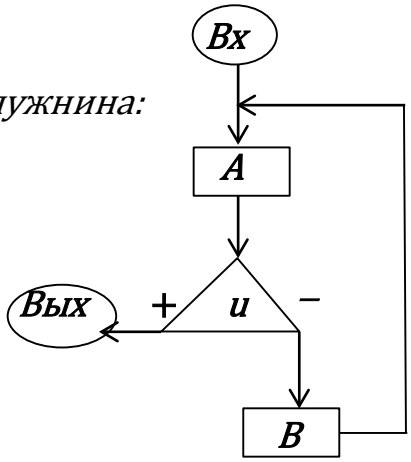


В алгебре Дэйкстры:

$\{[u]A\}$

***4. Цикл с условием в середине цикла*.**

В алгебре Калужнина:

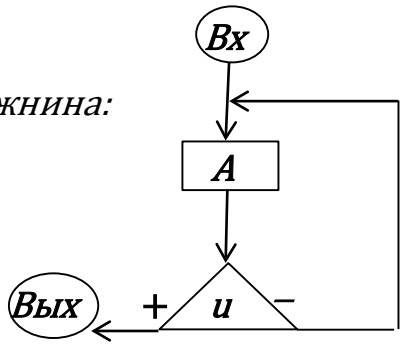


В алгебре Дэйкстры:

$\{A[u]B\}$

***5. Цикл с условием в конце цикла*.**

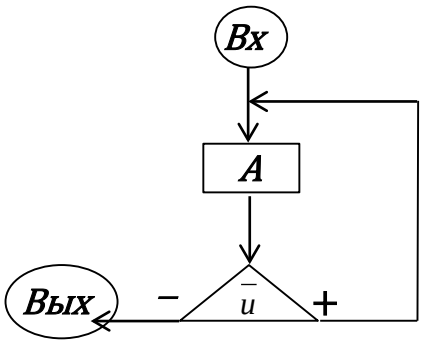
В алгебре Калужнина:



В алгебре Дэйкстры:

$\{A[u]\}$

Заметим, что *в алгебре Дэйкстры тела циклов выполняются только по отрицательным (ложным) условиям*. Поэтому цикл в алгебре Калужнина вида



должен быть преобразован к предыдущему виду заменой условия на его отрицание и соответствующим переобозначением ребер. И только после такого преобразования можно искать его выражение (формулу) в алгебре Дэйкстры.

Последовательности символов $A*B$, $([u]A, B)$, $\{[u]A\}$, $\{A[u]B\}$, $\{A[u]\}$ - примеры формул Алгебры Дэйкстры. Как видим, формулы основных программистских конструкций в алгебре Дэйкстры выглядят проще, чем в алгебре Янова.

Понятие интерпретации формул алгебры Дэйкстры в точности совпадает с введенным ранее для формул алгебры Янова (см. определение 3.1.1). Процесс преобразования слов (информации) интерпретированными формулами АД аналогичен ранее описанному в алгебре Калужнина (с учетом структуры формул). Преобразование входных слов интерпретированными СС осуществляется слева направо.

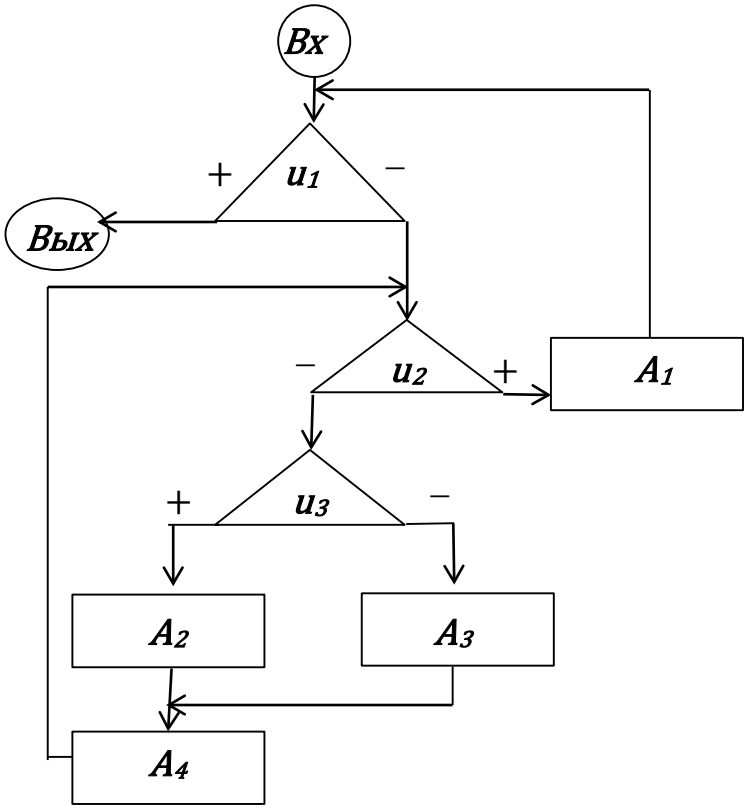
Например, пусть нам необходимо применить к входному слову m цикл с условием в середине цикла $\{A[u]B\}$. При выполнении такого цикла сначала ко входному слову m применяется оператор A (предоператор), затем на полученном слове $A(m)$ проверяется условие цикла u (вычисляется значение предиката) и в случае его невыполнения (значение предиката 0) продолжается выполнение тела цикла – к слову $A(m)$ применяется оператор B (постоператор) после чего полученное слово $B(A(m))$ снова подается на вход цикла. Выход из цикла происходит в том случае, когда условие цикла выполняется (предикат примет значение 1 (истина)).

