Линейные ограничения и методы их решения

С. Л. Крывый

Аннотация: *Описаны алгоритмы построения базиса множества решений систем линейных ограничений в поле действительных чисел, минимального порождающего множества решений систем линейных однородных уравнений в множестве натуральных чисел и базиса множества решений системы линейных уравнений в кольце целых чисел, в кольцах и полях вычетов по модулю простого и составного числа. Эти алгоритмы рассматриваются в контексте решения проблемы выполнимости системы ограничений.*

Abstract: *The algorithms for computation of basis of solutions systems of linear constraints over set of real numbers, minimal supported set of solutions for systems of linear Diophantine homogeneous equations over set of natural numbers and basis of systems of linear Diophantine homogeneous and inhomogeneous equations over ring of integer numbers, over ring and field of residues on modulo are described. Thies algorithms consider in context of solving of general constraint satisfaction problem*

ACM Classification Keywords: *Systems of linear Diophantine constraints, minimal supported set of solutions, basis of solutions, constraint satisfaction problem*

1. Введение

В работе представлен обзор методов решения систем линейных и линейных диофантовых ограничений над числовыми областями [5]–[12], [44]–[52]. Эти методы рассматриваются в контексте проблемы выполнимости ограничений, которая относится к области программирования с ограничениями (constraint programming). Одной из центральных проблем в современном программировании с ограничениями является проблема определения вычислительной сложности решения задач, включающих ограничения. Существует большое количество проблем выполнимости ограничений (constraint satisfaction problems – CSP), являющихся NP-трудными [23]. Поэтому маловероятно существование общего эффективного алгоритма для решения произвольной проблемы, связанной с ограничениями.

Установление класса сложности, которому принадлежит тот или иной язык ограничений, иногда не является достаточным. Необходимо находить сами решения проблемы выполнимости ограничений. Методам решения проблемы выполнимости для языка линейных ограничений и посвящена эта работа.

Вначале даются необходимые определения и понятия, а также краткий обзор фактов, связанных с решением проблемы выполнимости множества ограничений. Далее приводятся алгоритмы построения минимального порождающего множества решений и базиса множества решений систем линейных ограничений над полем действительных чисел, линейных диофантовых ограничений в множестве целых чисел, натуральных чисел, поле и кольце вычетов по модулю простого и составного числа. В основе предлагаемых алгоритмов лежит метод построения минимального порождающего множества решений систем линейных однородных диофантовых уравнений в множестве натуральных чисел [5]. К такого рода системам и методам их решений сводятся задачи математических игр [2], распознавания изображений и построение линейных мозаик [3], криптографии [15], распараллеливания циклов [20], анализа свойств сетей Петри [53], ассоциативно-коммутативной унификации в теориях первого порядка [21] и многие другие задачи.

В последнем разделе приведены некоторые приложения, представленных в этой работе методов.

2. Проблема выполнимости ограничений

2.1. Определения и обозначения

Введем определения основных понятий и обозначения, используемые в дальнейшем изложении.

Определение 1. *Пусть D – некоторое множество и $n \in \mathcal{N}$ – натуральное число. Декартовой степенью множества D называется множество D^n, состоящее из всех упорядоченных n-ок элементов из D, т. е. $D^n = \{(d_1, \ldots, d_n) | d_i \in D, i = 1, \ldots, n\}$.*

Произвольное подмножество $R \subseteq D^n$ называется n-арным отношением на D.

В зависимости от мощности множества D, отношения на D могут быть конечными или бесконечными. В дальнейшем будут рассматриваться только конечные отношения на D, а множество всех таких отношений на D будет обозначаться $\mathcal{R}_D$.

Языком ограничений L на D называется некоторое непустое множество $L \subseteq \mathcal{R}_D$.

Определение 2. *Для произвольного множества D и произвольного языка ограничений L на D проблемой выполнимости ограничений CSP(L) называется решение такой комбинаторной проблемы:*

дано: *тройка $P = (V,D,C)$, где*

- *V – множество переменных;*
- *C – некоторое множество ограничений $\{C_1, \ldots, C_q\}$;*
- *каждое ограничение $C_i \in C$ – это пара (s_i, R_i), где*
  - *$s_i = (v_1, v_2, \ldots, v_n)$ – n-ка длины n, называемая областью ограничения;*
  - *$R_i \in L$ – n_i-арное отношение на D, называемое отношением ограничения.*

Выяснить: *существует ли решение ограничения, т. е. функция $\varphi : V \to D$ такая, что $\forall (s_i, R_i) \in C$, n-ка $(\varphi(v_1), \ldots, \varphi(v_n)) \in R_i$ и если существует, то какая сложность его вычисления?*

Множество D в этом случае называется **областью проблемы**. Множество всех решений CSP вида $P = (V, D, C)$ обозначается $Sol(P)$.

Для того, чтобы установить вычислительную сложность CSP необходимо определить каким образом кодируется проблема в виде конечной последовательности символов. **Размером** данной проблемы будет называться длина переменных, длина проблемы, длина всех областей определений и соответствующих отношений ограничений. При этом предполагается, что во всех случаях такое представление выбрано так, что сложность установления того, допускает ли данное ограничение данную n-ку значений переменных из своей области ограничений, является полиномиальной функцией от длины представления. Для конечных областей этого требования более чем достаточно, поскольку n-ки в отношениях можно перечислить явно (хотя на практике отношения задаются, как правило, неявно).

Определение 3. *Язык ограничений L называется легко обрабатываемым (tractable), если $CSP(L')$ может быть решена в полиномиальном времени для каждого конечного подмножества $L' \subseteq L$.*

Язык ограничений L называется NP-полным, если $CSP(L')$ является NP-полной проблемой для некоторого конечного подмножества $L' \subseteq L$.

Известно множество вычислительных проблем, сложность решения которых не принадлежат ни классу полиномиальной сложности, ни классу NP-полной сложности. Но существуют языки ограничений, сложности решения которых принадлежат одному из этих двух классов [34].

$(n+1)$-арное отношение R на $(D^n \times B)$ называется *функциональным* (или функцией), если для любой n-ки $(d_1, d_2, \ldots, d_n) \in D^n$ существует не более одного элемента $b \in B$ такого, что $(d_1, d_2, \ldots, d_n, b) \in R$ (обозначение $R : D^n \to B$). Как правило, функции обозначаются малыми латинскими буквами $f : D^n \to B$. Этим обозначениям и будем следовать.

Частным случаем понятия функции является понятие операции. Функция $f : D^n \to D$ называется n-арной операцией на множестве D.

Определение 4. *Универсальной алгеброй называется пара* $G = (D, \Omega)$, *где* D – *некоторое множество, называемое* носителем *или* основным множеством *алгебры, а* $\Omega = \{f_1^{k_1}, \ldots, f_n^{k_n}, \ldots\}$ – *множество определенных на* D *операций конечной фиксированной арности, называемое* сигнатурой *алгебры. Операции из* Ω *называются* основными.

Алгебра называется конечной, если конечно ее основное множество D, *а множество всех операций конечной арности на* D *обозначается* O_D.

Две универсальные алгебры $G = (D, \Omega)$ и $G' = (D', \Omega')$ называются *однотипными*, если между их сигнатурами существует такое взаимно однозначное соответствие, при котором каждой n-арной операции $f \in \Omega$ соответствует операция $f' \in \Omega'$ той же арности.

2.2. Примеры языков ограничений

1. Язык линейных уравнений над бесконечным полем. Пусть $\mathcal{D}$ – поле действительных чисел, т. е. это алгебра вида $G = (\mathcal{D}, \{+, -, \cdot, :, 0, 1\}$ с бинарными операциями сложения, вычитания, умножения, деления и двумя нульарными операциями 0 и 1. Пусть $L = L_{lin}$ – язык ограничений, состоящий из всех конечных отношений на $\mathcal{D}$, элементами которых являются все решения некоторой системы линейных уравнений над $\mathcal{D}$.

Произвольное отношение из L_{lin}, а также произвольная проблема $CSP(L_{lin})$ могут быть представлены некоторой системой линейных уравнений над $\mathcal{D}$ и решены в полиномиальном времени [24] (например, с помощью алгоритма последовательного исключения неизвестных). Следовательно, L_{lin} является легко обрабатываемым языком.

Этот язык над различными числовыми областями является основным объектом исследования в данной работе.

2. Язык L_{ZOA}. Пусть – D – конечное множество и L_{ZOA} – язык всех отношений на D такого вида:

- все унарные отношения;
- все бинарные отношения вида $D_1 \times D_2$, где $D_1, D_2 \subseteq D$;
- все бинарные отношения вида $\{d, \pi(d) | d \in D_1\}$ для некоторого $D_1 \subseteq D$, и некоторой перестановки π множества D_1;
- все бинарные отношения вида $\{(a, b) \in D_1 \times D_2 | a = d_1 \vee b = d_2\}$ для некоторых подмножеств $D_1, D_2 \subseteq D$ и некоторых элементов $d_1 \in D_1$ и $d_2 \in D_2$.

Известно, что этот язык является легко обрабатываемым и что для любого бинарного отношения R на D, не принадлежащего к L_{ZOA}, язык $L_{ZOA} \cup \{R\}$ является NP-полным [37].

3. Язык строгих неравенств. Бинарное отношение строгого неравенства на (частично) упорядоченном множестве D имеет вид:

$$<_D = \{(d_1, d_2) \in D^2 | d_1 < d_2\}.$$

Если D является множеством натуральных чисел $\mathcal{N}$, то класс проблем выполнимости ограничений для $CSP(<_{\mathcal{N}})$ соответствует проблеме проверки ацикличности орграфа (*ACYCLIC DIGRAPH problem*). Эта проблема состоит в том, чтобы для заданного орграфа G выяснить, является ли G ациклическим орграфом. Известно, что орграф является ациклическим тогда и только тогда, когда его вершины можно перенумеровать таким образом, что каждая его дуга соединяет вершину с меньшим номером с вершиной с большим номером.

Язык $<_{\mathcal{N}}$ является легко обрабатываемым [22].

4. Язык отрицания равенства. Бинарное отношение на D, определяемое как

$$\neq_D = \{(d_1, d_2) \in D^2 | d_1 \neq d_2\},$$

соответствует проблеме раскраски графа $|D|$ красками. Известно, что данный язык легко обрабатываемый, если $|D| \leq 2$ или $|D| = \infty$, и является NP-полным, если $3 \leq |D| < \infty$ [39].

5. Язык булевских ограничений. Язык ограничений над двухэлементным множеством $D = \{d_0, d_1\}$ известен как булевский язык ограничений. Используя этот язык, можно выразить стандартную форму пропозициональной проблемы выполнимости, имеющей название ВЫПОЛНИМОСТЬ в виде CSP, путем интерпретации элементов из D как ложь и истина [39].

Известен результат Шафира [38] о том, что булевский язык ограничений L легко обрабатываем если выполняется хотя бы одно из следующих условий:

1. каждое отношение из L содержит n-ку, в которой все компоненты равны d_0;
2. каждое отношение из L содержит n-ку, в которой все компоненты равны d_1;
3. каждое отношение из L определяется КНФ, где каждый дизъюнкт имеет один негативный литерал;
4. каждое отношение из L определяется КНФ, где каждый дизъюнкт имеет один позитивный литерал (хорновский дизъюнкт);
5. каждое отношение из L определяется КНФ, где каждый дизъюнкт содержит два литерала;
6. каждое отношение из L является множеством решений системы линейных уравнений над полем F_2 вычетов по модулю 2.

Во всех остальных случаях язык L является NP-полным. Приведенный результат известен как *дихотомическая теорема Шафира*. Непосредственно из этой теоремы следует, что некоторые булевские языки ограничений, содержащие единственное отношение, являются NP-полными. Например, для $D = \{d_0, d_1\}$ и тернарного отношения N_D, называемого “НЕ-ВСЕ-РАВНЫ” и имеющего вид

$$N_D = D^3 \setminus \{(d_0, d_0, d_0), (d_1, d_1, d_1), (d_0, d_0, d_1), (d_0, d_1, d_0), (d_0, d_1, d_1), (d_1, d_0, d_1), (d_1, d_1, d_0)\},$$

язык N_D является NP-полным [38].

3. Алгебраический подход к проблеме выполнимости

Многие работы, посвященные CSP, показали, что сложность языка ограничений можно охарактеризовать, если использовать алгебраические свойства его отношений (см. рис. 3.1).

В связи с этим обстоятельством имеется алгебраическая теория, которую вкратце изложим здесь, а более полное ее изложение можно найти в монографии [29].

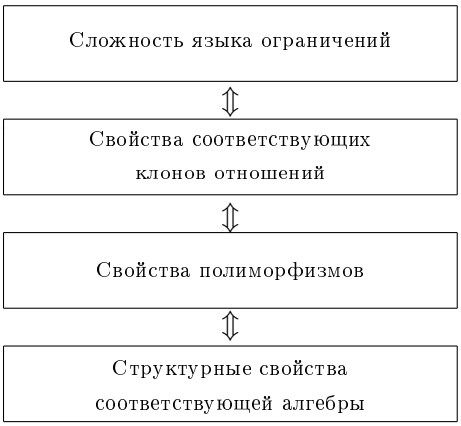


Рис. 3.1. Трансляция вопросов о сложности

В основе первого шага в алгебраическом подходе к анализу сложности языков ограничений является хорошо известная идея: дано исходное множество отношений, найти отношения, прибавление которых к исходному множеству не изменяет сложности соответствующего класса проблем. Известны примеры отношений, для которых добавление всех отношений, выводимых из исходных отношений, используя определенные простые правила, не изменяет сложности. Отношения, полученные таким образом, известны под названием **клоны отношений**. Следовательно, первый шаг в анализе сложности языков состоит в определении таких множеств отношений, которые являются клонами отношений.

