ПРОГРАММНО-
ТЕХНИЧЕСКИЕ
КОМПЛЕКСЫ

М.С. ЛЬВОВ

УДК 004.421.6

АЛГЕБРАИЧЕСКИЙ ПОДХОД К ЗАДАЧЕ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ НЕРАВЕНСТВ

Ключевые слова: *многосортные алгебры, системы линейных неравенств, алгебраическое программирование, алгебраические спецификации*.

В настоящей работе изложен алгебраический поход к построению алгоритма решения системы линейных неравенств (СЛН). Отметим, что многосортные (точнее, упорядоченосортные) алгебраические системы широко используются в качестве теоретической модели алгебраических вычислений [1, 2]. Эта модель реализована, например, в языках логико-алгебраических спецификаций серии Obj [3, 4], Eqlog [5].

Классические абстрактные алгебраические типы данных вместе с их представлениями в стандартных структурах данных (т.е. конструктивными представлениями) используются, например, в языке программирования системы компьютерной алгебры Axiom [6, 7].

Методология алгебраического программирования, использованная в настоящей работе, заключается в конструктивном определении многосортной алгебраической системы, объекты которой представлены в виде выражений, а алгоритмы, по существу, вычисляют значения этих выражений, т.е. строят их канонические формы. Результат применения этого подхода — алгебраические спецификации соответствующих предметных областей и алгоритмов. Спецификации используют понятия наследования, расширения и морфизмов многосортных алгебр. Подробно этот подход изложен в [8–11].

Отметим, что эта методология применима к широкому классу предметных областей, она позволяет хорошо структурировать алгебраические программы, а также решать задачи синтеза и верификации алгебраических программ. Например, в [12] этот подход применяется к такой предметной области, как алгебра высказываний. Подчеркнем, что мы специфицируем собственно предметные области, а не только алгоритмы.

В частности, описываемый алгоритм решения СЛН строит каноническую форму этой системы. Кроме того, в работе намечены и другие алгоритмы: алгоритм сведения кратного интеграла по многогранной области к повторным интегралам, алгоритм изменения порядка интегрирования, а также алгоритм решения задачи линейного программирования (поиск максимума линейной функции на многогранной области).

Каноническая форма использует идею проектирования (элиминации квантора) методов Фурье–Моцкина [13, 14], Черникова [15].

1. СИСТЕМЫ ЛИНЕЙНЫХ НЕРАВЕНСТВ. ОБЩИЕ СВЕДЕНИЯ

Линейное неравенство (ЛН) L имеет вид

$$L = a_1 x_1 + a_2 x_2 + \ldots + a_m x_m \le b,\ x_j \in \text{Variable},\ a_j, b \in \text{Coef},$$
$$A = (a_1, \ldots, a_m),\ X = (x_1, \ldots, x_m),\ L = A \cdot X \le b.$$

Таким образом, алгебра ЛН использует линейное пространство LinComb над упорядоченным полем Coef, а также алгебру переменных Variable.

СЛН задаются логическими формулами вида $S = L_1 \& L_2 \& \ldots \& L_n$, где L_j — линейные неравенства.

Основные задачи теории СЛН:

1) проверить совместность (существование решения);

2) привести к каноническому виду (решить); решение — выпуклый многогранник в n-мерном пространстве, таким образом, каноническая форма должна описывать многогранник решений, например, через вершины или уравнения граней.

Особая нормальная форма ЛН и СЛН — x-разрешенная форма. Пусть x — переменная, входящая в левую часть ЛН с ненулевым коэффициентом. Тогда ЛН можно преобразовать к виду

$$x \le B \cdot X + b \text{ или } x \ge B \cdot X + b. \tag{1}$$

Такую форму ЛН будем называть разрешенной относительно x, или x-разрешенной. Если каждое ЛН системы, зависящее от x, представлено в разрешенной форме, то такую СЛН будем называть представленной в x-разрешенной форме. Любую СЛН можно представить в x-разрешенной форме. Точнее, в виде $LL(x)\&GL(x)\&FL$, где каждое неравенство $LL(x)$ имеет вид $x \le B \cdot X + b$, каждое неравенство $GL(x)$ имеет вид $x \ge B \cdot X + b$, каждое неравенство FL не зависит от x.

Известный алгоритм Фурье–Моцкина проверки совместности СЛН [13, 14] и его усовершенствование — метод Черникова [15] — основан на следующем факте: СЛН $(x \le B_1 X + b_1)\&(x \ge B_2 X + b_2)$ совместна тогда и только тогда, когда совместно ЛН $B_2 X + b_2 \le B_1 X + b_1$. В теоретико-логической формулировке преобразование имеет вид

$$\exists X\ \exists x(x \le B_1 \cdot X + b_1)\&(x \ge B_2 \cdot X + b) \sim \exists X(B_1 \cdot X + b_1 \ge B_2 \cdot X + b_2). \tag{2}$$

По существу оно элиминирует квантор $\exists x$. Геометрически преобразование заключается в проектировании области решений на подпространство X.

Пусть дано два ЛН одного знака: $L_1 = x \le B_1 \cdot X + b_1$, $L_2 = x \le B_2 \cdot X + b_2$. Тогда множество решений $L_1 \& L_2$ описывается формулой

$$L_1 \& L_2 = \{x : x \le \min(B_1 \cdot X + b_1, B_2 \cdot X + b_2)\}. \tag{3}$$

Аналогично для ЛН противоположного знака получаем

$$G_1 \& G_2 = \{x : x \ge \max(B_1 \cdot X + b_1, B_2 \cdot X + b_2)\}. \tag{4}$$

Множество решений СЛН противоположных знаков $L\&G$ на числовом множестве S оси Ox обозначим $R(x, S, L, G)$. Тогда

$$R(x, S, L_1, G_1) \cap R(x, S, L_2, G_2) = R(x, S, \min(L_1, L_2), \max(G_1, G_2)). \tag{5}$$