В отличие от формул алгебры Янова формулы алгебры Дэйкстры передают структуру схемы алгоритма, т.е. по их виду можно судить, какая из основных программистских конструкций используется и в каком месте формулы. Поэтому формулы АД называются ***структурными схемами (СС)***.

4.2 Преобразования формул АК в формулы АД и наоборот.

Для преобразования формул АК в формулы АД мы используем соответствие базовых формул этих алгебр, отражающих основные программистские конструкции (см. 4.1). Ясно, что если ГС содержит конструкции (подсхемы) не совпадающие с основными программистскими конструкциями (а это вполне допустимо), то преобразовать такую ГС в СС невозможно. Таким образом, по своим изобразительным возможностям алгебра Калужнина шире, чем алгебра Дэйкстры.

Рассмотрим на конкретном примере идею перехода от ГС к СС. Рассмотрим неинтерпретированную ГС алгоритма «Пузырек» и преобразуем ее в структурную схему.



На начальное ребро ГС передается управление, замыкающее некоторый путь в ГС на его начало (например, $u_1 \longrightarrow u_2 \longrightarrow A_1 \longrightarrow u_1$). Следовательно, внешней конструкцией является цикл. Теперь важно определиться с типом цикла, т.е. какая из логических переменных является условием цикла. В ГС имеется три логические вершины. Вершина u_3 характерна тем, что открывает два пути из этой вершины, смыкающиеся ниже. Таким образом, u_3 – условие альтернативы. Мы можем сразу описать эту альтернативу структурной схемой $([u_3]A_2, A_3)$. Вершина u_2 отмеченным выше свойством альтернативы не обладает. Значит u_2 - условие цикла, но не начального, ибо существует замкнутый путь (цикл) $u_2 \longrightarrow u_3 \longrightarrow A_2 \longrightarrow A_4 \longrightarrow u_2$, не проходящий через вершину u_1. Итак, условие начального цикла - u_1. Таким образом, внешний цикл – цикл с условием в начале цикла. Если обозначить тело этого цикла через F_1, то СС внешнего цикла – $\{[u_1]F_1\}$. Разберемся подробнее с телом этого цикла, т.е. с F_1. Поскольку по положительному ребру вершины u_1 происходит выход из цикла и всей ГС, то тело внешнего цикла начинается первым встреченным оператором (вершиной) по отрицательному ребру вершины u_1. Таковой вершиной является вершина u_2, которая, как отмечалось ранее, является начальной вершиной внутреннего цикла с условием в начале цикла. Тело этого внутреннего цикла обозначим через F_2. Тогда $F_1 = \{[u_2]F_2\} * A_1$. Тем самым, $CC = \{[u_1]\{[u_2]F_2\} * A_1\}$. Остается определиться с F_2. F_2 представляет собой композицию описанной выше альтернативы и оператора A_4, т.е. $F_2 = ([u_3]A_2, A_3) * A_4$. Следовательно, окончательно получим: $CC = \{[u_1]\{[u_2] ([u_3]A_2, A_3) * A_4 \} * A_1\}$. Интерпретируя соответствующим образом последнюю СС с использованием

$$AC = \langle MASS; \{Уст(У, Н), ТРАНСП(l, r, У), П(У), E\} \cup \{U, d(У, К), (l > r, У\} \rangle),$$

получим алгоритм «Пузырек» в алгебре Дэйкстры:

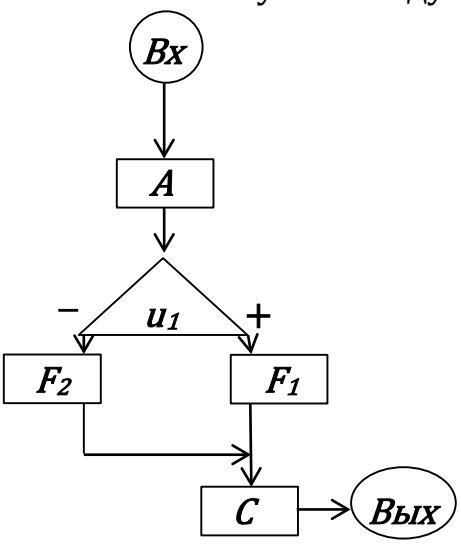
«Пузырек» $:= \{[U]\{[d(У,К)]([(l > r, У)]ТРАНСП(l, r, У), E) * П(У)\} * Уст(У, Н)\}$

Упражнение 4.2.1. Найти представление схемы алгоритмов сортировки «Челнок» и «Раствор» структурной схемой в алгебре Дэйкстры и сами эти алгоритмы в АД.