Следующий шаг в алгебраическом подходе состоит в описании клонов отношений с помощью **полиморфизмов**, которые являются алгебраическими операциями на некотором выделенном множестве [28, 30].

Напомним определение клона для универсальной алгебры.

Пусть D некоторое множество. Для натуральных $n \geq 1$ и $1 \leq m \leq n$ **операцией селекции** или просто **селекцией** на D называется n-арная операция вида $I_m^n(a_1, \ldots, a_n) = a_m$ для всех $a_1, a_2, \ldots, a_n \in D$. Если f – n-арная функция на D и $g_1, g_2, \ldots, g_n$ – k-арные функции, то k-арная функция

$$g(a_1, \ldots, a_k) = f(g_1(a_1, \ldots, a_k), \ldots, g_n(a_1, \ldots, a_k))$$

называется **суперпозицией функций** $f, g_1, \ldots, g_n$ для всех $a_1, \ldots, a_k \in D$.

Определение 5. *Множество операций, определенных на некотором заданном множестве D, называется* **клоном** *на D тогда и только тогда, когда оно содержит все селекции и замкнуто относительно суперпозиции.*

Последний шаг в алгебраическом подходе связывает языки ограничений с конечными универсальными алгебрами. Языки конечных алгебр позволяют создавать много новых достаточно мощных средств для анализа сложности ограничений.

3.1. От отношений к клонам отношений

Таким образом, первый шаг в алгебраическом подходе сводится к рассмотрению того, какие дополнительные отношения могут быть добавлены к языку ограничений без изменения сложности соответствующего класса проблем. Эта техника широко использовалась

при анализе проблем выполнимости булевских ограничений [25, 38], а также при анализе темпоральных и пространственных ограничений [26, 32, 33, 35, 36].

Определение 6. *Язык ограничений L выражает отношение R, если существует $P = (V, D, C) \in CSP(L)$ и список $(v_1, \ldots, v_n)$ переменных из V таких, что $R = \{(\varphi(v_1), \ldots, \varphi(v_n)) | \varphi \in Sol(P)\}$.*

Для произвольного языка ограничений L множество отношений, которые могут быть выражены с помощью языка L, называется **выразительной мощность** языка L. Выразительную мощность языка ограничений L можно охарактеризовать многими различными способами [31]. Например, это множество может быть равно множеству всех отношений, полученных из отношений языка L с помощью операций соединения и проекции реляционной алгебры. Можно также показать, что оно равно множеству отношений, определяемых примитивными позитивными формулами над отношениями из L вместе с отношением равенства, где примитивными позитивными формулами называются формулы языка предикатов первого порядка, построенные с помощью конъюнкции и кванторов существования. В алгебраической терминологии это множество называется **клоном отношений**, порожденных языком L и обозначается $\langle L \rangle$.

Пример 1. Рассмотрим булевский язык ограничений $L = \{R_1, R_2\}$ на $D = \{0, 1\}$, где $R_1 = \{(0,1), (1,0), (1,1)\}$ и $R_2 = \{(0,0), (0,1), (1,0)\}$. Непосредственно проверяется, что все 16 отношений могут быть выражены с помощью примитивных позитивных формул, включающих R_1 и R_2. Например, отношение $R_3 = \{(0,0), (1,0), (1,1)\}$ можно выразить формулой $R_3 = \exists y(R_1(x, y) \wedge R_2(y, z))$. Следовательно, $\langle L \rangle$ – клон отношений, порождаемый языком L и включающий все 16 булевских отношений.

Можно показать, что для этого языка ограничений L клон $\langle L \rangle$ в точности состоит из тех булевских отношений (произвольной арности), которые могут быть выражены конъюнкциями унарных и бинарных булевских отношений [42, 43]. ♠[1]

Связь между этими понятиями и сложностью языка ограничений устанавливается следующим утверждением.

Теорема 1. *Для произвольного языка ограничений L и любого конечного подмножества $L_0 \subseteq \langle L \rangle$ существует редукция в полиномиальном времени языка $CSP(L_0)$ к языку CSP(L).*

Из этой теоремы вытекает

Следствие 1. *Язык ограничений L легко обрабатываем тогда и только тогда, когда его клон $\langle L \rangle$ легко обрабатываемый. Аналогично L является NP-полным тогда и только тогда, когда $\langle L \rangle$ NP-полный.*

3.2. От клонов отношений к множествам операций

Как следует из предыдущего, анализ сложности произвольного языка ограничений над конечными областями сводится к анализу клонов отношений. Рассмотрим подход к представлению и описанию клонов отношений, который базируется на свойствах операций.

Произвольную операцию $f \in O_D$ можно естественным образом расширить на кортежи элементов из D. Для произвольной k-арной операции $f \in O_D$ и $t_1, \ldots t_k \in D^n$ определим

$$f(t_1, \ldots, t_k) = ((f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n])),$$

[1] Знак ♠ означает конец примера

где $t_i[j]$ – j-ая компонента $t_i,\ i = 1, \ldots, k,\ j = 1, \ldots, n$.

Определение 7. *Говорят, что k-арная операция $f \in O_D$ сохраняет n-арное отношение $R \in \mathcal{R}_D$ (или является полиморфизмом R, или R инвариантно относительно f, или R стабильно относительно f), если $f(t_1, \ldots, t_k) \in R$ для всех $t_1, \ldots, t_k \in R$.*

Это означает, что

$$f(t_1, \ldots, t_k) = ((f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n])) \in R \subseteq D^n.$$

Для произвольного языка $L \subseteq \mathcal{R}_D$ и $\Omega \subseteq O_D$ определим такие отображения:
$Pol(L) = \{f \in O_D | f$ сохраняет каждое отношение из $L\}$;
$Inv(\Omega) = \{R \in \mathcal{R}_D | R$ инвариантно относительно каждой операции из $\Omega\}$.
Из этого определения следует, что

$$Inv(\Omega) = Inv(Pol(Inv(\Omega)))$$

для произвольного множества операций Ω (операторы Inv и Pol образуют соответствие Галуа между O_D и $\mathcal{R}_D$).

Для универсальных алгебр известно, что клоны отношений, порожденные множеством отношений над конечным множеством, определяются полиморфизмами этих отношений [40].

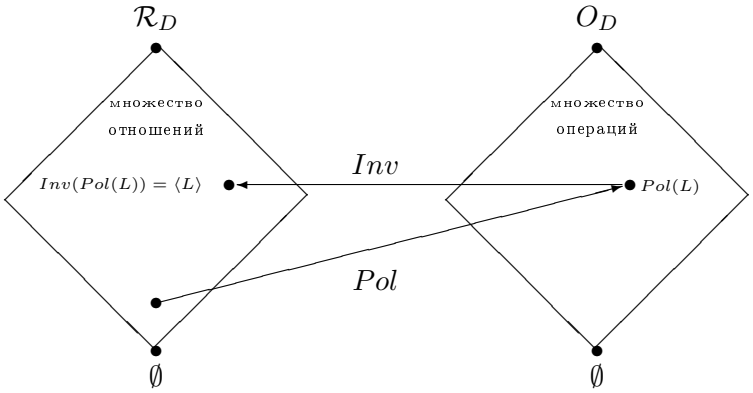


Рис 3.2. Операторы Inv и Pol

Определение 8. *Пусть L – конечный язык ограничений над множеством D. Для произвольного натурального числа k проблемой индикации порядка k для L называется CSP проблема $P = (V,D,C)$, где*

- $V = D^n$ *(каждая переменная в P является кортежом элементов длины k из D);*
- $C = \{(s, R) | R \in L$ *и s соответствует $R\}$.*

При этом список кортежей длины n для $s = (v_1, \ldots, v_n)$ соответствует R, если k – это арность R и для всех $i \in \{1, 2, \ldots, k\}$ n-ка $(v_1[i], \ldots, v_n[i]) \in R$. Следовательно, CSP проблема P имеет ограничения из языка ограничений L на всех возможных областях соответствующих отношению из L.

Заметим, что проблемы индикации порядка k для L являются отображениями из D^k в D, которые сохраняют каждое из отношений. Следовательно, эти решения в точности составляют k-арные элементы из $Pol(L)$.

Более подробная информация о проблеме индикации содержится в работе [27].

Теорема 2. *Для произвольного языка ограничений L над конечным множеством D имеет место равенство $\langle L\rangle = Inv(Pol(L))$ [28, 40].*

Поскольку клон отношений $\langle L\rangle$ состоит из таких отношений, которые могут быть выражены в языке отношений L, то отсюда следует строгая связь между полиморфизмами и выразительной мощностью языка L.

Из теорем 1 и 2 вытекает

Следствие 2. *Произвольное отношение R над конечным множеством D может быть выражено в языке ограничений L тогда, когда $Pol(L) \subseteq Pol(\{R\})$.*

Для произвольных языков ограничений L и L_0 над конечным множеством D, если L_0 конечный язык и $Pol(L) \subseteq Pol(L_0)$, то $CSP(L_0)$ редуцируется к $CSP(L)$ в полиномиальном времени.

Эти следствия показывают, что произвольный конечный язык ограничений L над конечной областью D определяется с точностью до редукций, выполняемых в полиномиальном времени, полиморфизмами из L. Следовательно, можно транслировать исходную проблему характеризации легко обрабатываемого языка ограничений в эквивалентную ей проблему для множества операций.

Определение 9. *Множество операций $\Omega \subseteq O_D$ называется легко обрабатываемым, если $Inv(\Omega)$ легко обрабатываемый. Множество $\Omega \subseteq O_D$ называется NP-полным, если $Inv(\Omega)$ является NP-полным.*

3.3. От множеств операций к алгебрам

Для выполнения трансляции множества операций в алгебры заметим, что произвольное множество операций Ω на заданном множестве D можно рассматривать как алгебру $G = (D, \Omega)$.

Определение 10. *Алгебра $G = (D, \Omega)$ называется легко обрабатываемой, если множество ее основных операций Ω легко обрабатываемое. Алгебра $G = (D, \Omega)$ называется NP-полной, если Ω является NP-полным.*

Таким образом, основная задача теперь свелась к задаче характеризации всех легко обрабатываемых алгебр.

Рассмотрим сначала отношение эквивалентности, связывающее алгебры и соответствующие языки ограничений. Как отмечалось выше, отображения Inv и Pol связаны равенством $Inv(Pol(Inv(\Omega))) = Inv(\Omega)$, а это позволяет расширить множество операций Ω до множества $Pol(Inv(\Omega))$ без изменения соответствующих инвариантных отношений. Множество $Pol(Inv(\Omega))$ состоит из всех операций, которые строятся из всех операций проекции, получаемых из произвольных операций суперпозиции. В алгебре такое множество операций называют **клоном операций**, а в компьютерных науках его называют **множеством термальных операций** над Ω.

Определение 11. *Для произвольной алгебры $G = (D, \Omega)$ операция f на D называется термальной операцией алгебры G, если $f \in Pol(Inv(\Omega))$. Множество всех термальных операций алгебры G обозначается $Term(G)$.*

Две алгебры, имеющие один и тот же носитель, называются **термально эквивалентными**, если они имеют одинаковые множества термальных операций. Следовательно, для

любой алгебры $G = (D, \Omega)$ имеет место равенство $Inv(\Omega) = Inv(Term(G))$ и две алгебры являются термально эквивалентными, если они имеют одно и то же множество соответствующих инвариантных отношений. Отсюда следует, что легко обрабатываемые алгебры можно характеризовать с точностью до термальной эквивалентности.

Можно ограничиться рассмотрением специального класса алгебр. Первое ограничение следует из того, что любой унарный полиморфизм языка ограничений, применяемый ко всем отношениям, не изменяет сложности.

Предложение 1. *Пусть L – язык ограничений над D и f – унарная операция из Pol(L). CSP(L) является полиномиально-сложностно эквивалентной CSP(f(L)), где* $f(L) = \{f(R)|R \in L\}$ *и* $f(R) = \{f(t)|t \in R\}$ *[28, 30].*

Если воспользоваться предложением 1 с унарным полиморфизмом f, который имеет наименьший возможный ранг среди всех унарных полиморфизмов из L, то получается язык ограничений $f(L)$, все унарные полиморфизмы которого сюръективны. Такой язык называют **редуцируемым языком ограничений**.

Определение 12. *Алгебра называется сюръективной, если все её термальные операции сюръективны.*

Легко проверить, что конечная алгебра сюръективна тогда и только тогда, когда все её унарные операции сюръективны и, следовательно, составляют группу перестановок. Из этого вытекает, что алгебра $G = (D, \Omega)$ является сюръективной тогда и только тогда, когда $Inv(\Omega)$ является редуцируемым языком ограничений. Используя предложение 1, это означает, что можно ограничиться сюръективными алгебрами.