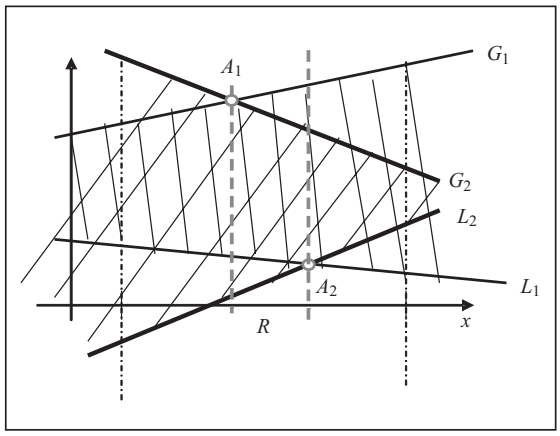

Рис. 1

На рис. 1 показано, что числовой промежуток проекциями точек пересечения $A_1 = L_1 \cap L_2$, $A_2 = R_1 \cap R_2$ разбивается на три части. На каждой из них множество решений описывается формулой $R(x, S, L, G)$.

2. ЧАСТНЫЙ СЛУЧАЙ: РЕШЕНИЕ СЛН НА ПЛОСКОСТИ

Задача построения области решения СЛН достаточно известна. В частном случае, когда $n = 2$, эту задачу можно переформулировать как задачу построения выпуклой линейной оболочки по системе ЛН, задающих стороны многоугольника решения — одну из основных задач вычислительной геометрии и компьютерной графики [16, 17]. В математических системах учебного назначения эту задачу можно сформулировать как задачу реализации так называемого графического метода в линейном программировании [18]. Основная идея метода проиллюстрирована рис. 1–3. Предположим, что многоугольник решения R, каждая сторона которого задана ЛН, разбит на области-трапеции R_1, R_2, R_3, R_4. $R = R_1 + + R_2 + + R_3 + + R_4$ (см. рис. 2). Добавление нового ЛН в СЛН заключается в том, что R_2 и R_4 разбиваются на две части, а у R_3 изменяется верхняя сторона (шапка):

$$R_2 = R_2' + + R_2'',\ R_4 = R_4' + + R_4'',\ R_3 = R_3^1. \tag{6}$$

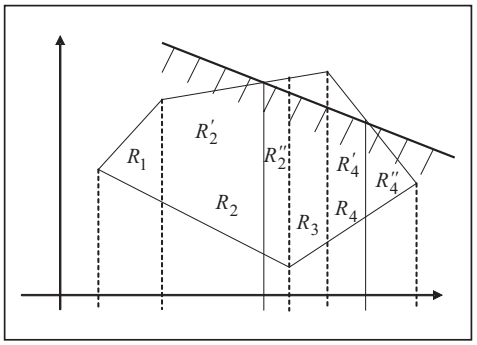

Рис. 2

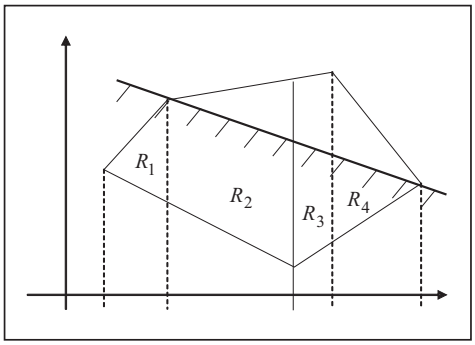

Рис. 3

Уточним, что равенства (6) интерпретированы как переписывающие правила. В результате преобразования получим

$$R' = R_1 + + R_2' + + R_2'' + + R_3' + + R_4' + + R_4''. \tag{7}$$

Возможен также вариант, при котором добавление нового неравенства приводит к слиянию нескольких областей в одну (см. рис. 3).

Таким образом, алгоритм последовательно добавляет к построенному многоугольнику новое полупространство, пересекая его с многоугольником. Отметим, что возможны и другие варианты управления вычислениями: например, можно применить метод «разделяй и властвуй». В этом варианте алгоритма СЛН разбивается на две подсистемы, содержащие примерно равное число неравенств. Рекурсивное применение основной процедуры строит два многоугольника, которые затем нужно пересечь, получив решение всей СЛН. Общий алгоритм вполне аналогичен алгоритму в этом частном случае.

3. КОНСТРУКТИВНАЯ АЛГЕБРА СЛН (АЛГЕБРА ВЫПУКЛЫХ МНОГОГРАННИКОВ)

Суть данного подхода к задаче решения СЛН состоит в следующем. Во-первых, определим такую исходную многосортную алгебру СЛН A_{init}, что СЛН описывается выражением в сигнатуре A_{init}. Во-вторых, определим специальную многосортную конструктивную алгебру СЛН A_{constr}. Ее конструктивность заключается в том, что в ее описание включены так называемые конструкторы элементов и интерпретаторы операций в виде систем переписывающих правил. В-третьих, определим изоморфизмы $\mu : A_{\text{init}} \to A_{\text{constr}}$, $\mu^{-1} : A_{\text{constr}} \to A_{\text{init}}$. Вычисления значения $\text{Val}(F(X))$ выражения $F(X)$ строятся по схеме

$$\text{Val}(F(X)) = \mu^{-1}(\text{Can}(F(\mu(X)))). \tag{8}$$

Подробное описание общего подхода изложено в [8–11].

Приведем краткое описание алгебр A_{init} и A_{constr}.

1. Многосортная алгебра СЛН A_{init} использует следующие сорта:

- Coef — упорядоченное числовое поле;
- Variable — линейно-упорядоченное множество переменных;
- LinComb — векторное пространство аффинных линейных комбинаций;
- LinUnEqu — алгебра линейных неравенств;
- SysLinUnEqu — алгебра систем линейных неравенств.

2. Для описания конструктивной алгебры СЛН A_{constr} нам понадобятся сорта, описанные ниже. Сорта Coef и Variable (базовые сорта) должны быть реализованны в системе алгебраического программирования.