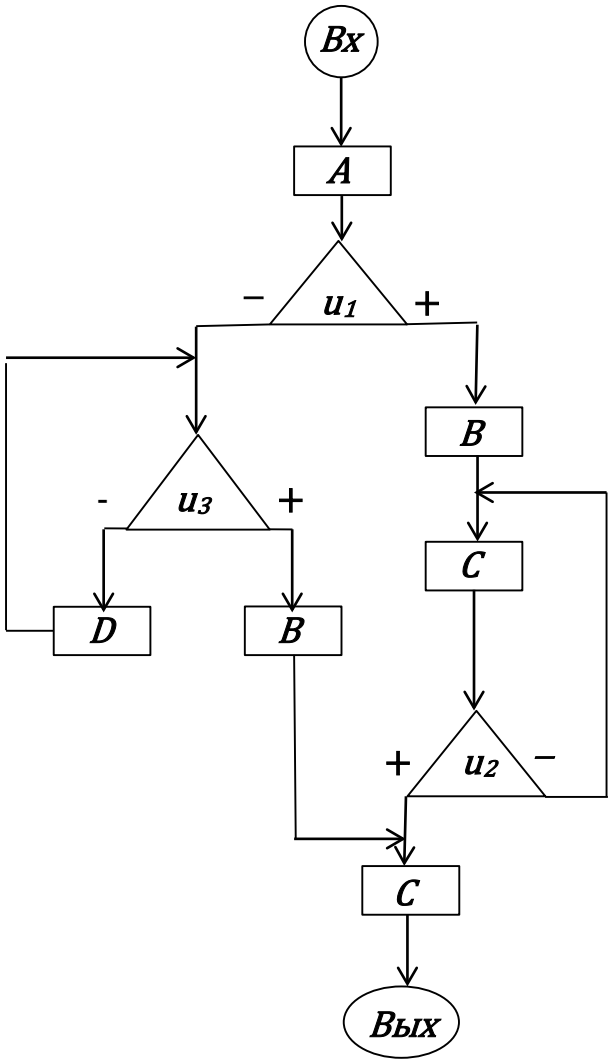
Обратный переход от СС к ГС осуществляется аналогичным образом, только рассуждения строятся в обратном направлении. Например, пусть нам задана СС

$$CC = A*([u_1]B*\{C[u_2]\},\{[u_3]D\}*B)*C$$

и требуется найти соответствующую ей граф-схему. Просматриваем СС слева направо и находим первую букву в записи формулы. Таковой является операторная переменная ***A***. Операторной переменной в ГС соответствует операторная вершина. Буква ∗ в алгебре Дэйкстры означает операцию композиции (последовательное выполнение операторов). Значит, в алгебре Калужнина ей соответствует ребро, исходящее из ранее выделенной операторной вершины ***A***. Следующим символом СС является открывающая круглая скобка. В АД круглые скобки используются только для передачи альтернатив. Обозначим на этом этапе ветви альтернатив через F_1 и F_2 (положительную и отрицательную соответственно), найдем условие альтернативы u_1 и ее конец (соответствующую закрывающую круглую скобку). Видим, что после альтернативы последовательно выполнится оператор С. Таким образом, на этом этапе анализа СС мы получим следующую ГС:



Далее, рассмотрим положительную ветвь альтернативы F_1 в СС (подформула в альтернативе, расположенная между условием альтернативы и запятой, разделяющей ветви альтернативы) $F_1 = B*\{C[u_2]\}$. Видим, что в F_1 последовательно выполняются оператор B и цикл с условием u_2 в конце цикла и телом C. Аналогично, рассматривая отрицательную ветвь альтернативы $F_2 = \{[u_3]D\}*B$ видим, что в F_2 последовательно выполняются цикл с условием u_3 в начале цикла и телом D, и оператор B. Преобразуя ветви альтернативы в соответствующие ГС, получим окончательно искомую ГС:



4.3 Преобразования формул АД в формулы АЯ и наоборот.

Ранее мы отмечали связь формул АК и АЯ, АК и АД, передающих основные программистские конструкции. Тем самым, мы неявно по-

лучили законы преобразования формул АД (структурных схем) в формулы АЯ (неструктурные схемы). Укажем эти законы.

$$A*B \;=\; A*B;$$

$$([u]A,\, B) \;=\; \Pi(u)\underset{m_1}{\downarrow} B * \Pi\underset{m_2}{\downarrow} m_1 : A * m_2 : E\,;$$

$$\{[u]A\} \;=\; m_1 : \Pi(u)\underset{m_2}{\downarrow} A * \Pi\underset{m_1}{\downarrow} m_2 : E\,;$$

$$\{A[u]B\} \;=\; m_1 : A * \Pi(u)\underset{m_2}{\downarrow} B * \Pi\underset{m_1}{\downarrow} m_2 : E\,;$$

$$\{A[u]\} \;=\; m_1 : A * \Pi(u)\underset{m_2}{\downarrow} \Pi\underset{m_1}{\downarrow} m_2 : E \;=\; m : A * \Pi(\bar{u})\underset{m}{\downarrow} E\,.$$