Следующая теорема показывает, что для наших целей необходимо рассматривать только сюръективные алгебры, которые удовлетворяют свойству идемпотентности.

Определение 13. *Операция f на D называется идемпотентной, если для всех x имеет место равенство f(x,. . . ,x) = x.*

Полным идемпотентным редуктом алгебры $G = (D, \Omega)$ *называется алгебра G’ = (D, Termid(G)), где Termid(G) состоит из всех идемпотентных операций из Term(G).*

Операция f на множестве D является идемпотентом тогда и только тогда, когда она сохраняет все отношения в множестве $L_{CON} = \{\{a\}|a \in D\}$, состоящим из всех унарных одноэлементных отношений на D. Следовательно, $Inv(Termid(G))$ является клоном отношений порожденным множеством $Inv(\Omega) \cup L_{CON}$. Отсюда следует такая

Теорема 3. *Конечная сюръективная алгебра легко обрабатываемая тогда и только тогда, когда её полный идемпотентный редукт* G_0 *является легко обрабатываемым. Алгебра G является NP-полной тогда и только тогда, когда* G_0 *NP-полна [24].*

Свяжем теперь сложность конечной алгебры со сложностью её подалгебр и гомоморфных образов. Во многих случаях этими результатами можно воспользоваться для сведения проблемы анализа сложности алгебры к аналогичной проблеме, включающей алгебру с меньшим носителем. А это позволяет свести проблему анализа сложности языка ограничений к языку ограничений над меньшей областью.

Определение 14. *Пусть* $G = (D, \Omega)$ *– некоторая алгебра и* $B \subseteq D$ *такое, что для произвольной k-арной операции* $f \in \Omega$ *и произвольных* $b_1, \ldots, b_k \in B$ *имеет место включение* $f(b_1, \ldots, b_k) \in B$. *Тогда алгебра* $G_1 = (B, \Omega|_B)$ *называется подалгеброй алгебры G,*

где $\Omega|_B$ – ограничение операций из Ω на множество B. Если $D \neq B$, то G_1 называется собственной подалгеброй алгебры G.

Пусть $G_1 = (D_1, \Omega_1)$ и $D_2 = (D_2, \Omega_2)$ две однотипные алгебры. Отображение $\gamma : G_1 \to G_2$ называется гомоморфизмом, алгебры G_1 в G_2, если для любого $i = 1, 2, \ldots, |\Omega_1|$ выполняется равенство

$$\gamma(f_i^1(a_1, \ldots, a_k)) = f_i^2(\gamma(a_1), \ldots, \gamma(a_k)),$$

где $f_i^1 \in \Omega_1, f_i^2 \in \Omega_2$ – соответствующие k-арные операции.

Если γ является сюрьективным отображением, то алгебра G_2 называется гомоморфным образом алгебры G_1.

Гомоморфный образ подалгебры некоторой алгебры G называется **фактором алгебры** *G.*

Теорема 4. *Если G является легко обрабатываемой конечной алгеброй, то таким будет и каждый её фактор. Если G имеет хотя бы один NP-полный фактор, то G NP-полна [24].*

Как отмечалось выше, в практических применениях необходимо устанавливать не только класс сложности CSP для некоторого языка ограничений, но и находить сами решения этой проблемы. В частности, такая ситуация наблюдается в области числовых ограничений.

4. Числовые ограничения и их классификация

Если область, над которой рассматриваются ограничения, является числовой, то такие ограничения называются числовыми.

В теории числовых ограничений **числовой областью** O называют одну из таких областей:

$\mathcal{D}$ – поле действительный чисел;

$\mathcal{Q}$ – поле рациональных чисел;

F_p – поле вычетов по модулю простого числа p;

$\mathcal{Z}$ – кольцо целых чисел;

Z_m – кольцо вычетов по модулю составного числа m;

$\mathcal{N}$ – коммутативная полугруппа натуральных чисел;

$\{0, 1\}$ – двухэлементное множество, называемое **булевским**.

Если множество O есть одним из таких множеств: $\mathcal{Z}, \mathcal{N}, Z_m, F_p$ или $\{0, 1\}$, то множество O в таком случае называется **дискретной областью**. А если множество O является полем $\mathcal{D}$ или $\mathcal{Q}$, то множество O называется **непрерывной областью**.

Символом F далее будем обозначать одно из полей $\mathcal{D}, \mathcal{Q}, F_p$, а символом O или O с индексом – одну из числовых областей.

Определение 15. Числовой функцией *называется отображение $f : O_1 \times \ldots \times O_n \to O$, где O, O_i – числовые области, $i = 1, 2, \ldots, n$.*

В практических приложениях наиболее часто встречаются ограничения, которые являются **линейными числовыми функциями**

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} c_i x_i, \quad c_i \in O,$$

и **полиномами**

$$P(x_1,\dots,x_n)=\sum_{(i_1,\dots,i_n)\in\mathcal{N}^n, i_1+\dots+i_n=k} c_{i_1\dots i_n}x_1^{i_1}\dots x_n^{i_n},$$

где $c_{i_1\dots i_n}\in O,\ k\in\mathcal{N}$,

Определение 16. Числовым ограничением *называется равенство* $(=)$, *неравенство* $(\leq,<,\geq,>)$ *или отрицание равенства* $(\neq)$ *между двумя числовыми функциями.*

Классификация числовых ограничений выполняется в соответствии с видом их

– числовых функций (линейные или нелинейные) и

– числовых областей, в которых ищутся решения этих ограничений (непрерывные $\mathcal{D},\mathcal{Q}$ или дискретные $\mathcal{Z},\mathcal{N},F_p,Z_m,\{0,1\}$).

Числовые ограничения, которые рассматриваются над дискретными числовыми областями, называются **диофантовыми**.

Дальше будут рассматриваться только линейные ограничения и системы таких ограничений.

При рассмотрении линейных ограничений, можно предполагать, что все отношения ограничений имеют одинаковую арность. Действительно, если два отношения R и R_1 имеют арности m и n, где $m>n$, то в решении этих ограничений для второго отношения последние $m-n$ координат игнорируются.

Среди линейных ограничений системы линейных однородных уравнений (СЛОУ) (т. е. отношения R_i являются линейными однородными уравнениями) занимают особое место. Это объясняется тем, что в общем случае к методам решения таких систем сводятся системы линейных однородных уравнений (СЛОУ), системы линейных неоднородных уравнений (СЛНУ), системы линейных однородных неравенств (СЛОН) и системы линейных неоднородных неравенств (СЛНН).

Пусть

$$S=\left\{\begin{array}{rcrcrcrcrcr} L_1(x) & = & a_{11}x_1 & + & \dots & + & a_{1q}x_q & = & 0,\\ L_2(x) & = & a_{21}x_1 & + & \dots & + & a_{2q}x_q & = & 0,\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ L_s(x) & = & a_{s1}x_1 & + & \dots & + & a_{sq}x_q & = & 0 \end{array}\right. \qquad (1)$$

– СЛОУ над полем F и $e_1=(1,0,\dots,0,0),e_2=(0,1,\dots,0,0),\dots\dots,e_q=(0,\,0,\dots,0,1)$ – единичные векторы из множества F^q, которые называются **векторами канонического базиса** множества F^q.

Пусть M – множество решений системы S. Поскольку система S однородная, то нулевой вектор всегда будет ее решением. Это решение называется **тривиальным**, а любое решение системы S, отличное от нулевого, называется **нетривиальным** решением.

СЛОУ S будем называть **несовместной** (а соответствующее ей множество отношений ограничения *противоречивым*), если множество M состоит только из тривиального решения, в противном случае она будет называться **совместной**.

Множество $B\subseteq M$ решений системы S называется **базисом** M, если произвольное решение из M единственным образом представляется в виде линейной комбинации решений из множества B.

Таким образом, решить СЛОУ означает построить базис множества ее решений. Известно, что для совместной СЛОУ базис множества ее решений всегда существует, он конечен и существуют алгоритмы его вычисления [17].

Рассмотрим СЛНУ $Ax = b$ вида

$$\left\{\begin{array}{ccccccccc} a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & b_s. \end{array}\right. \tag{2}$$

СЛОУ, которая получается из этой неоднородной системы уравнений путем замены в ней свободных членов нулями, называется **приведенной** СЛОУ для системы $Ax = b$.

Между решениями СЛНУ и приведенной СЛОУ существует связь, которая вытекает из такой простой теоремы.

Теорема 5. *а) Сумма произвольного решения x_1 СЛНУ $Ax = b$ и любого решения x_2 приведенной СЛОУ $Ax = 0$ является решением СЛНУ $Ax = b$.*

б) Разница $x_1 - x_2$ любых двух решений x_1 и x_2 СЛНУ $Ax = b$ является решением приведенной СЛОУ $Ax = 0$.

в) Общее решение СЛНУ $Ax = b$ представляется в виде

$$x = y + x_0, \tag{3}$$

где x_0 – общее решение приведенной СЛОУ, а y – произвольное решение СЛНУ $Ax = b$.

Доказательство. *а*) Из условия получаем, что $Ax_1 = b$ и $Ax_2 = 0$. Тогда

$$A(x_1 + x_2) = Ax_1 + Ax_2 = b + 0 = b,$$

но это значит, что $x_1 + x_2$ – решение СЛНУ $Ax = b$.

б) В силу равенств $Ax_1 = b$ и $Ax_2 = b$ имеем

$$A(x_1 - x_2) = Ax_1 - Ax_2 = b - b = 0,$$

но это значит, что $x_1 - x_2$ – решение приведенной СЛОУ $Ax = 0$ для $Ax = b$.

в) В силу пункта *а*) произвольный вектор x, который имеет вид $x = y + x_0$, является решением СЛНУ $Ax = b$. Пусть x – любое решение СЛНУ $Ax = b$. Тогда, на основе пункта *б*), вектор $x - y = x_0$ – решение приведенной СЛОУ $Ax = 0$. Следовательно, вектор $x = y + x_0$ принадлежит к множеству решений, которые имеют вид (3). ■[2]

Теперь покажем, что процесс решения произвольной неоднородной системы линейных уравнений или неравенств может быть сведен к процессу решения СЛОУ, чем и объясняется особенность этих систем в теории числовых ограничений. Действительно, если дано СЛНУ (2), где $b_1, \ldots, b_s \in F$ – свободные члены и коэффициенты $a_{ij} \in F$, то введем новое дополнительное неизвестное x_0 и редуцируем систему (2) к такому виду:

$$\left\{\begin{array}{ccccccccccc} -b_1x_0 & + & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0, \\ -b_2x_0 & + & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ -b_sx_0 & + & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & 0. \end{array}\right.$$

Отсюда вытекает, что когда множество M решений этой СЛОУ не включает ни одного решения, у которого первая координата равна 1, то СЛНУ (2) несовместна. А проверка несовместности СЛНУ вытекает из такой теоремы [52].

[2]Знак ■ означает конец доказательства

Теорема 6. *Система $A \cdot x = b$ несовместна над F^q тогда и только тогда, когда существует вектор $v \in F^s$ такой, что $vA = 0$ и $(v, b^T) \neq 0$, где (v, b^T) – скалярное произведение векторов v и b^T, а b^T – вектор-строка свободных членов системы.*

Следовательно для проверки несовместности СЛНУ $A \cdot x = b$ необходимо найти решения СЛОУ $A^T \cdot x = 0$, а потом проверить наличие хотя бы одного решения, неортогонального вектору b^T, где A^T – матрица, транспонированная к матрице A.

Если дана система линейных неравенств

$$\begin{cases} a_{11}x_1 & + & \ldots & + & a_{1q}x_q & \geq & b_1, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{i1}x_1 & + & \ldots & + & a_{iq}x_q & \leq & b_i, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & > & b_s, \end{cases}$$

то, вводя новые дополнительные переменные $x_{q+1}, x_{q+2}, \ldots, x_{q+s}$ так, чтобы $x_{q+1} \geq 0, \ldots, x_{q+i} \leq 0, \ldots, x_{q+s} > 0$, можем записать эту систему неравенств в виде системы неоднородных уравнений:

$$\begin{cases} a_{11}x_1 & + & \ldots & + & a_{1q}x_q & - & x_{q+1} & = & b_1, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ a_{i1}x_1 & + & \ldots & + & a_{iq}x_q & + & x_{q+i} & = & b_i, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & - & x_{q+s} & = & b_s. \end{cases}$$

От этой системы уравнений переходим к СЛОУ описанным выше способом. Следует заметить, что в случае такого перехода в связи с увеличением числа неизвестных увеличивается размерность системы, что негативно сказывается на эффективности вычислений.

Далее основное внимание будет уделяться методам решения систем линейных однородных уравнений.

5. Язык линейных ограничений над полем $\mathcal{D}$

Однородные уравнения. Пусть дано линейное однородное уравнение (ЛОУ)

$$a_1x_1 + a_2x_2 + \ldots + a_qx_q = 0, \tag{4}$$

где $a_i \in \mathcal{D}, i = 1, 2, \ldots, q$.