3.1. Сорт LinUnEqu. Элементами этого сорта являются ЛН, представленные в x-разрешенной форме, где x — старшая переменная ЛН:

$$x_n \le a_{n-1} \cdot x_{n-1} + \ldots + a_1 \cdot x_1 + b_1 \quad \text{или} \quad x_n \ge a_{n-1} \cdot x_{n-1} + \ldots + a_1 \cdot x_1 + b_1. \tag{9}$$

Специальная операция $X\,\text{Can}(L)$ приводит произвольное выражение $L = B_1 \cdot X + b_1 \le$ $\le B_2 \cdot X + b_2$ в сигнатуре алгебры ЛН к канонической форме [9]. Селекторы сорта выделяют имя переменной, знак неравенства ($\le$ или $\ge$) и его правую часть.

3.2. Сорт Interval — это сорт числовых промежутков на расширенной числовой оси. Расширение числовой оси осуществляется добавлением к упорядоченному числовому полю Coef символов $-\text{Inf}$, $+\text{Inf}$ (минус бесконечность, плюс бесконечность). Расширенную числовую ось будем называть ExtCoef.

В алгебре Interval числовых промежутков на расширенной числовой оси определены аннулирующий элемент — пустой отрезок O и нейтральный элемент $I = [-\text{Inf}, +\text{Inf}]$. По определению для любого элемента $c \in \text{Coef}$ имеет место двойное неравенство $-\text{Inf} < c < +\text{Inf}$. Относительно операции пересечения алгебра Interval образует идемпотентную коммутативную полугруппу с нулем и единицей.

Кроме операции пересечения отрезков, обозначаемой «&», для сорта Interval определена операция разбиения отрезка:

$$\text{Partition}\,([a, b], c) = [a, c] + + [c, b]. \tag{10}$$

Результат этой операции определен в некотором расширении сорта Interval. Обратная операция — операция сложения отрезков — определена соотношением

$$[a, c] + + [c, b] = [a, b]. \tag{11}$$

3.3. Сорт VarSegment. Это сорт элементарных многогранников пространства решений СЛН. Элемент этого сорта описывается конструктором

$$S = [L(Y), x, G(Y)] \tag{12}$$

с семантикой

$$L(Y) \le x \le G(Y), \tag{13}$$

где x — переменная, $Y = \{x_1, \ldots, x_k\}$, $L(Y)$, $G(Y)$ — линейные комбинации переменных из Variable с коэффициентами из Coef, т.е. элементами сорта $\text{LinComb}\,(Y, \text{Coef})$. Элементы сорта VarSegment назовем x-сегментами, или просто сегментами. Сорт Interval можно рассматривать как частный случай сорта VarSegment.

3.4. Сорт Trapezoid — это сорт элементарных трапецоидов в пространстве решений СЛН. Для двумерного пространства $W(x, y)$ элемент этого сорта задан конструкцией $T = [l, x, g].[L(x), y, Q(x)]$ с семантикой $T = (l \le x \le g) \& (L(x) \le y \le Q(x))$.

Для n-мерного пространства $W(X)$, где $X=\{x_1,\ldots,x_n\}$, элемент этого сорта задан конструкцией

$$T=[L_n(X_{n-1}),x_n,G_n(X_{n-1})].\ldots.[L_2(x_1),x_2,G_2(x_1)].[L_1,x_1,G_1], \qquad (14)$$

где $X_{k-1}=(x_1,\ldots,x_{k-1})$, с семантикой

$$T=(L_n(X_{n-1})\le x_n\le G_n(X_{n-1})\&\ldots\&(l_2(x_1)\le x_2\le g_2(x_1))\&(L_1\le x_1\le G_1). \qquad (15)$$

Точка «.» — отметка конструктора сорта Trapezoid. Допускается использование и других обозначений:

$$T=S_x\cdot S_y,\ S_x=[l,x,g],\ S_y=[L,y,Q],\ T=S_l^g(x)\cdot S_L^Q(y).$$

Замечание 1. Трапецоид — выпуклая фигура в n-мерном пространстве, последовательные проекции которой на подпространства меньшей размерности — также трапецоиды (рис. 4). Уравнения $x_n=L(X_{n-1})$, $x_n=G(X_{n-1})$ задают нижнюю и верхнюю «шапки» трапецоида, если числовую ось Ox_n рассматривать как вертикальную. Таким образом, если трапецоид T задан формулой $T=S.T_1$, x_n— отрезок, S определяет T по координате x_n, а T_1 — проекция T на подпространство $W(X_n-1)$. ♦

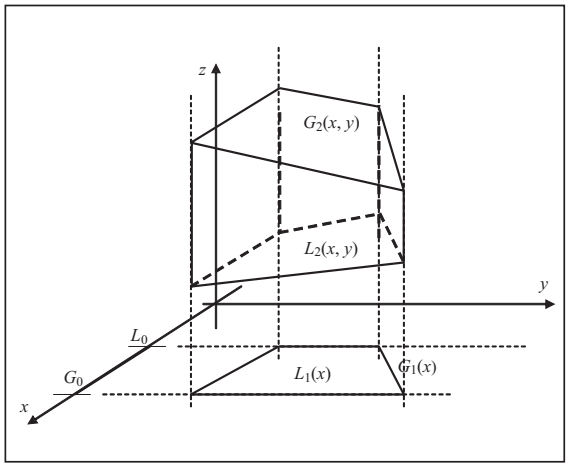

Рис. 4

Точка — отметка конструктора сорта Trapezoid — интерпретируется как операция пересечения, если ввести в рассмотрение единицу полугруппы — бесконечный сегмент $I(x)=[-\mathrm{Inf},x,+\mathrm{Inf}]$. Тогда для $T=S_1(y).S_2(x)$, $T_1=S_1(y).I(x)$, $T_2=I(y).S_2(x)$ справедливо соотношение $T=T_1\&T_2$.

Таким образом, алгебра трапецоидов TR представлена в виде прямого предела возрастающей последовательности алгебр:

$$\mathrm{TR}(x_1)\subset\mathrm{TR}(x_1,x_2)\subset\ldots\subset\mathrm{TR}(x_1,\ldots,x_n)\subset\ldots, \qquad (16)$$

$$\mathrm{TR}=\bigcup_{j=1}^{\infty}\mathrm{TR}(X_j),\ X_j=\{x_1,\ldots,x_j\}\subset\mathrm{Variable}. \qquad (17)$$

В терминологии [8] TR — линейное динамическое расширение алгебры VarSegment.