Рассмотрим пример их применения при формальном преобразовании формулы АД в формулу АЯ. Пусть нам задана

$$CC = A*([u_1]B*\{C[u_2]\},\{[u_3]D\}*B)*C$$

и требуется найти аналог этой формулы в алгебре Янова, т.е. найти соответствующую НС. Просматривая СС слева направо, обнаруживаем первую последовательность символов, недопустимую в АЯ. В нашем случае таковой является последовательность букв « $*($ ». Значит, следующий за « $*$ » символ « $($ » является символом формулы АД. В АД он означает начало альтернативы. Поэтому выделяем в СС условие и ветви альтернативы и применяем закон, выражающий альтернативу АД формулой АЯ. Получим:

$$CC = A*([u_1]B*\{C[u_2]\},\{[u_3]D\}*B)*C \;=$$

$$= A*\Pi(u_1)\underset{m_1}{\downarrow}\{[u_3]D\}*B*\Pi\underset{m_2}{\downarrow} m_1 : B*\{C[u_2]\}*m_2 : E*C \;=$$

Далее, просматривая полученное слово справа налево, обнаруживаем открывающую фигурную скобку – символ, не использующийся при записи формул АЯ. В алгебре Дэйкстры фигурные скобки используются для записи циклов. Видим, что встретившийся нам цикл – это цикл с условием в начале цикла. Преобразуем его в НС, используя соответствующий закон для такого цикла с выбором номеров меток, продолжающих нумерацию меток последнего полученного слова. Получим:

$$= A * П(u_1)\downarrow_{m_1} m_3 : П(u_3)\downarrow_{m_4} D * П\downarrow_{m_3} m_4 : E * B * П\downarrow_{m_2} m_1 : B * \{C[u_2]\} * m_2 : E * C =$$

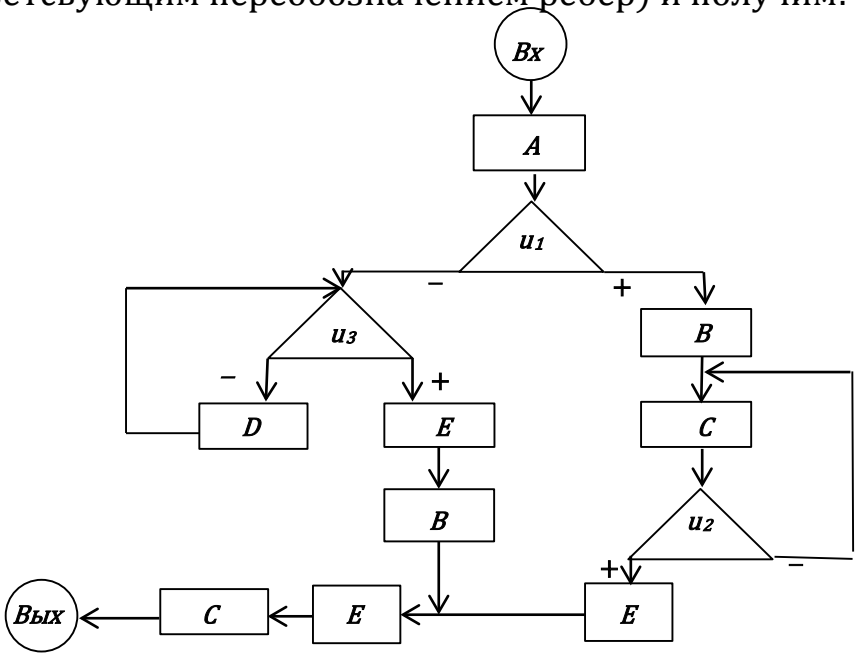
Снова просматриваем полученное слово справа налево, обнаруживаем открывающую фигурную скобку и, аналогично предыдущему, определяем тип цикла (цикл с условием в конце цикла) и применяем соответствующий закон. Получим:

$$= A * П(u_1)\downarrow_{m_1} m_3 : П(u_3)\downarrow_{m_4} D * П\downarrow_{m_3} m_4 : E * B * П\downarrow_{m_2} m_1 : B * m_5 : C * П(\overline{u_2})\downarrow_{m_5} E * m_2 : E * C,$$