Не ограничивая общности предположим, что $a_1 \neq 0$, тогда комбинируя этот коэффициент с остальными построим такое множество векторов:

$$s_1 = (a_2, -a_1, 0, 0, \ldots, 0, 0),\ s_2 = (a_3, 0, -a_1, 0, \ldots, 0, 0), \ldots,$$
$$s_{q-2} = (a_{q-1}, 0, 0, 0, \ldots, -a_1, 0),\ s_{q-1} = (a_q, 0, 0, 0, \ldots, 0, -a_1).$$

Очевидно, что построенные векторы являются линейно независимыми решениями уравнения (4). Если некоторые коэффициенты в ЛОУ равны нулю, то это множество векторов пополняется соответствующими векторами канонического базиса. Полученное таким способом множество решений будем называть TSS (TSS от английского Truncated Set of Solutions).

Лемма 1. *TSS является базисом множества всех решений уравнения* (4). *Сложность построения TSS пропорциональна величине* q^2.

Доказательство. Пусть $s = (b_1, b_2, \ldots, b_q)$ – произвольное решение (4), тогда

$$s + \frac{b_2}{a_1}s_1 + \frac{b_3}{a_1}s_2 + \ldots + \frac{b_q}{a_1}s_{q-1} = (b_1 + \frac{a_2b_2+a_3b_3+\ldots+a_qb_q}{a_1}, 0, 0, \ldots, 0) =$$
$$= (\frac{a_1b_1+a_2b_2+a_3b_3+\ldots+a_qb_q}{a_1}, 0, 0, \ldots, 0) = (0, 0, \ldots, 0)$$

в силу того, что s – решение уравнения (4). Отсюда получаем, что

$$s = -\frac{b_2}{a_1}s_1 - \frac{b_3}{a_1}s_2 - \ldots - \frac{b_q}{a_1}s_{q-1},$$

а это, в силу произвольности s означает, что TSS – базис множества всех решений (4).

Сложность построения одной комбинации требует q шагов, а всех таких векторов $q-1$. Следовательно, сложность построения всего TSS $O(q^2)$. ■

Пример 2. Найти базис множества решений ЛОУ $1,2x_1 + 1,5x_2 + 0,7x_3 + 0,4x_4 = 0$.
Решение. Согласно доказанному выше, этот базис составляют векторы

$$s_1 = (1,5; -1, 2; 0; 0),\ s_2 = (0, 7; 0; -1, 2; 0),\ s_3 = (0, 4; 0; 0; -1, 2).$$

Например, решение $s = (4, -1, 1, -10)$ в этом базисе имеет представление $s = \frac{1}{1,2}s_1 - \frac{1}{1,2}s_2 + \frac{10}{1,2}s_3$. ♠

5.1. Системы уравнений

Системы однородных уравнений. Пусть дана СЛОУ S вида (1). Рассмотрим множество векторов канонического базиса $M_0' = \{e_1, \ldots, e_n\}$ и первое уравнение $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_n = 0$ системы S. Построим базис $B_1 = \{e_1, \ldots, e_m\}$ множества всех решений этого ЛОУ описанным выше способом. Возьмем функцию $L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q$ и рассмотрим ЛОУ вида

$$L_2(e_1)y_1 + L_2(e_2)y_2 + \ldots + L_2(e_m)y_m = 0. \tag{5}$$

Заметим, что если все $L_2(e_i) = 0$, то уравнение $L_2(x)$ линейно выражается через $L_1(x)$ и его можно удалить из СЛОУ S. Поэтому будем предполагать, что все уравнения в S линейно независимы.

Найдем базис $B' = \{r_1, r_2, \ldots, r_{m-1}\}$ множества решений ЛОУ (5) и построим по векторам из B' соответствующие комбинации векторов из B_1. Обозначим это множество $M = \{s_1, s_2, \ldots, s_{m-1}\}$.

Лемма 2. *Множество M является базисом множества решений СЛОУ*

$$S_1 = \left\{ \begin{array}{lcccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0. \end{array} \right. \tag{6}$$

Доказательство. Очевидно, что все элементы из M являются решениями СЛОУ S_1. Пусть $x = (x_1, \ldots, x_q)$ – произвольное решение СЛОУ (6), тогда в силу леммы 1

$$x = d_1e_1 + \ldots + d_me_m,$$

где $e_i \in B_1,\ i = 1, \ldots, m$. Подставляя x в $L_2(x)$, получаем ЛОУ

$$d_1L_2(e_1) + \ldots + d_mL_2(e_m) = c_1d_1 + \ldots + c_md_m = 0, \tag{7}$$

т. е. вектор $(d_1, d_2, \ldots, d_m)$ является решением ЛОУ (7) и, следовательно, он представляется в виде линейной комбинации векторов из B':

$$(d_1, \ldots, d_m) = f_1 r_1 + \ldots + f_{m-1} r_{m-1}.$$

Но тогда

$$x = d_1 e_1 + \ldots + d_m e_m = f'_1 s_1 + \ldots + f'_{m-1} s_{m-1},$$

а это значит, что x представляется в виде линейной комбинации векторов из M. В силу произвольности вектора x получаем справедливость леммы. ■

Теорема 7. *Пусть M – TSS, построенное описанным выше способом для СЛОУ S вида (1), тогда M является базисом множества всех решений этой СЛОУ.*

Сложность построения базиса пропорциональна величине sq^2, где s – число уравнений, а q – количество неизвестных в СЛОУ.

Доказательство индукцией по числу k уравнений в СЛОУ S.

Базис индукции ($k = 2$) имеет место в силу леммы 2.

Предположим, что теорема верна для всех $k < s$. Тогда TSS решений СЛОУ S', состоящей из первых $s - 1$ уравнений, в силу предположения индукции является базисом множества решений S'.

Повторяя рассуждения, аналогичные рассуждениям, которые использовались при доказательстве леммы 2, получаем справедливость утверждения теоремы. Оценка временной сложности, приведенная в формулировке теоремы, очевидным образом вытекает из построений TSS и леммы 1. ■

Пример 3. Найти базис множества решений СЛОУ

$$S = \left\{ \begin{array}{rcrcrcrcrcl} 2,1x_1 & + & 1,1x_2 & + & 0x_3 & + & x_4 & + & 2,3x_5 & = & 0, \\ 0,5x_1 & + & 0,2x_2 & + & 1x_3 & + & 0x_4 & - & 1,3x_5 & = & 0, \\ 0,1x_1 & + & 0x_2 & + & 2,5x_3 & + & 2x_4 & + & 0x_5 & = & 0. \end{array} \right.$$

Решение. Базис множества решений первого ЛОУ этой системы:

$$s_1 = (1, 1; -2, 1; 0; 0; 0),\ s_2 = (0; 0; 1; 0; 0),\ s_3 = (1; 0; 0; -2, 1; 0),\ s_4 = (2, 3; 0; 0; 0; -2, 1).$$

Находим $L_2(s_1) = 0,13$, $L_2(s_2) = 1$, $L_2(s_3) = 0,5$, $L_2(s_4) = 3,88$ и строим ЛОУ $0,13y_1 + 1y_2 + 0,5y_3 + 3,88y_4 = 0$. Базис множества решений этого ЛОУ состоит из векторов $r_1 = (1; -0,13; 0; 0)$, $r_2 = (0,5; 0; -0,13; 0)$, $r_3 = (3,88; 0; 0; -0,13)$. Получаем векторы множества решений для первых двух уравнений из S, соответствующие векторам r_1, r_2, r_3:

$$\begin{array}{c} s'_1 = s_1 - 0,13s_2 = (1, 1; -2, 1; -0, 13; 0; 0), \\ s'_2 = 0,5s_1 - 0,13s_3 = (0, 42; -1, 05; 0; 0, 273; 0), \\ s'_3 = 3,88s_1 - 0,13s_4 = (3, 969; -8, 148; 0; 0; 0, 273). \end{array}$$

Находим значения $L_3(s'_1) = -0,215$, $L_3(s'_2) = 0,588$, $L_3(s'_3) = 0,3969$, строим ЛОУ $0,215y_1 + 0,588y_2 + 0,3969y_3 = 0$ и получаем его решения: $r'_1 = (0,588; 0,215; 0)$, $r'_2 = (0,3969; 0; 0,215)$. Строим соответствующие им векторы базиса множества решений СЛОУ S:

$$\begin{array}{c} m_1 = 0,588s'_1 + 0,215s'_2 = (0,7371; -1,46055; -0,7644; 0,058695; 0), \\ m_2 = 0,3969s'_1 + 0,215s'_3 = (1,289925; -2,58531; -0,051597; 0; 0,058695). \spadesuit \end{array}$$

Системы неоднородных уравнений. Пусть дано СЛНУ вида (2). Имея в распоряжении метод решения СЛОУ, преобразуем СЛНУ (2) к виду

$$S' = \left\{ \begin{array}{rcrcrcrcrcl} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & - & b_1x_0 & = 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & - & b_2x_0 & = 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & - & b_sx_0 & = 0. \end{array} \right. \quad (8)$$

Пусть $B = \{s_1, s_2, \ldots, s_m\}$ – базис множества решений СЛОУ (8). Разделим множество B на два подмножества

$$B_0 = \{s_i \in B : s_i = (d_{i1}, d_{i2}, \ldots, d_{iq}, 0\},$$
$$B_1 = \{s_j \in B : s_j = (d_{j1}, d_{j2}, \ldots, d_{jq}, d\},$$

где $d \neq 0$. Если в множестве B_1 нет вектора, у которого $d = 1$, то разделим все координаты вектора $s_j \in B_1$ на число d. Получим вектор $s'_j = (\frac{d_{j1}}{d}, \frac{d_{j2}}{d}, \ldots, \frac{d_{jq}}{d}, 1)$. Тогда общее решение СЛНУ (2) принимает вид

$$x = s''_j + c_1 s'_{i1} + \ldots + c_r s'_{ir},$$

где $s''_j = (\frac{d_{j1}}{d}, \frac{d_{j2}}{d}, \ldots, \frac{d_{jq}}{d})$ – частное решение СЛНУ (2), а $s'_{ik} = (d_{i1}, d_{i2}, \ldots, d_{iq})$ получаются из векторов множества B_0 путем отбрасывания последней координаты.

Действительно, множество B_0 составляют векторы, являющиеся базисом множества всех решений приведенной СЛОУ для СЛНУ (2) в силу теоремы 7. А векторы из множества B_1 – это частные решения этой СЛНУ. По любому из векторов этого множества в случае необходимости строим вектор s''_j. Однако стоит обратить внимание на порядок построения векторов множества B. Проиллюстрируем это на примере.

Пример 4. Построить множество решений СЛНУ

$$S' = \left\{ \begin{array}{lcccccccccc} L'_1(x) & = & - & 3x & - & 4y & + & 5z & - & 6u & = 2, \\ L'_2(x) & = & & 2x & + & 3y & - & 2z & + & u & = -1, \\ L'_3(x) & = & & x & - & y & - & z & + & 2u & = 1. \end{array} \right.$$

Решение. СЛОУ S для системы S' принимает вид:

$$S = \left\{ \begin{array}{lcccccccccccc} L_1(x) & = & - & 3x & - & 4y & + & 5z & - & 6u & - & 2x_0 & = & 0, \\ L_2(x) & = & & 2x & + & 3y & - & 2z & + & u & + & x_0 & = & 0, \\ L_3(x) & = & & x & - & y & - & z & + & 2u & - & x_0 & = & 0. \end{array} \right.$$

Строим базис множества решений СЛОУ S вышеприведенным методом. Для первого из уравнений системы TSS состоит из таких векторов:

$$s_1 = (-4, 3, 0, 0, 0), s_2 = (5, 0, 3, 0, 0), s_3 = (-2, 0, 0, 1, 0), s_4 = (-2, 0, 0, 0, 3).$$

Заметим, что первые три вектора составляют базис множества решений ЛОУ, соответствующего первому уравнению системы S', а последний вектор, преобразованный к виду $(-\frac{2}{3}, 0, 0, 0, 1)$ – частное решение этого уравнения. Если упорядочивать векторы-решения СЛОУ S на каждом шаге вычислений так, чтобы в начале стояли векторы-решения приведенной СЛОУ для S', то это сразу приводит к базису множества решений без дополнительных преобразований построенных векторов.

Следуя этому порядку, находим значения $L_2(x)$ на векторах s_1, s_2, s_3, s_4: 1,4,-3,-1, откуда получаем уравнение $w_1 + 4w_2 - 3w_3 - w_4 = 0$. Решения этого уравнения имеют вид $(4, -1, 0, 0)$, $(-3, 0, -1, 0)$ $(-1, 0, 0, -1)$, а этим решениям соответствуют такие решения системы первых двух уравнений (после сокращения на их НОД координат):

$$s'_1 = 4s_1 - s_2 = (-7, 4, -1, 0, 0),$$
$$s'_2 = -3s_1 - s_3 = (14, -9, 0, -1, 0),$$
$$s'_3 = -s_1 - s_4 = (2, -1, 0, 0, -1).$$

Значения $L_3(x)$ на этих векторах равны $-10, 21, 4$, откуда получаем уравнение $-10w_1 + 21w_2 + 4w_3 = 0$. Решения этого уравнения имеют вид $(21, 10, 0)$, $(4, 0, 10)$, а этим решениям соответствуют такие решения всей системы (после сокращения на -1 – НОД их координат):

$$(7, 6, 21, 10, 0), \quad (4, -3, 2, 0, 5).$$

Таким образом, базис множества решений СЛНУ состоит из векторов $B_0 = \{s_1 = (7, 6, 21, 10, 0)\}$ и $B_1 = \{s_2 = (4, -3, 2, 0, 5)\}$. Тогда $s''_2 = (\frac{4}{5}, -\frac{3}{5}, \frac{2}{5}, 0)$ – частное решение СЛНУ. Следовательно, общее решение имеет вид

$$x = s''_2 + cs'_1,$$

где $c \in \mathcal{D}$ – произвольное, а $s'_1 = (7, 6, 21, 10)$. ♠

Заметим, что поскольку при построении базиса множества решений системы уравнений не используется операция деления, то когда коэффициенты системы являются целыми числами, то и базисные решения будут целочисленными. Кроме того, поскольку базисные решения составляют фундаментальную систему решений, то отсюда следует линейная независимость векторов из TSS.