Кроме единицы, в алгебре TR есть еще и ноль (аннулятор) — пустой числовой промежуток. Нулем является элемент сорта VarSegment вида $[a,x,b]$, $a,b\in\mathrm{Coef}$, $a>b$. Таким образом, имеют место соотношения

$$(a\in\mathrm{Coef})\&(b\in\mathrm{Coef})\&(a>b)\to[a,x,b]=O, \qquad (18)$$

$$S.O=O.$$

Кроме этого, свойствами нулевого элемента операции сложения сегментов обладает сегмент вида $[a,x,a]$, точнее, имеют место соотношения

$$[L,x,G]++[G,x,G]=[L,x,G],\ [L,x,L]++[L,x,G]=[L,x,G].$$

4. СОРТ CONPOL

Сорт ConPol (Convex Polygon) — это сорт выпуклых многогранников в пространстве решений СЛН. В n-мерном пространстве элемент этого сорта (выпуклый многогранник) задан конструкцией

$$P = T_1 + + T_2 + + \ldots + + T_k . \tag{19}$$

Замечание 2. Отметка «+ +» конструктора сорта отмечает разбиение выпуклого многогранника на (непересекающиеся) трапецоиды. Линейный порядок, зафиксированный на множестве переменных Variable, определяет последовательность проектирований выпуклого многогранника P на подпространства все меньшей и меньшей размерности. Основная идея данной конструкции исходит из следующей задачи.

Пусть G — многогранная область n-мерного пространства, заданная СЛН, необходимо вычислить (кратный) интеграл $\int\limits_G f(X)dX$. При переходе от кратного интеграла к повторным нужно определить порядок и пределы интегрирования. Отметим, что пределы интегрирования должны быть аналитическими выражениями, т.е. линейными комбинациями переменных. Таким образом, искомый кратный интеграл должен быть представлен в виде суммы повторных интегралов, в каждом из которых пределы интегрирования определяют именно трапецоиды:

$$\int\limits_G f(X)dX = \sum_j \int\limits_{T_j} f(X)dX ., \quad \int\limits_{T_j} f(X)dX = \int\limits_{L_n}^{G_n G_{n-1}} \int\limits_{L_{n-1}} \ldots \int\limits_{l_1}^{g_1} f(x_1, \ldots, x_n)\, dx_1\, dx_2 \ldots dx_n .$$

Ясно, что решение этой задачи существует, и если установить порядок интегрирования и потребовать минимальности количества разбиений, такое представление области единственно с точностью до коммутативности. ♦

Основная операции сорта ConPol — операция пересечения двух выпуклых многогранников. Еще одна операция — операция сложения/разбиения (разбиение множества на два непересекающихся подмножества или объединение). Сорт ConPol является линейным динамическим расширением сорта Trapezoid, поэтому эти операции определяются через соответствующие операции сорта Trapezoid. Сорт Trapezoid, в свою очередь, является расширением сорта VarSegment, поэтому эти операции, в свою очередь, определяются через соответствующие операции сорта VarSegment. Отметим, что операция разбиения определена частично.

4.1. Операция пересечения на сорте ConPol. Эффективность алгоритма в целом зависит от эффективности реализации операции пересечения на сорте ConPol. Символ операции пересечения — знак конъюнкции &. Эта операция задана на сорте ConPol через ее определения на базовых сортах VarSegment и Trapezoid. На сорте VarSegment пересечение определено формулами

$$[L_1(X),\, y, G_1(X)]\&[L_2\,(X),\, y, G_2(X)] = [\max(L_1, L_2),\, y,\, \min(G_1, G_2)]. \tag{20}$$

При этом результат операции должен удовлетворять соотношению

$$\max(L_1, L_2) \le \min(G_1, G_2). \tag{21}$$

Заметим, что $\max(L_1, L_2)$, $\min(G_1, G_2)$ не принадлежат базовому сорту LinComb. Поэтому операцию пересечения нужно определить более точно. Следующие рисунки иллюстрируют всевозможные варианты взаимного расположения границ сегментов для случая $n = 2$, при котором рассматриваемые многогранники — суть трапеции. Во-первых, ограничимся рассмотрением пересечения трапеции с полуплоскостью, заданной неравенством $y \le G(X)$. Вполне аналогично рассматривается пересечение трапеции с полуплоскостью $y \ge L(X)$. Во-вторых, будем считать, что пересекаемый полуплоскостью сегмент $S_1 = [L_1(X),\, y, G_1(X)]$ определен на трапецоиде T подпространства $W(x_{n-1}) = \{u : u = (x_1, \ldots, x_{n-1})\}$. Операция пересече-

ния определена на элементах $S_1.T$, $S.T$, причем $L_1(X) \underset{X \in T}{\leq} G_1(X)$. Для случая $n = 2$ трапецоид T — отрезок числовой оси $T = [L_0, x, G_0]$.

Логически возможно шесть вариантов взаимного расположения трапецоида и полуплоскости, которая его пересекает (рис. 5).

Вариант 1 (рис. 5, *а*): $G \geq_T G_1 \Rightarrow S_1.T \cap S = S_1.T$.

Вариант 2 (рис. 5, *б*) : $(G \& G_1 \neq_T \varnothing) \& (G \geq_T L_1) \Rightarrow S_1.T \cap S = [L_1, y, G_1].T_1 + [L_1, y, G].T_2$, $T_1 = \{X : G_1 \leq_T G\}$, $T_2 = T - T_1$.

Вариант 3 (рис. 5, *в*): $(G \& G_1 \neq_T \varnothing) \& (G \& L_1 \neq_T \varnothing) \Rightarrow S_1.T \cap S = [L_1, y, G_1].T_1 + [L_1, y, G].T_2$, $T_1 = \{X : G_1 \leq_T G\}$, $T_2 = \{X : G \leq_T G_1\}$.