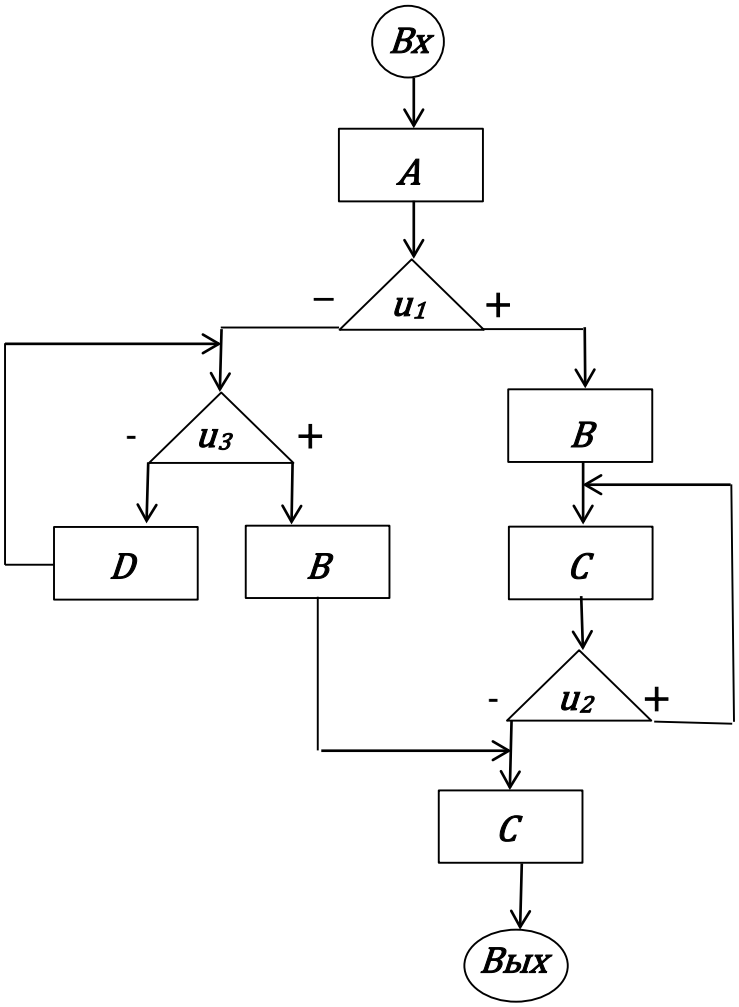
Просматривая справа налево последнее полученное слово видим, что оно не содержит символов и их последовательностей, запретных для алгебры Янова, т.е. является формулой АЯ. Тем самым, процесс преобразования формулы АД в формулу АЯ завершен. Ответ:

НС $= A * П(u_1)\downarrow_{m_1} m_3 : П(u_3)\downarrow_{m_4} D * П\downarrow_{m_3} m_4 : E * B * П\downarrow_{m_2} m_1 : B * m_5 : C * П(\overline{u_2})\downarrow_{m_5} E * m_2 : E * C.$

Обратное преобразование формул АЯ (т.е. НС) в формулы АД (т.е. в СС) напрямую невозможно. Подобный переход можно осуществить только с промежуточным использованием алгебры Калужнина. Поставим перед собой задачу преобразования последней НС в СС. Построим сначала ГС, отвечающую заданной НС, удалим в ней разметку, в цикле по условию $\overline{u_2}$ заменим условие на противоположное (с соответсвующим переобозначением ребер) и получим:



Как видим, вхождения тождественного оператора в последнюю ГС можно безболезненно удалить. Получим в итоге:



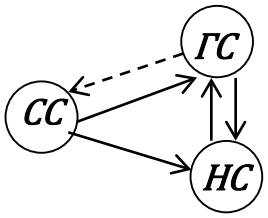
Теперь, переходя от ГС к СС, получим:

$$CC = A*([u_1]B*\{C[u_2]\},\{[u_3]D\}*B)*C$$

Формальное (аналитическое) преобразование структурной схемы в неструктурную дает многочисленное присутствие тождественного оператора E в итоговой НС. Это вызывается особенностями тождеств, связывающих основные программистские конструкции в алгебрах Дэйкстры и Янова – каждое из них заканчивается тождественным оператором. Избавиться от лишних вхождений этого оператора можно непосредственно в НС (но нужен некоторый опыт) или опосредованно, с помощью соответствующей ГС (где «лишние» вхождения

тождественного оператора наглядны), как это было сделано в предыдущем примере.

В заключение отметим на орграфе возможности преобразования формул различных алгебр друг в друга. При этом, непрерывное ребро означает безусловную возможность такого преобразования, а пунктирное - преобразование возможно не всегда.



4.4. Алгебра Дэйкстры.

Определение 3.3.1 *Алгебра Дэйкстры – формальная двухосновная универсальная метаалгебра, в которой основами служат множество* ***АСС*** *операторных формул этой алгебры, представленных неинтерпретированными структурными схемами и множество булевых функций* P_2*, представленных формулами алгебры логики (использующими, как правило, операции отрицания, дизъюнкции и конъюнкции), а сигнатура содержит единственную операцию* ***СУПЕР*** *суперпозиции, которая на* ***АСС*** *и на* P_2 *в отдельности является алгебраической, а на основах* ***АСС*** *и* P_2 *, рассматриваемых совместно, – полиморфной, т.е.*

$$АД = (АСС, P_2; СУПЕР).$$

Поскольку основы этой алгебры – множества формул в привычном понимании, то операция суперпозиции в этой алгебре – операция суперпозиции ***формул*** (см. определение 1.2.1). Эта операция является алгебраической на каждой из основ алгебры и, одновременно, полиморфной если ее рассматривать сразу на двух основах. Полиморфное ее проявление состоит в подстановке в произвольную СС вместо некоторой логической переменной произвольной булевой функции, вы-

раженной логической формулой над булевой тройкой операций – отрицания, дизъюнкции и конъюнкции, т.е. в выполнении суперпозиции вида $G = G_1(u_i \longrightarrow f_r)$, где G_1 – некоторая СС из ***ACC***, u_i – некоторая логическая переменная в G_1, а f_r – некоторая формула из P_2. Подобное применение операции суперпозиции позволяет сложные условия СС строить из более простых с помощью логических операций.