5.2. Системы неравенств

Системы однородных неравенств. Рассмотрим СЛОН

$$S' = \left\{ \begin{array}{rcccccccccc} L'_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & \geq & 0, \\ L'_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & \geq & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L'_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & \geq & 0. \end{array} \right. \tag{9}$$

Преобразуем эту систему к СЛОУ, вводя s дополнительных неизвестных:

$$S = \left\{ \begin{array}{rcccccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & - & y_1 & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & - & y_2 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & - & y_s & = & 0. \end{array} \right.$$

Построим базис B множества решений этой СЛОУ следуя порядку генерации, определенному выше, и разделим это множество на два подмножества $B_0 = \{s_1, \ldots, s_k\}$ и $B_1 = \{r_1, \ldots, r_m\}$, где в первое множество включаются векторы, последние s координат которых равны 0, а во второе множество включаются векторы, последние s координат которых неотрицательны. Отбрасывая в векторах множеств B_0 и B_1 последние s координат, получаем базис множества решений исходной СЛОН.

Пример 5. Построить множество решений СЛОН

$$S' = \left\{ \begin{array}{rcccccccccc} L'_1(x) & = & - & 3x & - & 4y & + & 5z & - & 6u & \geq & 0, \\ L'_2(x) & = & & 2x & + & 3y & - & 2z & + & u & \geq & 0, \\ L'_3(x) & = & & x & - & y & - & z & + & 2u & \geq & 0. \end{array} \right.$$

Решение. СЛОУ S для системы S' принимает вид:

$$S' = \left\{ \begin{array}{rcccccccccccccc} L_1(x) & = & - & 3x & - & 4y & + & 5z & - & 6u & - & y_1 & + & 0 & + & 0 & = & 0, \\ L_2(x) & = & & 2x & + & 3y & - & 2z & + & u & + & 0 & - & y_2 & + & 0 & = & 0, \\ L_3(x) & = & & x & - & y & - & z & + & 2u & + & 0 & + & 0 & - & y_3 & = & 0. \end{array} \right.$$

Строим базис множества решений СЛОУ S' вышеприведенным методом. Для первого из уравнений системы TSS состоит из таких векторов:

$$(-4,3,0,0,0,0,0),\ \ (5,0,3,0,0,0,0),\ \ (-6,0,0,3,0,0,0),$$
$$(-1,0,0,0,3,0,0),\ \ (0,0,0,0,0,1,0),\ \ (0,0,0,0,0,0,1).$$

Первые три вектора составляют базис множества решений ЛОУ, соответствующего первому неравенству системы S, а последние три – частные решения первого неравенства этой системы.

Следуя вышеустановленному порядку решений, находим значения $L_2(x)$ на этих векторах равны 1,4,-9,-2,-1,0, откуда получаем уравнение $w_1 + 4w_2 - 9w_3 - 2w_4 - w_5 + 0w_6 = 0$. Решения этого уравнения имеют вид (4,-1,0,0,0,0), (9,0,1,0,0,0), (2,0,0,1,0,0), (1,0,0,0,1,0), (0,0,0,0,0,1), а этим решениям соответствуют такие решения системы первых двух уравнений (после сокращения на их НОД):

$$(-7,4,-1,0,0,0,0),\ (14,-9,0,-1,0,0,0),\ (-3,2,0,0,1,0,0),$$
$$(-4,3,0,0,0,1,0),\ (0,0,0,0,0,0,1).$$

Значения $L_3(x)$ на этих векторах равны $-10, 21, -5, -7, -1$, откуда получаем уравнение $-10w_1 + 21w_2 - 5w_3 - 7w_4 - w_5 = 0$. Решения этого уравнения имеют вид (21,10,0,0,0), (-1,0,2,0,0), (-7,0,0,10,0), (-1,0,0,0,10), а им соответствуют такие решения

$$(-7,-6,-21,-10,0,0,0),\ (1,0,1,0,2,0,0),\ (9,2,7,0,0,10,0),\ (7,-4,1,0,0,0,10).$$

В результате получаем

$$B_0 = \{(7,6,21,10,0,0,0)\},$$
$$B_1 = \{(1,0,1,0,2,0,0),\ (9,2,7,0,0,10,0), (7,-4,1,0,0,0,10)\},$$

а отсюда базис множества решений всей системы:

$$(-7,-6,-21,-10),\ (1,0,1,0),\ (9,2,7,0),\ (7,-4,1,0).\ \spadesuit$$

Системы неоднородных неравенств (СЛНН) сводятся к СЛОУ путем введения $s+1$ дополнительных неизвестных, где s – количество неравенств в СЛНН. Пусть имеем СЛНН

$$S=\left\{\begin{array}{rcccccccccc} L'_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & \geq & b_1, \\ L'_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & \geq & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L'_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & \geq & b_s. \end{array}\right.$$

Этой СЛНН соответствует СЛОУ вида

$$S'=\left\{\begin{array}{rcccccccccccc} L'_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & - & y_1 & - & b_1x_0 & = & 0, \\ L'_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & - & y_2 & - & b_2x_0 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ L'_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & - & y_s & - & b_sx_0 & = & 0. \end{array}\right.$$

Построим базис B множества решений СЛОУ S исходя из системы S', и его подмножества B_0 и B_1. Если среди векторов из множества B_1 нет векторов, у которых последние s координат имеют разные знаки, то СЛНН S совместна. Поскольку значения дополнительных неизвестных должны быть неотрицательными, то в противном случае получить такие решения из векторов множества B_1 невозможно, что означает несовместность СЛНН.

Построение базиса множества решений СЛОН без введения дополнительных неизвестных. Поскольку процесс решения СЛНН сводится к решению СЛОН с увеличением размерности системы, то естественно возникает вопрос: нельзя ли найти способ решения СЛОН без увеличения ее размерности?

Анализ процесса построения базиса множества решений СЛОН с введением дополнительных неизвестных показывает, что явно вводить дополнительные неизвестные нет необходимости. Для правильности построения базиса множества решений СЛОН необходимо вводить дополнительно всего два вектора той же размерности, что и размерность СЛОН. Объясним этот способ с помощью примера.

Пример 6. Рассмотрим СЛОН из примера 5

$$S=\left\{\begin{array}{rcccccccccc} L'_1(x) & = & - & 3x & - & 4y & + & 5z & - & 6u & \geq & 0, \\ L'_2(x) & = & & 2x & + & 3y & - & 2z & + & u & \geq & 0, \\ L'_3(x) & = & & x & - & y & - & z & + & 2u & \geq & 0. \end{array}\right.$$

Строим TSS для первого неравенства $L_1(x)\geq 0$:

$$(-4,3,0,0),\ \ (5,0,3,0),\ \ (-6,0,0,3).$$

Дополним это множество двумя векторами вида (-1,0,0,0) и (0,0,0,0) или векторами (1,0,0,0) и (0,0,0,0) в зависимости от того, какой знак у a_{11}. Если -1 и a_{11} имеют одинаковые знаки, то выбирается первый вариант, а если они имеют разные знаки, то выбирается второй вариант. Это делается с целью обеспечения положительного значения координаты, которая соответствует первому дополнительному неизвестному (это правило выбора сохраняется и в дальнейшем). В данном случае $a_{11}=-3$ и множество векторов принимает вид:

$$(-4,3,0,0),\ \ (5,0,3,0),\ \ (-6,0,0,3),\ \ (-1,0,0,0),\ \ (0,0,0,0).$$

Заметим, что первые три вектора составляют базис множества решений ЛОУ, соответствующего первому неравенству системы S, четвертый – частное решение первого ЛНУ этой системы, а пятый – заготовка на следующее неравенство.

Следуя вышеустановленному порядку решений, находим значения $L_2(x)$ на первых четырех векторах: 1,4,-9,-2, а на нулевом векторе значение $L_2(x)$ полагаем равным -1. Откуда получаем уравнение $w_1+4w_2-9w_3-2w_4-w_5=0$. Решения этого уравнения имеют вид (4,-1,0,0,0), (9,0,1,0,0), (2,0,0,1,0), (1,0,0,0,1) (поскольку -1 и 1 разных знаков), а этим решениям соответствуют такие решения системы первых двух неравенств:

$$(-7,4,-1,0),\ (14,-9,0,-1),\ (-3,2,0,0),\ (-4,3,0,0).$$

Находим значения $L_3(x)$ таким же образом: $-10, 21, -5, -7, -1$, откуда получаем уравнение $-10w_1 + 21w_2 - 5w_3 - 7w_4 - w_5 = 0$. Поскольку $L_3(x) \geq 0$ последнее неравенство в системе, то дополняем эти векторы только одним (-1,0,0,0) (знаки -1 и -10 одинаковые) так как заготовки на следующее неравенство делать не нужно. Решения полученного уравнения имеют вид (21,10,0,0,0), (-1,0,2,0,0), (-7,0,0,10,0), (-1,0,0,0,10), а им соответствуют решения:

$$s_1 = (-7,-6,-21,-10), s_2 = (1,0,1,0), s_3 = (9,2,7,0), s_4 = (7,-4,1,0),$$

которые составляют базис множества решений всей системы. Это значит, что любое решение СЛОН S представляется в виде линейной неотрицательной комбинации векторов s_1, s_2, s_3, s_4.

Если проследить порядок построения базиса, то можно определить, что решение (-7,-6,-21,-10) – решение СЛОУ, которая соответствует СЛОН, а остальные решения – это частные решения СЛОН. Следует заметить, что решение (-7,-6,-21,-10) можно сократить на -1, поскольку оно решение СЛОУ. ♠

Резюмируя сказанное, отметим, что дополнительные векторы и значения на этих векторах участвуют во всех неравенствах, кроме последнего. А при обработке последнего неравенства к TSS первых $s-1$ неравенств добавляется только один вектор с первой ненулевой координатой (-1 или 1) и остальными нулевыми координатами. Метод строит базис множества всех решений.

Эксперименты показали, что TSS-метод решения систем линейных ограничений эффективно работает для систем с разреженными матрицами, коэффициенты которых имет небольшие нормы. Этот метод работает без каких-либо предварительных условий с любыми матрицами, но в процессе вычислений наблюдается быстрый рост координат векторов. Это требует время от времени вычисления наибольшего общего делителя координат векторов и сокращения их координат на этот общий делитель. Однако, несмотря на этот недостаток (который присущ и другим традиционным методам), точность выисления решений выше по сравнению, например, с алгоритмом Жордана-Гаусса.

6. Диофантовы ограничения

В теоретическом плане диофантовы уравнения пользовались особым вниманием математиков аж до 70-х голов прошлого столетия в связи с 10-й проблемой Гильберта, сформулированной в начале 20-го столетия [1]:

Пусть дано диофантово уравнение с рациональными коэффициентами и произвольным конечным числом неизвестных;

необходимо указать способ, при помощи которого путем выполнения конечного числа операций можно определить, имеет, или нет, решения это уравнение в множестве целых чисел.

Способ, о поиске которого шла речь в этой проблеме, теперь понимают как **алгоритм**. В начале 20-го столетия теория алгоритмов еще не была построена и речь могла идти только о положительном решении проблемы. Большие трудности, возникшие при решении этой проблемы, привели к мнению о неразрешимости 10-й проблемы Гильберта и в 1970 году Ю. В. Матиясевичем была строго доказана алгоритмическая неразрешимость этой проблемы [14].

Хорошо известно, что неразрешимость некоторой проблемы для определенного класса объектов, не исключает ее разрешимости для подкласса объектов из этого класса. Таким подклассом в классе диофантовых уравнений являются линейные диофантовы уравнения и системы таких уравнений над множеством натуральных чисел. Для этого подкласса систем

разрешимость была установлена еще в конце 19 столетия Жорданом [18] и Гильбертом [19], а позже были предложены алгоритмы построения базиса множества всех решений.

В данном разделе рассматриваются алгоритмы построения минимального порождающего множества решений и базиса множества решений для СЛОДУ, критерии совместности, базирующиеся на этих алгоритмах для СЛОДУ, СЛНДУ над полем и кольцом вычетов по модулю простого и составного числа, над множествами натуральных и целых чисел.