Вариант 4 (рис. 5, *г*): $(G \leq_T L_1) \& (G \leq_T G_1) \Rightarrow S_1.T \cap S = [G, y, L_1].T$.

Вариант 5 (рис. 5, *д*): $(G \leq_T L_1) \& (G \& L_1 \neq_T \varnothing) \Rightarrow S_1.T \cap S = [L_1, y, G].T_1$, $T_1 = \{X : L_1 \leq_T G\}$.

Вариант 6 (рис. 5, *е*): $G \leq_T G_1 \Rightarrow S_1.T \& S = \varnothing$.

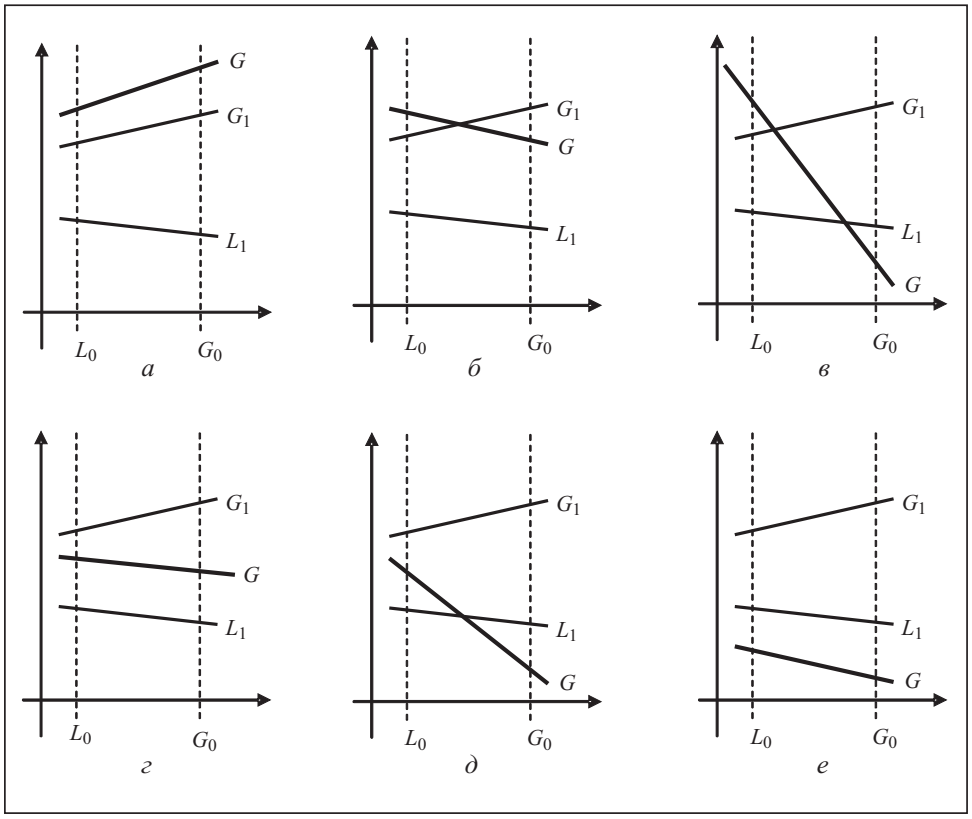

Рис. 5

Основной источник неэффективности алгоритма, описываемого соотношениями этих вариантов, — наличие двух слагаемых в правых частях соотношений и необходимость вычисления новых трапецоидов-проекций: T_1 и T_2. В варианте 6 количество трапецоидов в пересечении уменьшается, в вариантах 1, 4 не изменяется, причем проекции остаются неизменными, в варианте 5 их количество не изменяется, но изменяется проекция. Худшие случаи — варианты 2, 3, в которых количество трапецоидов увеличивается и проекции изменяются.

4.2. Алгоритм распознавания варианта. Все соотношения, определенные в вариантах, условные. Условия определены в терминах отношения частичного по-

рядка «$\le_T$». $L \le_T G$ означает, что трапеция $y = L(X)$ расположена ниже трапеции $y = G(X)$ на области T, точнее, на T выполняется неравенство $G(X) - L(X) \ge 0$. В свою очередь, это означает, что

$$y_{\max} = \max_{X \in T} (L(X) - G(X)) \le 0. \qquad (22)$$

Из того, что функция $y = L(X) - G(X)$ линейна и что трапецоид T — многогранная область, следует, что максимум достигается в одной из вершин трапецоида T. Наконец, форма представления трапецоида в виде последовательности его проекций на подпространства все меньшей размерности позволяет вычислить искомый максимум достаточно эффективно — за время $O(n^2)$. Покажем это.

1. Приведем трапецоид $T_n = S_1 . S^2 . \ldots . S_n$ к виду $S_i = [0, x_i, g_i(X_{i-1})]$ преобразованием

$$S_i = [l_i(X_{i-1}), x_i, g_i(X_{i-1})] \Rightarrow [0, x_i - l_i(X_{i-1}), g_i(X_{i-1}) - l_i(X_{i-1})] \qquad (23)$$

и последующей заменой переменных $x_i = x_i - l_i(x_{i-1})$. Эти преобразования выполняются на трапецоиде за время $O(n^2)$.

2. Рассмотрим следующую задачу линейного программирования (ЗЛП): найти $\max F$ линейной функции $F = c_1 x_1 + \ldots + c_{n-1} x_{n-1} + c_n x_n$ на приведенном трапецоиде T_n. Поскольку решение ЗЛП достигается на границах трапецоида T_n, рассмотрим два случая:

• $x_n = 0$, ЗЛП сводится к задаче размерности $n-1$: на трапецоиде $T_{n-1} = S_1 . S_2 . \ldots . S_{n-1}$ найти $\max F_1$ функции

$$F_1 = c_1 x_1 + \ldots + c_{n-1} x_{n-1}. \qquad (24)$$

• $x_n = g_n(x_{n-1})$, ЗЛП сводится к следующей ЗЛП размерности $n-1$: на трапецоиде $T_{n-1} = S_1 . S_2 . \ldots . S_{n-1}$ найти $\max F_2$ функции

$$F_2 = c_1 x_1 + \ldots + c_{n-1} x_{n-1} + c_n g_n(x_{n-1}). \qquad (25)$$

Таким образом, $\max F = \max(\max F_1, \max F_2)$.