Пример 4.4.1. Пусть G_1 и G_2 – неинтерпретированные СС алгоритмов примеров 2.4.1 и 3.3.1, т.е. $G_1 = \{[u_1]\{[u_2]([u_3]C, A)*B\}*B\}$, $G_2 = \{F[u_4]D\}$. Тогда:

$$G = G_1(B \longrightarrow G_2) = \{[u_1]\{[u_2]([u_3]C, A)* \{F[u_4]D\}\}*\{F[u_4]D\}\}$$

Пример 4.4.1. Пусть $G_2 = \{F[u_4]D\}$ и $f = u_1u_2 \vee u_3$ - структурная схема и булева функция примера 3.3.2. Тогда:

$$G = G_1(u_4 \longrightarrow f) = \{F[u_1u_2 \vee u_3]D\}.$$

5. Алгебра Глушкова (АГ).

5.1. Еще раз об алгебре Дэйкстры.

Ранее рассмотренные нами алгебры Калужнина, Янова и Дэйкстры использовались на уровне их метаалгебр. Алгебра Глушкова – это модифицированная алгебра Дэйкстры. Для того, чтобы понять суть модификации, нам придется вернуться к ***определению*** алгебры Дэйкстры, т.е. рассмотреть ее на самом нижнем уровне формализации.

Определение 5.1.1. С формальной точки зрения алгебра Дэйкстры – это двухосновная универсальная алгебра $АД = \langle \mathcal{A}, \mathcal{U}; \Sigma_0 \rangle$, где $\mathcal{A}$ - непустое множество «операторных» переменных, $\mathcal{U}$ – непустое множество логических (пропозициональных) переменных, Σ_0 - сигнатура алгебры, включающая операции:

1) композицию $A*B: \mathcal{A} \times \mathcal{A} \longrightarrow \mathcal{A}$, - бинарную, алгебраическую на $\mathcal{A}$;

2) альтернативу $([u]A, B): U \times A \times A \to A$, - тернарную, полиморфную;

3) цикл с условием в начале цикла $\{[u]A\}: U \times A \to A$, - бинарную, полиморфную;

4) цикл с условием в середине цикла $\{A[u]B\}: A \times U \times A \to A$, - тернарную, полиморфную;

5) цикл с условием в конце цикла $\{A[u]\}: A \times U \to A$, - бинарную, полиморфную;

6) отрицание $\bar{x}: U \to U$, - унарную, алгебраическую;

7) дизъюнкцию $x \vee y: U \times U \to U$, - бинарную, алгебраическую;

8) Конъюнкцию $x \wedge y: U \times U \to U$, - бинарную, алгебраическую;

В настоящее время алгебра Дэйкстры и ее обобщение – алгебра Глушкова (на уровне их метаалгебр) являются наиболее разработанными алгебраическими системами в алгоритмике и широко используются в макропрограммировании.

Если рассмотреть АД на следующем уровне формализации (втором), то $АД = \langle \mathcal{ACC}, P_2; \Sigma_0 \rangle$, где $\mathcal{ACC}$ – множество формул АД (структурных схем), а P_2 – множество булевых функций, представленных формулами алгебры логики в булевской тройке операций – отрицания, дизъюнкции и конъюнкции. Сигнатура Σ_0 этой алгебры совпадает с сигнатурой, данной в вышеприведенном определении АД, с соответствующим обобщением операций над элементами основ на операции над формулами.

5.2. Алгебра Глушкова (АГ).

Определение 5.2.1. Алгеброй Глушкова называется двухосновная универсальная алгебра $АГ = \langle A, U; \Sigma_0 \rangle$, где A - непустое множество «операторных» переменных, U – непустое множество логических (пропозициональных) переменных, Σ_0 - сигнатура алгебры, включа-

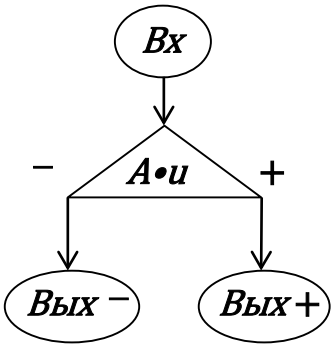
ющая операции 1) - 8) алгебры Дэйкстры и еще одну дополнительную операцию «прогнозирования»:

9) прогноз $A \bullet u$: $\mathcal{A} \times \mathcal{U} \to \mathcal{U}$, - бинарную, полиморфную.