В настоящее время имеется несколько методов и алгоритмов решения ЛОДУ, СЛОДУ и СЛНДУ над множеством натуральных чисел, которые с успехом применяются на практике. Все эти методы и алгоритмы в основном разрабатывались для решения задач ассоциативно-коммутативной унификации в эквациональных теориях первого порядка. Это методы Фортенбахера [45], Комона [46], Ромеуфа [44], Контежан-Деви [47], Потье [51], Доменжуда [49], Филгуэйрас-Томас [50].

Далее рассматривается TSS-метод и упомянутые методы решения проблемы выполнимости линейных ограничений над дискретными областями.

6.1. Язык линейных уравнений над кольцом вычетов

Кольцом вычетов Z_m по модулю числа m называется алгебра $Z_m = (D = \{0, 1, \ldots, m-1\}, \Omega = \{+, \cdot, -, ^{-1}, 0, 1\})$, где $+$ и $\cdot$ являются бинарными ассоциативными, коммутативными и дистрибутивными операциями сложения и умножения по модулю m, операции $-$ и $^{-1}$ – унарные операции взятия противоположного и обратного элемента относительно операций $+$ и $\cdot$ соответственно, 0 и 1 – нульарные операции – аддитивный нуль и мультипликативная единица. Заметим, что операция взятия обратного элемента в кольце Z_m в общем случае является частичной, поскольку кольцо Z_m может содержать делители нуля и для них эта операция не определена.

На основании законов для операций в кольце Z_m вытекает справедливость такого тождества:

$$(\forall x, y \in Z_m)\ x + y = 0 = m \ \rightarrow\ x = -y.$$

Из этого тождества следует, что когда в кольце Z_m $x = m - y$, то $-y = x - m$, что дает возможность заменять положительное число x на отрицательное число $-y = x - m$ и наоборот. Такие элементы x и $-y$ будем называть дополнениями (x дополняет $-y$ и наоборот).

Кольцо вычетов Z_m называется **примарным**, если модуль m является степенью простого числа p, т. е. $m = p^t$, где $t > 1,\ t \in \mathcal{N}$. Поскольку m – не объязательно простое число, то в кольце Z_m при $a \neq 0$ сравнение $ax \equiv b\ (mod\ m)$ не всегда имеет решение. Оно будет иметь решение, если НОД$(a, m) = 1$ или НОД$(a, m) = d$ и d – делитель числа b.

С учетом свойств кольца Z_m рассмотрим систему линейных уравнений

$$S = \left\{ \begin{array}{rcccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & b_s, \end{array} \right. \tag{10}$$

где $a_{ij}, b_i \in Z_m$.

Пусть модуль имеет разложение на простые множители $m = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, где $p_1 < p_2 <$

$\ldots < p_r$. Тогда системе S соответствует эквивалентная ей система с $r \cdot s$ уравнениями вида

$$S' = \left\{ \begin{array}{l} \left\{ \begin{array}{lllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & b_s, \end{array} \right. (mod\ p_1^{k_1}) \\ \left\{ \begin{array}{lllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & b_s, \end{array} \right. (mod\ p_2^{k_2}) \\ \vdots \\ \left\{ \begin{array}{lllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & = & b_s. \end{array} \right. (mod\ p_r^{k_r}) \end{array} \right. \quad (11)$$

Эквивалентность систем S и S' очевидна. Действительно, если x – решение системы S, то это решение будет решением и каждой из подсистем по модулю $p_i^{k_i}$, поскольку модуль m делится на каждое из чисел $p_i^{k_i}$, $i = 1, 2 \ldots, r$. Если же x – решение системы S', то оно будет решением каждой из ее подсистем по модулю $p_i^{k_i}$, а поэтому и решением системы S по модулю m, поскольку числа $p_i^{k_i}$ взаимно простые и их произведение равно m.

Перейдем от системы S' к системе однородных уравнений

$$S'' = \left\{ \begin{array}{l} \left\{ \begin{array}{llllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & -b_1x_0 & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & -b_2x_0 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & -b_sx_0 & = & 0, \end{array} \right. (mod\ p_1^{k_1}) \\ \left\{ \begin{array}{llllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & -b_1x_0 & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & -b_2x_0 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & -b_sx_0 & = & 0, \end{array} \right. (mod\ p_2^{k_2}) \\ \vdots \\ \left\{ \begin{array}{llllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & -b_1x_0 & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & -b_2x_0 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & -b_sx_0 & = & 0. \end{array} \right. (mod\ p_r^{k_r}) \end{array} \right. \quad (12)$$

Пусть x – решение системы вида

$$S_1 = \left\{ \begin{array}{llllllllllll} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & -b_1x_0 & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & -b_2x_0 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sq}x_q & -b_sx_0 & = & 0. \end{array} \right. (mod\ p_1^{k_1})$$

Рассмотрим вектор $m_1 x$, где $m_1 = \frac{m}{p_1^{k_1}}$, который будет решением системы S''. Действительно, для второй системы S_2, аналогичной предыдущей по модулю $p_2^{k_2}$, получаем для любого ее уравнения L_i

$$L_i(m_1 x) = m_1 L_i(x) = m_1 d_i \equiv 0\ (mod\ p_2^{k_2}),$$

$i = 1, 2 \ldots, s$, поскольку m_1 кратно $p_2^{k_2}$, а d_i кратно $p_1^{k_1}$.

Аналогично, если y – решение S_2, то $m_2 y$, где $m_2 = \frac{m}{p_2^{k_2}}$, будет решением системы S'' и т. д. для любой из систем $S_3, \ldots, S_r$. Тогда общее решение системы S'' принимает вид

$$x = m_1 x_1 + m_2 x_2 + \ldots + m_r x_r, \tag{13}$$

где x_i – решения системы S_i.

Обозначим $e_i = m_i x_i$, где x_i – решение системы $S_i,\ i = 1, 2, \ldots, r$. Покажем, что эти векторы линейно независимы над кольцом Z_m. Для этого представим векторы e_i в координатной форме

$$e_1 = (c_{11}, c_{12}, \ldots, c_{1q}), e_2 = (c_{21}, c_{22}, \ldots, c_{2q}), \ldots, e_k = (c_{k1}, c_{k2}, \ldots, c_{kq})$$

и допустим, что существуют числа $a_1, a_2, \ldots, a_k$, где $a_i < p_i^{k_i}$, такие, что

$$a_1 e_1 + a_2 e_2 + \ldots + a_k e_k \equiv 0 \ (mod\ m)$$

или, что тоже самое,

$$a_2 e_2 + \ldots + a_k e_k \equiv b_1 e_1 \ (mod\ m),$$

где b_1 дополнение a_1 в кольце Z_m и $a_1 e_1 \not\equiv 0 \ (mod\ m)$. Принимая во внимание координатную форму векторов $e_i, i = 1, 2, \ldots . k$, получаем систему

$$S_1' = \left\{ \begin{array}{ccccccc} a_2 m_2 c_{21}' & + & \ldots & + & a_k m_k c_{k1} & \equiv b_1 m_1 c_{11}', \\ a_2 m_2 c_{22}' & + & \ldots & + & a_k m_k c_{k2} & \equiv b_1 m_1 c_{12}', \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_2 m_2 c_{2q}' & + & \ldots & + & a_k m_k c_{kq} & \equiv b_1 m_1 c_{1q}', \end{array} \right. (mod\ p_1^{k_1}),$$

где c_{ij}' – координаты векторов $x_i,\ i = 1, 2, \ldots, r$.

Из сравнений системы S_1' вытекает, что левая часть каждого из сравнений кратна $p_1^{k_1}$, как и сам модуль m. Но тогда система S_1' будет иметь решение только в случае кратности числа $b_1 m_1$ (и тем самым кратности $a_1 c_{1i}$) числу $p_1^{k_1}$. Но если все числа $a_1 c_{1i}$ кратны $p_1^{k_1}$, то получаем $a_1 e_1 \equiv 0 \ (mod\ m)$, что противоречит предположению $a_1 e_1 \not\equiv 0 \ (mod\ m)$. Полученное противоречие показывает несовместность системы S_1', а это означает линейную независимость совокупности векторов $e_1, e_2, \ldots, e_k$.

Таким образом, достаточно уметь решать систему вида S'. А решение такой системы сводится к решению систем либо в полях вычетов по модулю простого числа (случай когда показатель степени $k_i = 1$ для некоторого $i \in [1, r]$ и в этом случае кольцо Z_m становится полем), либо к решению систем в примарных кольцах по модулю степени простого числа.

Рассмотрим отдельно каждый из этих случаев.

6.1.1. Язык линейных уравнений над полем вычетов

Полем вычетов по модулю простого числа p называется алгебра $F_p = (D = \{0, 1, \ldots, p-1\}, \Omega = \{+, \cdot, -, ^{-1}, 0, 1\})$, где $+$ и $\cdot$ являются бинарными ассоциативными, коммутативными и дистрибутивными операциями сложения и умножения по модулю p, операции - и $^{-1}$ – унарные операции взятия противоположного и обратного элемента относительно операций $+$ и $\cdot$ соответственно, 0 и 1 – нульарные операции – аддитивный нуль и мультипликативная единица.

Поскольку p – простое число, то в поле F_p при $a \neq 0$ сравнение $ax \equiv b \ (mod\ p)$ всегда имеет решение и это решение единственно.

Принимая во внимание эти свойства поля F_p, рассмотрим TSS-метод [6] решения систем линейных диофантовых уравнений в этом поле.

TSS-метод решения ЛОДУ. Пусть дано ЛОДУ

$$L(x) = a_1x_1 + \ldots + a_ix_i + \ldots + a_nx_n = 0, \tag{14}$$

где $a_i, x_i \in F_p,\ i = 1, \ldots, n$. Допустим, что $a_i \neq 0$, тогда имеет место

Лемма 3. *Если $c = (c_1, \ldots, c_n)$ – решение ЛОДУ (14) в поле F_p, то оно будет решением ЛОДУ $a_1x_1 + \ldots - b_ix_i + \ldots + a_nx_n = 0$, где $-b_i$ – дополнение коэффициента a_i.*

Доказательство. По условию леммы имеем $a_1c_1 + \ldots + a_ic_i + \ldots + a_nc_n = 0$, но тогда

$$a_1c_1 + \ldots - b_ic_i + \ldots + a_nc_n = -pc_i + a_1c_1 + \ldots + a_ic_i + \ldots + a_nc_n = 0.$$

Заметим, что если $c = k \cdot c'$, где $k =$НОД$(c_1, c_2, \ldots, c_n)$, то $c' = (c'_1, c'_2, \ldots, c'_n)$ также будет решением (14). Действительно, если c – решение, то

$$a_1c_1 + a_2c_2 + \ldots + a_nc_n = k(a_1c'_1 + a_2c'_2 + \ldots + a_nc'_n) = 0$$

и поскольку $k \neq 0$, то $a_1c'_1 + a_2c'_2 + \ldots + a_nc'_n = 0$. ■

Рассмотрим множество векторов канонического базиса $M_0 = \{e_1, \ldots, e_n\}$ и функцию $L_1(x) = a_1x_1 + a_2x_2 + \ldots + a_nx_n$ ЛОДУ (14). Заменим в функции $L(x)$ первый ненулевой коэффициент a_k его отрицательным дополнением $-b_k$ и построим множество векторов

$$B = \{(0, \ldots, a_j, 0, \ldots, 0, b_k, 0, \ldots, 0)\} \cup M_0,$$

где $M_0 = \{e_r : L_1(e_r) = 0\}$, $a_j \neq 0$, а b_k является j-й координатой в векторах из B. Причем если для некоторого a_j НОД$(a_j, b_k) \neq 1$, то сократим координаты такого вектора на этот общий делитель (что возможно в силу леммы 3). Таким образом, можно считать, что все векторы в множестве B таковы, что a_j и b_k взаимно просты, а множество B строится путем комбинирования дополнения первого ненулевого коэффициента, взятого с отрицательным знаком, с остальными ненулевыми коэффициентами и пополненное векторами канонического базиса, которые соответствуют нулевым коэффициентам ЛОДУ (14). Построенное таким образом множество, как и раньше, будем называть TSS. Очевидно, что векторы из множества B являются решениями ЛОДУ (14).

Лемма 4. *Пусть $d = (0, \ldots, 0, d_i, 0, \ldots, 0, d_j, \ldots, 0)$ – решение ЛОДУ (14), тогда если $d \notin B$, то d представляется в виде неотрицательной линейной комбинации векторов из B.*

Доказательство. Если $d = (0, \ldots, 0, d_i, 0, \ldots, 0, d_j, 0, \ldots, 0) \notin B$, то возможны два случая.

Случай 1. В множестве B существуют векторы $s_1 = (0, \ldots, 0, a'_i, 0, \ldots, 0, a'_k, 0, \ldots, 0)$ и $s_2 = (0, \ldots, 0, a'_j, 0, \ldots, 0, a'_k, 0, \ldots, 0)$, в которых a'_i и a'_j являются k-ми координатами, а a'_k в s_1 и s_2 является соответственно i-й и j-й координатой. Рассмотрим вектор

$$s = us_1 + vs_2 = (0, \ldots, 0, a'_iu + a'_jv, 0, \ldots, 0, d_i, 0, \ldots, 0, d_j, 0, \ldots, 0),$$

где u, v – решения сравнений $a'_k x \equiv d_i \ (mod\ p)$ и $a'_k y \equiv d_j \ (mod\ p)$ соответственно. В поле F_p решения u и v единственны. Покажем, что $a'_i u + a'_j v = 0$. Для этого подставим вектор s в ЛОДУ (14):

$$L(s) = a_k(a'_i u + a'_j v) + a_i d_i + a_j d_j = a_k(a'_i u + a'_j v) + 0 = a_k(a'_i u + a'_j v) = 0$$

и поскольку $a_k \neq 0$, то $a'_i u + a'_j v = 0$, что и требовалось показать.