Поскольку на T_{n-1} $g_n(X_{n-1}) \ge 0$, имеем несколько случаев:

а) $c_n < 0$: $\max(\max F_1, \max F_2) = \max(\max F_1, F_1 - g) = \max F_1$, $g \ge 0$;

б) $c_n \ge 0$: $\max(\max F_1, \max F_{2)} = \max(\max F_1, F_2) = \max F_2$, $g \ge 0$.

Итак, ЗЛП размерности n на трапецоиде сводится к задаче размерности $n-1$ на трапецоиде за время $O(n)$.

Эти соотношения следует дополнить частными случаями, которые определяются тем, что сорт ExtCoef — расширение поля Coef элементами $-\mathrm{Inf}$, $+\mathrm{Inf}$. Если сегмент S имеет вид $S = [-\mathrm{Inf}, x_n, G(x_{n-1})]$, его нельзя привести к виду (23). Однако при $c_n < 0$ $\max F = +\mathrm{Inf}$. При $c_n > 0$, по существу, имеет место случай б) и ЗЛП решается формулой (25). Аналогично рассматривается случай, когда неограниченный сегмент задается формулой $S = [L(x_{n-1}), x_n, +\mathrm{Inf}]$. Для реализации вычислений в этих случаях определяется сорт ExtCoef. Вычисления в алгебре ExtCoef определяются следующими соотношениями:

$a > 0 \to a * (+\mathrm{Inf}) = +\mathrm{Inf}$,

$a > 0 \to a * (-\mathrm{Inf}) = -\mathrm{Inf}$,

$a < 0 \to a * (+\mathrm{Inf}) = -\mathrm{Inf}$,

$a < 0 \to a * (-\mathrm{Inf}) = +\mathrm{Inf}$,

$a + +\mathrm{Inf} = +\mathrm{Inf}$,

$a + -\mathrm{Inf} = -\mathrm{Inf}$,

$a - +\mathrm{Inf} = -\mathrm{Inf}$,

$a - -\mathrm{Inf} = +\mathrm{Inf}$.

Следовательно, ЗЛП размерности n решается на трапецоиде за время $O(n^2)$.

Замечание 3. Описанный алгоритм решает ЗЛП на трапецоиде. Из формул (27), (28) следует, что многогранная область представляется в виде разбиения на трапецоиды, поэтому ЗЛП может быть решена этим методом на произвольной многогранной области. ♦

4.3. Канонические представления элементов сорта ConPol. Как уже отмечалось, выпуклый многогранник — элемент сорта ConPol, представим в виде разбиения его на трапецоиды. Таким образом,

$$P = T_1 + + T_2 + + \ldots + + T_k . \tag{26}$$

На множестве трапецоидов естественным образом может быть задано отношение линейного порядка, индуцированное отношениями порядка на сортах Coef, Variable и расширенное сначала на сорт Trapezoid, а затем и на сорт ConPol. Таким образом, можно считать, что $T_1 > T_2 > \ldots > T_k$, $T_j = S_{jn} . S_{jn-1} . \ldots . S_{j1}$.

Применяя соотношения $S . O = O$, $S . I = S$, $T + + O = T$, $O + + T = T$, можно избавиться от «лишних» слагаемых и сомножителей. Тогда элемент P представим в канонической форме

$$P = \sum_{j=1}^{k} \prod_{i=1}^{n} S_{ji}, \tag{27}$$

например, $P = S_{13} . S_{12}$, $S_{11} + + S_{23} . S_{22} . S_{21} + + S_{33} . S_{32} . S_{31} + + S_{43} . S_{42} . S_{41}$.

Таким образом, сорт ConPol — линейное динамическое расширение сорта Trapezoid. Здесь предикатные переменные S_{jk} интерпретированы в алгебре VarSegment.

Легко показать, что множество выражений такого вида удовлетворяет всем свойствам (аксиомам) кольца многочленов Жегалкина в сигнатуре операций-отметок $< + +, ., O, I >$, за исключением свойства коммутативности умножения. Поэтому многогранники можно называть полиномами, трапецоиды — мономами, а сегменты — переменными.

Замечание 4. Перестановка сомножителей в произведениях $S_{jn} . S_{jn-1} . \ldots . S_{j1}$ — это задача изменения порядка интегрирования (см. замечание 2). Эта задача представляет самостоятельный интерес, причем не только для интегралов по выпуклым многогранникам, но и для областей, заданных системами выпуклых неравенств. ♦

Поскольку в конструктивной алгебре ConPol имеет место закон дистрибутивности, наряду с представлением многочлена P в виде суммы мономов T_j (26) можно определить еще одну каноническую форму — так называемое рекурсивное представление P.

Индексом (размерности) переменной — элемента $S = [L, y, G]$, назовем порядковый номер переменной y в сорте Variable, индексом элемента (монома) T — индекс его старшего элемента, наконец, индексом полинома — индекс его старшего монома.

Пусть в (27) несколько слагаемых имеют вид $S . T_1, \ldots, S . T_k$, где S — переменная максимального индекса. Тогда S можно вынести за скобки. Осуществив это преобразование для всех переменных S_j максимального индекса, из (27) получим

$$P = S_1 . P_1 + + S_2 . P_2 + \ldots + S_k . P_k . \tag{28}$$

В (28), индексы полиномов P_j меньше индексов переменных S_j. Применяя рекурсивно описанное преобразование ко всем полиномам все меньшего индекса, получаем рекурсивную каноническую форму элемента алгебры ConPol, рассматриваемого как многочлен Жегалкина.