Рассмотрим более подробно операцию прогнозирования в АГ. В интерпретациях A - некоторый конкретный оператор, а u - конкретный предикат. Тогда операция прогнозирования $A \bullet u$, примененная к указанной паре оператор-предикат, порождает новый предикат $u_1 = A \bullet u$, который определяется следующим образом:

$$u_1(m) = (A \bullet u)(m) = \begin{cases} 1, \text{если } u(A(m)) = 1 \\ 0, \text{если } u(A(m)) = 0 \end{cases}$$

для любого входного слова m. Таким образом, для того, чтобы вычислить значение предиката u_1 на конкретном входном слове m, необходимо сначала к этому входному слову применить оператор A (т.е. найти $A(m)$), затем на полученном слове $A(m)$ найти значение предиката u (т.е. найти $u(A(m))$) и это значение объявить значением предиката u_1 на исходном входном слове m. Операцию прогнозирования можно для наглядности изобразить модифицированной граф-схемой:



Эта граф-схема отличается от ранее рассматриваемых тем, что «условная» вершина обозначена не пропозициональной логической переменной, а в ней реализуется операция «прогноз» и имеет соответствующее обозначение. Такого типа граф-схемы, в которых допуска-

ется двоякое обозначение «условных» вершин будем называть ***обобщенными граф-схемами.***

Введение операции прогнозирования в сигнатуру АД приводит к расширению понятия формулы в этой алгебре. Теперь формулами новой алгебры будут не только СС, логические формулы, но и конструкции, возникающие в результате применения операции прогнозирования к операторным и логическим формулам, т.е. конструкции вида G•f. Тем самым, строится новая алгебра – алгебра Глушкова (АГ). Операторные формулы АГ будем называть ***регулярными схемами.***

Определение 5.2.2. *Алгебра Глушкова – формальная двухосновная универсальная метаалгебра, в которой основами служат множество* ***АРС*** *операторных формул этой алгебры, представленных неинтерпретированными регулярными схемами и множество* ***УС*** *- множество логических формул, включающее множество булевых функций* P_2*, представленных формулами алгебры логики (использующими операции отрицания, дизъюнкции и конъюнкции), и этими же формулами, в которых на местах некоторых пропозициональных переменных используются результаты применения операции прогнозирования над формулами этой алгебры. Сигнатура содержит единственную операцию – операцию* ***СУПЕР*** *суперпозиции, которая на* ***АРС*** *и на* ***УС*** *в отдельности является алгебраической, а на основах* ***АРС*** *и* ***УС,*** *рассматриваемых совместно, – полиморфной, т.е.*

$$АГ = (АРС, УС; СУПЕР).$$

Поскольку формулы АГ – формулы в привычном понимании, то действие операции суперпозиции в этой алгебре и интерпретация формул АГ аналогичны, например, подобным действиям над формулами алгебры Дэйкстры (см. 4.4 и 4.2).

Рассмотрим примеры неинтерпретированной регулярной схемы и ее интерпретацию на основе схемы алгоритма «Пузырек». Пусть $M=НУa_1a_2...a_mК$ - размеченный массив данных. Пусть, далее, оператор ***СКАН*** передвигает указатель по массиву слева направо до обнаружения первой неупорядоченной пары элементов массива или до установки указателя в конец массива (сканирует массив), т.е.

$$СКАН = \{[d(У,К) \vee (l > r, У)]Р(У)\}.$$

Тогда предикат

$$U(M) = \begin{cases} 1, \text{если } M - \text{упорядоченный}; \\ 0, \text{если } M \text{ неупорядоченный}. \end{cases}$$

может быть представлен с помощью операции прогнозирования $U = СКАН \bullet d(У,К)$

Используя это выражение для U в СС алгоритма «Пузырек», получим интерпретированную регулярную схему этого алгоритма в алгебре Глушкова:

«Пузырек» $:=$

$$= \{[\, СКАН \bullet d(У,К)]\{[d(У,К)]([(l > r, У)]ТРАНСП(l,r,У), Е) * П(У)\} * Уст(У,Н)\}.$$

Алгоритм является результатом интерпретации регулярной схемы (операторной формулы) алгебры Глушкова, а именно