Случай 2. В множестве B существует вектор $s_1 = (0, \ldots, 0, b_i, 0, \ldots, 0, b_j, 0, \ldots, 0)$. Рассмотрим вектор

$$s = ys_1 = (0, \ldots, 0, c_i, 0, \ldots, 0, yb_j, 0, \ldots, 0),$$

где y – единственное решение сравнения $b_i y \equiv c_i \ (mod\ p)$. Поскольку x и s – решения ЛОДУ (14), то $x - s$ тоже будет решением этого ЛОДУ, т. е. $x - s = (0, \ldots, 0, d_j - b_j y, 0, \ldots, 0)$ и

$$L(x - s) = a_j(d_j - yb_j) = 0.$$

Поскольку $a_j \neq 0$, то $d_j = yb_j$. ■

Теорема 8. *TSS В ЛОДУ (14), построенное комбинированием дополнения первого ненулевого коэффициента, взятого с отрицательным знаком, с остальными ненулевыми коэффициентами и пополненное векторами канонического базиса, которые соответствуют нулевым коэффициентам ЛОДУ (14), является базисом множества всех решений этого ЛОДУ.*

Сложность алгоритма пропорциональна величине l^3, где $l = \max(t, n)$, t – количество двоичных разрядов числа p, а n – количество неизвестных в ЛОДУ.

Доказательство проводится индукцией по числу k ненулевых координат в решении ЛОДУ. Пусть $x = (x_1, x_2, \ldots, x_n)$ – произвольное решение ЛОДУ (14).

Базис индукции. Если $k = 1$, то x должен совпадать с одним из векторов канонического базиса, который по построению является элементом B. При $k = 2$ справедливость теоремы вытекает из леммы 4.

Шаг индукции. Допустим, что теорема верна для всех $2 \leq k < n$ и x имеет n ненулевых координат. Рассмотрим ненулевые координаты x_i, x_j вектора x. Возможны три случая.

Случай 1. В B существует вектор канонического базиса s с i-й координатой, равной единице. Тогда вектор $y = x - x_i s$ будет иметь $n-1$ ненулевых координат. По предположению индукции для этого вектора существует представление $x - x_i s = d_1 e_1 + \ldots + d_r e_r$, где $e_i \in B$, $i = 1, \ldots, r$. Но тогда $x = x_i s + d_1 e_1 + \ldots + d_r e_r$.

Случай 2. В B существует вектор $s = (0, \ldots, 0, b_i, 0, \ldots, 0, b_j, 0, \ldots, 0)$ с ненулевыми координатами b_i и b_j. Рассмотрим вектор

$$x - us = (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_j - ub_j, \ldots, x_n),$$

где u – решение сравнения $b_i u \equiv x_i \ (mod\ p)$. Построенный вектор имеет $n - 1$ ненулевых координат и по предположению индукции получаем

$$x - us = d_1 e_1 + \ldots + d_r e_r \text{ или } x = us + d_1 e_1 + \ldots + d_r e_r.$$

Случай 3. В B существуют векторы $s_1 = (0, \ldots, 0, b_i, 0, \ldots, 0, b_k, 0, \ldots, 0)$ и $s_2 = (0, \ldots, 0, b_j, 0, \ldots, 0, b_k, 0, \ldots, 0)$ с ненулевыми координатами b_i, b_j и b_k. Построим вектор

$$s = b_j s_1 - b_i s_2 = (0, \ldots, 0, b_k b_j, 0, \ldots, 0, -b_k b_i, 0, \ldots, 0),$$

у которого после замены $-b_k b_i$ его положительным дополнением, ненулевыми координатами будут координаты с номерами i и j. Очевидно, что вектор s является решением ЛОДУ (14), тогда согласно лемме 4 он выражается через векторы из множества B. Пусть

$$s' = (0, \ldots, 0, b_j, 0, \ldots, 0, b_i, 0, \ldots, 0) = d_1 e_1 + \ldots + d_r e_r$$

это представление. Далее доказательство сводится к случаю 2.

При получении оценки сложности элементарными считаются операции сложения и вычитания. При построении TSS для ЛОДУ вычисляется $n-1$ вектор и не больше, чем $n-1$ раз вычисляется НОД двух чисел в поле F_p. Построение векторов требует не более чем $n \cdot (n-1)$ шагов, а сложность вычисления НОД с помощью алгоритма Эвклида пропорциональна t^2, где t – разрядность чисел, для которых вычисляется НОД [15]. Суммируя все эти величины, получаем l^3, где $l = \max(t, n)$. ■

Пример 7. Построить базис множества всех решений ЛОДУ $2x_1 + x_2 + 0x_3 + x_4 + 2x_5 = 0$ в поле вычетов F_3.

Решение. Первый ненулевой коэффициент в данном ЛОДУ есть $a_1 = 2$, а его дополнение равно 2 - 3 = -1. Получаем ЛОДУ вида $-x_1 + x_2 + 0x_3 + x_4 + 2x_5 = 0$. Применяя TSS-метод, находим такие базисные решения:

$$e_1 = (1,1,0,0,0),\ e_2 = (1,0,0,1,0),\ e_3 = (2,0,0,0,1),\ e_4 = (0,0,1,0,0).$$

Очевидными решениями данного ЛОДУ являются векторы $c_1 = (1,1,1,1,1)$ и $c_2 = (0,2,0,1,0)$. Представления этих векторов через базисные векторы имеют вид: $c_1 = e_1 + e_2 + e_3 + e_4 = (4\ (mod\ 3), 1, 1, 1, 1) = (1,1,1,1,1)$, $c_2 = 2e_1 + e_2 = (0,2,0,1,0)$. ♠

TSS-метод решения СЛОДУ. Пусть дана приведенная СЛОДУ S для СЛНДУ (10). Рассмотрим первое уравнение $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q = 0$ этой системы. Построим базис $B_1 = \{e_1, \ldots, e_m\}$ множества всех решений этого ЛОДУ описанным выше способом. Возьмем функцию $L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q$ и рассмотрим ЛОДУ вида

$$L_2(e_1)y_1 + L_2(e_2)y_2 + \ldots + L_2(e_m)y_m = 0. \tag{15}$$

Заметим, что если все $L_2(e_i) = 0$, то уравнение $L_2(x)$ линейно выражается через $L_1(x)$ и его можно удалить из СЛОДУ S. Поэтому будем предполагать, что все уравнения в S линейно независимые.

Найдем TSS-методом базис $B' = \{r_1, r_2, \ldots, r_{m-1}\}$ множества решений ЛОДУ (15) и построим по векторам из B' соответствующие комбинации векторов из B_1. Обозначим это множество $M = \{s_1, s_2, \ldots, s_{m-1}\}$.

Лемма 5. *Множество M является базисом множества решений СЛОДУ*

$$S_1 = \left\{ \begin{array}{rcccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0. \end{array} \right. \tag{16}$$

Доказательство. Очевидно, что все элементы из M являются решениями СЛОДУ S_1. Пусть $x = (x_1, \ldots, x_q)$ – произвольное решение СЛОДУ S_1, тогда

$$x = d_1 e_1 + \ldots + d_m e_m,$$

где $e_i \in B_1,\ i = 1, \ldots, m$. Подставляя x в $L_2(x)$, получаем ЛОДУ

$$d_1 L_2(e_1) + \ldots + d_m L_2(e_m) = c_1 d_1 + \ldots + c_m d_m = 0,$$

т. е. вектор $(d_1, d_2, \ldots, d_m)$ является решением ЛОДУ (15) и, следовательно, он представляется в виде неотрицательной линейной комбинации векторов из B':

$$(d_1, \ldots, d_m) = f_1 r_1 + \ldots + f_{m-1} r_{m-1}.$$

Но тогда

$$x = d_1 e_1 + \ldots + d_m e_m = f_1 s_1 + \ldots + f_{m-1} s_{m-1},$$

а это значит, что x представляется в виде неотрицательной линейной комбинации векторов из M. В силу произвольности вектора x получаем справедливость леммы. ■

Теорема 9. *Пусть M – TSS-множество, построенное описанным выше способом для СЛОДУ S вида (1), тогда M является базисом множества всех решений этой СЛОДУ.*

Сложность построения базиса пропорциональна величине sl^3, где s - число уравнений в СЛОДУ, $l = \max(t, q)$, $t = \log p$ – количество двоичных разрядов модуля p, а q – количество неизвестных в СЛОДУ.

Доказательство выполняется индукцией по числу k уравнений в СЛОДУ.

Базис индукции при $k = 2$ имеет место в силу леммы 5.

Шаг индукции. Предположим, что теорема верна для всех $k < q$. Тогда TSS СЛОДУ S', состоящей из первых $s - 1$ уравнений, в силу предположения индукции является базисом множества решений S'.

Повторяя рассуждения, аналогичные рассуждениям, которые использовались при доказательстве леммы 5, получаем справедливость утверждения теоремы. Оценка временной сложности, приведенная в формулировке теоремы, очевидным образом вытекает из теоремы 8. ■

Пример 8. Найти в поле F_3 базис множества решений СЛОДУ

$$S = \left\{ \begin{array}{ccccccccccc} 2x_1 & + & x_2 & + & 0x_3 & + & x_4 & + & 2x_5 & = & 0, \\ x_1 & + & 2x_2 & + & 1x_3 & + & 0x_4 & + & x_5 & = & 0, \\ x_1 & + & x_2 & + & 2x_3 & + & 2x_4 & + & 0x_5 & = & 0. \end{array} \right.$$

Решение. В примере 7 был найден базис множества решений первого ЛОДУ этой системы:

$$e_1 = (1,1,0,0,0),\ e_2 = (1,0,0,1,0),\ e_3 = (2,0,0,0,1),\ e_4 = (0,0,1,0,0).$$

Находим $L_2(e_1) = 0,\ L_2(e_2) = 1,\ L_2(e_3) = 0,\ L_2(e_4) = 1$ и строим ЛОДУ $0y_1 + y_2 + 0y_3 + y_4 = 0y_1 - 2y_2 + 0y_3 + y_4 = 0$. Базис множества решений состоит из векторов $r_1 = (1,0,0,0),\ r_2 = (0,1,0,2),\ r_3 = (0,0,1,0)$. Получаем векторы множества решений для первых двух уравнений из S, соответствующие векторам r_1, r_2, r_3:

$$e'_1 = (1,1,0,0,0),\ e'_2 = (1,0,2,1,0),\ e_3 = (2,0,0,0,1).$$

Находим значения $L_3(e'_1) = 2,\ L_3(e'_2) = 1,\ L_3(e'_3) = 2$, строим ЛОДУ $2y_1 + y_2 + 2y_3 = -y_1 + y_2 + 2y_3 = 0$ и получаем его решения: $r_1 = (1,1,0),\ r_2 = (2,0,1)$. Строим соответствующие им векторы базиса множества решений СЛОДУ S:

$$s_1 = (2,1,2,1,0),\ s_2 = (1,2,0,0,1).\ \spadesuit$$

TSS-метод решения СЛНДУ. Пусть дано ЛНДУ

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = b. \tag{17}$$

Рассмотрим уравнение $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n - b x_0 = 0$, решения которого при $x_0 = 1$ будут решениями (17). Применяя TSS-метод к этому ЛОДУ, получаем

$$s_1 = (b, 0, \ldots, a_1), \ldots, s_n = (0, \ldots, 0, b, a_n).$$

Среди этих решений необходимо выделить такие, у которых $x_0 = 1$. Однако $x_0 \in \{a_1, a_2, \ldots, a_n\}$ и тогда искомыми будут те решения x, которые являются решениями сравнения $a_i x \equiv 1 \ (mod\ p)$. В силу простоты p это сравнение имеет единственное решение, причем это верно для любого $a_i \neq 0,\ i = 1, 2, \ldots, n$. Следовательно, можно выбрать любое $a_i \neq 0$ и для него решать сравнение. Поскольку сравнение $a_i x \equiv 1 \ (mod\ p)$ всегда имеет решение, то и уравнение (17) тоже будет всегда иметь решение.