Следующее преобразование, основанное на свойстве сложения (11), назовем приведением подобных. Пусть $P = S_1 . P_1 + + S_2 . P_1$ и $S_1 = [A, x, C]$, $S_2 = [C, x, B]$. Тогда $P = [A, x, B] . P_1$. Таким образом, приведение подобных описывается равенством

$$[A, x, C] . T + + [C, x, B] . T = [A, x, B] . T. \tag{29}$$

Рекурсивной нормальной формой (РНФ) полигона — элемента алгебры ConPol, будем называть форму (28), в которой приведены все подобные слагаемые по правилу (29). Понятно, что РНФ представляет полигон экономнее, чем полиномиальная нормальная форма (ПНФ) (27).

В заключение отметим, что можно было бы рассматривать и другие канонические формы представления полигонов. Например, интересно представлять полигоны как пересечения «нижних» и «верхних» шапок. Возможно, такое представление будет эффективнее, чем РНФ.

4.4. Изоморфизмы инициального и конструктивного представлений алгебры СЛН. Алгоритм решения СЛН начинает свою работу с преобразований каждого линейного неравенства системы в элементарный сегмент (атом). Соответствующие правила имеют вид

$$\mu\,(y \le G) = [-\mathrm{Inf},\, y, G],\ \ \mu\,(y \ge L) = [L,\, y,\, +\mathrm{Inf}]. \tag{30}$$

Эти преобразования определяют изоморфизм алгебры $\mu : A_{\mathrm{init}} \to A_{\mathrm{constr}}$. В реализации имеет смысл совместить эти преобразования с вычислениями x-разрешенной формы ЛН.

По сути дела, полигон P, представленный в РНФ (28), описывает решение СЛН. Однако, если необходимо представить только оболочку решения в n-мерном пространстве, игнорируя проекции в пространства меньшей размерности, это можно сделать с помощью обратного изоморфизма $\mu^{-1}: A_{\mathrm{constr}} \to A_{\mathrm{init}}$. Соответствующее основное правило имеют вид

$$\mu^{-1}([L, x, G].P + +Q) = (x \le G) \& (x \ge L) \& \mu^{-1}(Q). \tag{31}$$

5. ОБЩЕЕ УПРАВЛЕНИЕ И СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ

По-видимому, наиболее простой подход к управлению вычислениями заключается в последовательном добавлении к уже построенной РНФ нового неравенства

$$S \,\&\, P = S \,\&\, (S_1 .P_1 + + S_2 .P_2 + + \ldots$$

$$\ldots + + S_k .P_k) = (S \,\&\, S_1).P_1 + \ldots + (S \,\&\, S_k).P_k. \tag{32}$$

Итак, алгоритм перевычисляет неравенства, определяющие трапецоид, за время $O(n^2)$. Пусть $C(m, n)$ — количество трапецоидов (мономов) в текущем представлении (26). Тогда на перевычисление полинома (27) тратится $O(n^2 C(m, n))$ шагов. Поскольку алгоритм добавления неравенства повторяется m раз,

$$T(m, n) = O(mn^2 C(m, n)). \tag{33}$$

Нетрудно показать, что в разбиении многогранной области P на трапецоиды каждый трапецоид можно ассоциировать, по крайней мере, с одной (своей) вершиной P. Иначе трапецоид можно было бы «расширить», увеличив один из его сегментов. Поэтому количество трапецоидов в P не превосходит количества V его вершин (0-граней). При этом необходимо учитывать и бесконечно удаленные вершины, одна из координат которых равна $\pm\mathrm{Inf}$. Однако даже один трапецоид, который определяется $2n$ неравенствами (по две на каждую переменную), содержит 2^n вершин, поэтому оценку метода

$$T(m, n) = O(mn^2 V(m, n)), \tag{34}$$

в которой $V(m, n)$ — количество вершин P [19], следует считать экспоненциальной.

```
Sort ExtCoef::LinOrdField;
Constructor {
ExtCoef a = Coef a | Const -Inf | Const +Inf;
};
// Определения операций и отношений порядка на константах –Inf, +Inf
Operations
Mult:{
  a > 0 - a*(+Inf = +Inf,
  a > 0 - a*(-Inf = -Inf,
  a < 0 - a*(+Inf = -Inf,
  a < 0 - a*(-Inf = +Inf
};
...
};
Sort VarSegment::Interval;
Constructor {
  VarSegment S = [LinComb L,Variable x, LinComb G];
// Селекторы
  LowGran(S) = L,
  UpGran(S) = G,
  Var(S) = x;
// Контекстные условия
  L < = G,
  x < LeadVar(L),
  x < LeadVar(G);
// Редукции
[-Inf,x,+Inf] = I,
(a∈Coef)&(b∈Coef)&(ab)-> [a,x,b] = O;                    };
Signature
  XCan(1):LinUnEqu -> VarSegment,
  Intersection(2): ()&(),VarSegment^2->ConPol, ac;
  Addition(2): ()+(),VarSegment^2->VarSegment, ac;
Operations
XCan{
      L ≤ R = LinCombCan(L-R),
      L ≥ R = LinCombCan(R-L),

      a≤ 0 -> a = I,
      a>0 -> a = O
};
// Правила, унаследованные от сорта Interval
Intersection{
  S&O = O,
  S&I = S
  };
/
```