$$РС = \{[A_1 \bullet u_1]\{[u_2]([u_3]A_2, A_3) * A_4\} * A_5\}$$

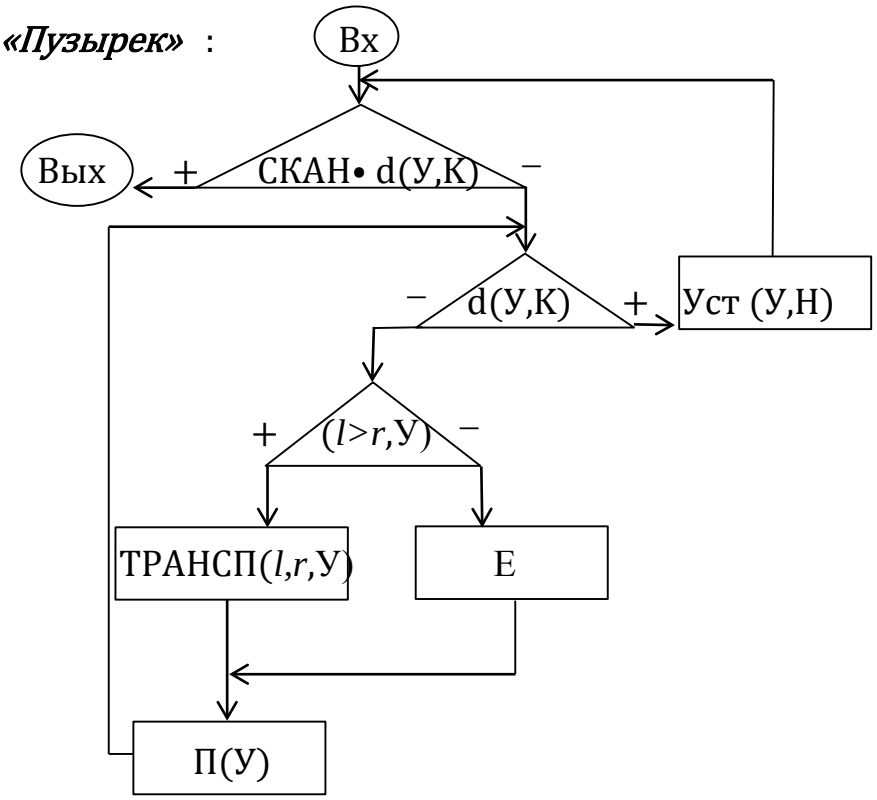
с использованием алгебраической системы

$$АС = \langle\, MASS; \{СКАН, Уст(У,Н), ТРАНСП(l,r,У), П(У), Е\} \cup$$

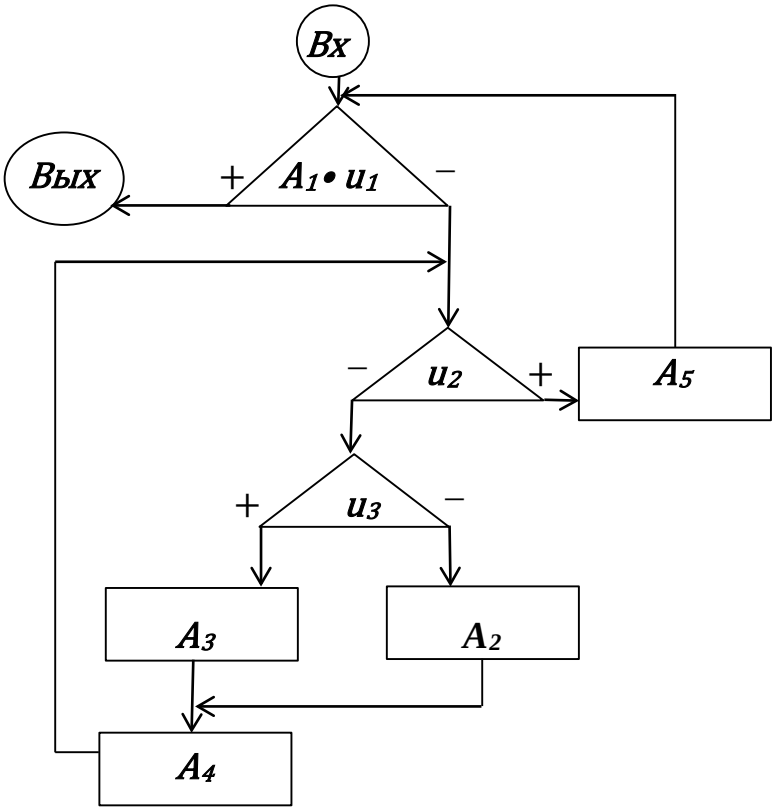
$$\cup \{d(У,К), (l > r, У)\}\, \rangle.$$

И, в заключение, приведем обобщенную граф-схему (интерпретированную и нет) алгоритма «Пузырек».

«Пузырек» :



Неинтерпретированная обобщенная граф-схема алгоритма «Пузырек»



Как уже отмечалось ранее, алгебра Дэйкстры и алгебра Глушкова в наше время являются наиболее развитыми и наиболее употребительными в алгоритмике. Имеет место

Тезис Глушкова. *Схема любого алгоритма может быть представлена подходящей регулярной схемой в алгебре Глушкова. Иначе,* ***любой*** *алгоритм может быть представлен подходящей интерпретированной регулярной схемой в алгебре Глушкова.*

Тезис Глушкова, как и тезисы Тьюринга, Чёрча, Маркова в других системах, уточняющих интуитивное понятие алгоритма, не является строгой математической теоремой, но его справедливость подтверждается всей программистской практикой вплоть до настоящего времени.

В силу определения обобщенной граф-схемы ясно, что всякая регулярная схема (операторная формула АГ) может быть представлена подходящей обобщенной граф-схемой. Поэтому можно рассмотреть алгебру обобщенных граф-схем по аналогии с алгеброй Калужнина и сформулировать тезис Глушкова на языке обобщенных граф-схем.

Литература.

1. Андерсон Джеймс А. Дискретная математика и комбинаторика: Пер. с англ. - М.: Издательский дом «Вильямс», 2003.

2. Белоусов А.И., Ткачев С.Б. Дискретная математика: учебник для ву зов. - М. Издательство МГТУ им. Н.Э. Баумана, 2004.

3. Воеводин В.В. Линейная алгебра. М.: Наука, 1980.

4. Цейтлин Г.Е. Введение в алгоритмику. К.: Сфера, 1998.