Пересечение полуосей. Основной случай. Дополнительные случаи: $x < y$ или $x > y$. Случай $x < y$ или $x > y$ выводится из основного: $x > y \to S(y) = I(x).S(y)$. Операция «+» дистрибутивна относительно операции «&». Она упорядочивает слагаемые в сумме, возвращая сумму со знаками «++».

```
/

Intersection{
 GrEq(G,G1,T) -> S1.T&S = S1.T,                                  //1
~Eq(G&G1,0,T)&GrEq(G,L1,T)->{S1.T&S=[L1,y,G1].T1+ [L1,y,G].T2,
 where T1 = GrEq(G,G1,T), T2 = T-T1},                           //2
 ~Eq(G&G1,0,T)&~Eq(G&L1,Empty,T)->{S1.T&S=[L1,y,G1].T1+[L1,y,G].T2,
 where T1 = GrEq(G,G1,T), T2 = GrEq(G,G1,T)                     //3
 GrEq(L1,G,T)&GrEq(G,G1,T) -> S1.T&S = [G,y,L1].T,              //4
 GrEq(L1,G,T)&~Eq(G&G1,0) -> {S1.T&S =[L1,y,G].T1,
 where T1 = GrEq(G,L1,T),                                        //5
 GrEq(G,G1,T) -> S1.T & S = 0                                    //6
};

      // Приведение подобных
      Addition{
      [L,x,H] + [H,x,G] = [L,x,G]
      };

      Sort Trapezoid::ConSemyGroup;
      Constructor {
      Trapezoid T = (VarSegment S).(Trapezoid R);
      // Селекторы
      Head(T) = S,
      Tail(T) = R,
      LeadVar(T) = Var(S);
      // Контекстные условия
      LowGran(S) <=R UpGran(S),
      Var(S) > LeadVar(R);
      // Редукции
      T.I = T,
      I.T = T,
      T.O = O;
      };
      Signature
      Intersection(2):,()&(),VarSegment*Traperzoid->ConPol, ac;
      Addition(2): ()+(),Trapezoid^2->Trapezoid, ac;
      Operations
      Intersection{
      // Основное правило
         Var(S1) == Var(S) -> S1&S.T = (S1&S).T

// Дополнительные правила выводятся из основного с использованием редукций
      };
      // Дистрибутивный закон
      Addition{
      // Основное правило
         S1.T + S2.T = (S1 + S2).T
      };
      Sort ConPol::GegalkinPolynomials;
      Constructor {
      ConPol P = (VarSegment S).(ConPol Q) ++ ConPol R;
      // Селекторы
      Head(P) = T.Q,
      Tail(P) = R,
```

```
LeadVar(P) = LeadVar(T);
// Контекстные условия
LeadVar(P) = LeadVar(R),
LeadVar(P) LeadVar(Q);
// Редукции
 T.Q ++ O = T.Q,
 O ++ P = P
};
Signature
 Intersection(2):, ()&(),VarSegment*ConPol -> ConPol, ac;
 Addition(2): ()+(),ConPol^2->ConPol, ac;
Operations
Intersection{
// Основное правило
 Var(S1) = Var(S) - S1&(S.Q ++ R) = (S1&S).Q + S1&R,

// Дополнительные правила выводятся из основного с использованием редукций
};
// Дистрибутивный закон
Addition{
// Основное правило
(S1.Q ++ R1) + (S2.Q ++ R2) = (S1 + S2).Q ++ (R1 + R2)

// Дополнительные правила ДЗ вводятся с помощью соотношений редукции
// Дополнительные правила — правила слияния — упорядочивают слагаемые

Q1>Q2 -> (Q1++R1) + (Q2++R2) = Q1++(R1 + (Q2++R2),
Q1>Q2 -> (Q1++R1) + (Q2++R2)=Q2++(Q1++R1) + Q2),

// Остальные правила выводятся с помощью соотношений редукции
};
```

Автор благодарит академика А.А. Летичевского, который акцентировал внимание на задаче построения алгоритма решения СЛН методами алгебраического программирования и тем самым инициировал данную работу.

СПИСОК ЛИТЕРАТУРЫ

1. Goguen J.A., Thatcher J.W., Wagner E. An initial algebra approach to the specification, correctness and implementation of abstract data types. — Current trends in programming methodology (R.Yeh ed.). — Englewood Cliffs; New York: Prentice Hall, 1978. — P. 80–149.
2. Goguen J., Meseguer J. Ordered-sorted algebra I: partial and overloaded operations. Errors and inheritance. — SRI International, Comput. Scie. Lab., 1987.
3. Goguen J.A. Parameterized programming // IEEE Transact. On Soft Engineering. — 1984. — **SE-10**, N 5. — P. 528–543.
4. Futatsugi K., Goguen J.A., Jouannaud J.P., Meseguer J. Principles of OBJ-2 // Proc. 12th ACM Symposium on Principles of Program. Languages (B. Reid ed.). — 1985. — **ACM**. — P. 52–66.
5. Goguen J.A., Meseguer J. Eqlog: equality, types and generic modules for logic programming // Functional and Logic Programming / Ed. by D. DeGroot, G. Lindstrom. — New York: Prentice Hall, 1986. — P. 295–363.
6. http://www.axiom-developer.org/
7. http://www.axiom-developer.org/axiom-website/bookvol10.2full.html

8. Львов М.С. Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр // Вісн. Харків. нац. ун-ту. Сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». — 2009. — № 847. — С. 221–238.
9. Львов М.С. Верифікація інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр // Зб. наук. праць Харків. ун-ту Повітряних Сил. — Харків: ХУПС, 2009. — Вип. 3 (21). — С. 127–137.
10. Львов М.С. Метод морфізмів реалізації алгебраїчних обчислень в математичних системах навчального призначення // Системи обробки інформації. — 2009. — Вип. 6(80). — С. 183–190.
11. Львов М.С. Метод спадкування при реалізації алгебраїчних обчислень в математичних системах навчального призначення // Системи управління, навігації та зв'язку. — 2009. — Вип. 3 (11). — С. 120–130.
12. Львов М. Об одном подходе к реализации алгебраических вычислений: вычисления в алгебре высказываний // Вісн. Харків. нац. ун-ту. Сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». — 2009. — № 848. — С. 211–226.
13. Моцкин Т.С., Райфа Х., Томпсон Дж.Л., Тролл Р.М. Метод двойного описания. Матричные игры. — М.: Физматгиз, 1961. — С. 81–109.
14. Zeidler G.L. Lectures on polytopes. Graduate texts in mathematics. — New York: Springer-Verlag, 1994. — 370 p.
15. Черников С.Н. Линейные неравенства. — М.: Наука, 1968. — 490 с.
16. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. — М.: Мир, 1989. — 478 с.
17. Ласло М. Вычислительная геометрия и компьютерная графика на С++ / Пер. с англ. — М.: Бином, 1997. — 304 с.
18. Солодовников А.С. Системы линейных неравенств. — М.: Наука, 1977. — 112 с.
19. Емеличев В.А., Ковалев М.М., Кравцов М.К. Многогранники, графы, оптимизация. — М.: Наука, 1981. — 348 с.

Поступила 13.07.2009