Пусть $x^1 = (c_1, c_2, \ldots, c_n)$ – некоторое частное решение (17), найденное описанным выше способом, а $B = \{e_1, e_2, \ldots, e_m\}$ – базис множества решений ЛОДУ

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = 0. \tag{18}$$

Лемма 6. *Произвольное решение ЛНДУ (17) представляется в виде* $u = x^1 + \sum\limits_{i=1}^{m} b_i e_i$, *где* x^1 *– частное решение ЛНДУ (17), а* $e_1, \ldots, e_m$ *– базисные векторы множества решений ЛОДУ (18), которое соответствует ЛНДУ (17).*

Доказательство. Пусть $u = (u_1, u_2, \ldots, u_n, 1)$ – произвольное решение ЛНДУ (17), а $B = \{e_1, e_2, \ldots, e_m\}$ – базис множества решений ЛОДУ, которое соответствует (17). Рассмотрим вектор $y = u - x^1 = (d_1, d_2, \ldots, d_n)$. Если некоторые координаты в векторе y стали отрицательными, то заменим их положительными дополнениями. Вектор y является решением ЛОДУ и, следовательно, представляется в виде неотрицательной линейной комбинации векторов из B, т. е. $y = \sum\limits_{i=1}^{m} b_i e_i,\ e_i \in B,\ i = 1, 2, \ldots, m$. Но тогда $u = x^1 + \sum\limits_{i=1}^{m} b_i e_i$. ■

Пример 9. Найти в поле F_{13} общее решение ЛНДУ $2x + 3y + 5z + 6u + 4v = 7$.

Решение. Выбираем первый ненулевой коэффициент $a_1 = 2$ и строим вектор $(7, 0, 0, 0, 0, 2)$. Решаем сравнение $2s \equiv 1 \ (mod\ 13)$. Этим решением будет очевидно, $s = 7$. Тогда вектор $x^1 = 7(7, 0, 0, 0, 0) = (10, 0, 0, 0, 0)$ будет искомым частным решением ЛНДУ.

Найдем базис множества решений ЛОДУ $2x+3y+5z+6u+4v = 0$. С этой целью заменим, например, коэффициент 3 его дополнением -10 и построим базис множества решений ЛОДУ $2x - 10y + 5z + 6u + 4v = 0$. Этими решениями будут векторы:

$$e_1 = (5, 1, 0, 0, 0),\ e_2 = (0, 1, 2, 0, 0),\ e_3 = (0, 3, 0, 5, 0),\ e_4 = (0, 2, 0, 0, 5).$$

Следовательно, общее решение данного ЛНДУ имеет вид:

$$x = x^1 + b_1 e_1 + b_2 e_2 + b_3 e_3 + b_4 e_4.$$

Например, при $b_1 = b_2 = b_3 = b_4 = 1$ получаем $u = (2, 7, 2, 5, 5)$. ♠

TSS-метод решения СЛНДУ. Пусть дана СЛНДУ

$$S = \left\{ \begin{array}{rcccccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1n}x_n & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2n}x_n & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sn}x_n & = & b_s, \end{array} \right. \tag{19}$$

где $a_{ij}, b_i, x_i \in F_p,\ i = 1, \ldots, n,\ j = 1, \ldots, s,\ s < n$. Поскольку обе части уравнения можно умножать на число, а сами уравнения складывать и вычитать, то преобразуем S к виду (полагая, что $b_s \neq 0$)

$$S' = \left\{ \begin{array}{rcccccccccc} L'_1(x) & = & a'_{11}x_1 & + & \ldots & + & a'_{1n}x_n & = & 0, & \\ L'_2(x) & = & a'_{21}x_1 & + & \ldots & + & a'_{2n}x_n & = & 0, & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L'_{s-1}(x) & = & a'_{s-11}x_1 & + & \ldots & + & a'_{s-1n}x_n & = & 0, & \\ L_s(x) & = & a_{s1}x_1 & + & \ldots & + & a_{sn}x_n & = & b_s, & \end{array} \right. \tag{20}$$

Пусть $B' = \{e'_1, e'_2, \ldots, e'_m\}$ – базис множества решений СЛОДУ, состоящей из первых $q-1$ уравнений системы S', а $B = \{e_1, e_2, \ldots, e_k\}$ – базис множества решений СЛОДУ, которая соответствует S'. Построим уравнение $L_s(e'_1)y_1 + \ldots + L_s(e'_m)y_m = b_s$. Если все $L_s(e_i) = 0$, то данное уравнение не имеет решений, а вместе с ним не имеет решений и исходная СЛНДУ (в этом случае $L_s(x)$ линейно зависит от $L'_1(x), \ldots, L'_{s-1}(x)$). Если хотя бы одно $L_s(e_i) \neq 0$, то решение всегда существует и пусть $y = (d_1, \ldots, d_m)$ – решение этого ЛОДУ. Тогда вектор $x^1 = d_1e_1 + \ldots + d_me_m$ является частным решением уравнения $L_s(x) = b_s$. Следовательно, общее решение СЛОДУ S', а вместе с ней и системы S, представляется в виде

$$u = x^1 + \sum_{i=1}^{k} b_i e_i,$$

где $e_i \in B, i = 1, 2, \ldots, k$.

Отсюда получаем оценку временной сложности алгоритма построения общего решения СЛНДУ над F_p. Действительно, к оценке, которая приведена в теореме 9, добавляется оценка вычисления частного решения ЛОДУ. Эта оценка включает сложность решения сравнения $ax \equiv b \ (mod\ p)$ и построение вектора, являющегося частным решением. Используя, например, способ поиска решения сравнения с помощью непрерывных дробей, получаем, что число элементарных операций, которые необходимо выполнить, не превосходит числа t – максимального числа двоичных разрядов в модуле p. А построение вектора требует не более n операций. Тогда общая оценка сложности всего алгоритма выражается оценкой, приведенной в теореме 9.

Таким образом, имеет место

Теорема 10. *СЛНДУ S, все уравнения которой линейно независимы и размерность которой $s \times n$, при $s < n$, всегда совместна над полем F_p и ее общее решение имеет вид $u = x^1 + \sum_{i=1}^{k} b_i e_i$, где x^1 – некоторое частное решение S, а e_i – базисные векторы множества решений СЛОДУ, которая соответствует данной СЛНДУ S.*

Сложность построения общего решения СЛНДУ пропорциональна величине sl^3, где s – количество уравнений в системе, $l = \max(t, n)$, t – количество двоичных разрядов числа p, а n – количество неизвестных в СЛНДУ.

Пример 10 (задача о математическом сейфе). Эта задача возникает в теории компьютерных игр [2]. Математическая постановка задачи о математическом сейфе состоит в следующем:

- дана матрица, состоящая из нулей и единиц;

- один ход игры состоит в выборе какого-нибудь элемента матрицы, после чего все нули в строке и столбце, к которым принадлежит этот элемент, превращаются в единицы, а все единицы – в нули.

Игра заканчивается в том случае, когда после некоторой последовательности ходов все элементы матрицы станут нулевыми (или единичными).

Реальная интерпретация этой задачи имеет такой вид. На дверях сейфа расположены в виде матрицы одинакового типа замки. Известны состояния каждого замка – он либо открытый либо закрытый. Если вставить ключ в один из замков и сделать один поворот ключа, то такой же поворот будет сделан во всех замках того же столбика и строки. Для того чтобы открыть сейф необходимо найти такую последовательность ходов переключения замков, чтобы поворотами в них ключа открыть сейф и это произойдет тогда, когда все замки будут открыты.

Данная задача имеет такую математическую постановку. Пусть задана матрица $B = ||b_{ij}||$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, которая состоит из нулей и единиц, а $X = ||x_{ij}||$ – решение задачи, причем $x_{ij} = 1$, если элемент b_{ij} принимает участие в последовательности ходов, и $x_{ij} = 0$ в противоположном случае. Тогда условие того, что элемент b_{ij} преобразуется матрицей X в нуль, выражается условием

$$\sum_{k=1}^{n} x_{ik} + \sum_{k=1, k\neq i}^{n} x_{kj} \equiv b_{ij} \ (mod\ 2). \tag{21}$$

Обозначим $x = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}, \ldots, x_{m1}, \ldots, x_{mn})$ вектор-столбик, полученный из матрицы X последовательной записью ее строк. Аналогично из матрицы B получаем вектор b. Пусть I_n – квадратная матрица $n \times n$, состоящая из единиц, а E_n – единичная матрица той же размерности. Тогда условие (21) запишется в виде СЛНДУ

$$A \cdot x \equiv b \ (mod\ 2),$$

где матрица A имеет размер $mn \times mn$ и состоит из m^2 клеток:

$$A = \begin{pmatrix} I_n & E_n & E_n & \dots & E_n \\ E_n & I_n & E_n & \dots & E_n \\ \dots & \dots & \dots & \dots & \dots \\ E_n & E_n & E_n & \dots & I_n \end{pmatrix},$$

в которой I_n – квадратная матрица размерности $n \times n$, состоящая из единиц, а E_n – единичная матрица той же размерности.

Рассмотрим конкретный пример, иллюстрирующий работу алгоритма в процессе решения задачи о математическом сейфе

Пусть имеем сейф в поле F_2 с матрицей

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Тогда матрица A принимает вид:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$

а соответствующая СЛНДУ будет такой:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{21} = 1, \\ x_{11} + x_{12} + x_{13} + x_{22} = 1, \\ x_{11} + x_{12} + x_{13} + x_{23} = 0, \\ x_{11} + x_{21} + x_{22} + x_{23} = 0, \\ x_{12} + x_{21} + x_{22} + x_{23} = 1, \\ x_{13} + x_{21} + x_{22} + x_{23} = 1. \end{cases}$$

Применяя TSS-алгоритм для решения этой СЛНДУ, получаем единственное решение $(1,0,0,0,0,1)$, т. е. $x_{11} = 1, x_{12} = 0, x_{13} = 0, x_{21} = 0, x_{22} = 0, x_{23} = 1$. Этому решению соответствуют такие преобразования матрицы B:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \spadesuit$$

Для иллюстрации того, как усложняется задача о математическом сейфе, рассмотрим ее в поле F_3 с матрицей

$$B = \begin{pmatrix} 2 & 0 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{pmatrix}.$$

Тогда матрица системы уравнений $Ax - b = 0$ имеет вид:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & -2 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & -2 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & -2 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & -2 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & -2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & -1 \end{pmatrix},$$

Применяя TSS-алгоритм для решения этой СЛНДУ, находим решение СЛНДУ $(0,2,2,0,0,2,0,0)$, т. е. $x_{11} = 0, x_{12} = 2, x_{13} = 2, x_{14} = 0, x_{21} = 0, x_{22} = 2, x_{23} = 0, x_{24} = 0$. Этому решению соответствуют такие преобразования матрицы B:

$$\begin{pmatrix} 2 & 0 & 2 & 2 \\ 1 & 2 & 2 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \spadesuit$$

6.1.2. Язык линейных уравнений над примарными кольцами

Частный случай ЛОДУ. Пусть дано ЛОДУ

$$L(x) = a_1x_1 + \ldots + a_ix_i + \ldots + a_nx_n = 0, \tag{22}$$

где $a_i, x_i \in Z_m,\ i = 1, \ldots, n$, которое удовлетворяет следующему условию.

Условие 1. Среди коэффициентов ЛОДУ существует коэффициент, который взаимно прост с модулем m.

Заметим, что для этого ЛОДУ тоже справедлива лемма 3. Допустим, что в данном ЛОДУ коэффициентом, который удовлетворяет условию 1, является первый ненулевой коэффициент a_k, $k \in [1, n]$. Рассмотрим функцию $L(x) = a_1x_1 + a_2x_2 + \ldots + a_nx_n$ ЛОДУ (22). Заменим в ней первый ненулевой коэффициент a_k, который взаимно прост с модулем, его отрицательным дополнением $-b_k$ и построим множество векторов

$$B = \{(0, \ldots, a_j, 0, \ldots, 0, b_k, 0, \ldots, 0)\} \cup M_0,$$

где $M_0 = \{e_r : L(e_r) = 0\}$, $a_j \neq 0$, а b_k является j-й координатой в векторах из B. Причем, если для некоторого a_j НОД$(a_j, b_k) \neq 1$, то сократим координаты такого вектора на этот общий делитель. Таким образом, можно считать, что все векторы в множестве B таковы, что a_j и b_k взаимно просты. Построенное таким образом множество также будем называть TSS. Очевидно, что векторы из множества B являются решениями ЛОДУ (22).

Лемма 7. *Если $d = (0, \ldots, 0, d_i, 0, \ldots, 0, d_j, 0, \ldots, 0)$ – решение ЛОДУ (22), то оно либо является элементом B, либо представляется в виде неотрицательной линейной комбинации векторов из B.*

Доказательство. Если $d \in B$, то доказывать нечего. Если $d = (0, \ldots, 0, d_i, 0, \ldots$ $\ldots, 0, d_j, 0, \ldots, 0) \notin B$, то возможны два случая.

Случай 1. В множестве B существуют векторы $s_1 = (0, \ldots, 0, a_i', 0, \ldots, 0, a_k', 0, \ldots, 0)$ и $s_2 = (0, \ldots, 0, a_j', 0, \ldots, 0, a_k', 0, \ldots, 0)$, в которых a_i' и a_j' являются k-ми координатами, а a_k' в s_1 и s_2 является соответственно i-й и j-й координатами. Поскольку s_1 и s_2 – решения ЛОДУ (22), то

$$s = us_1 + vs_2 = (0, \ldots, 0, a_iu + a_jv, 0, \ldots, 0, d_i, 0, \ldots, 0, d_j, 0, \ldots, 0)$$

тоже является решением этого ЛОДУ, где u, v – решения сравнений $a_k'x \equiv d_i\ (mod\ m)$ и $a_k'y \equiv d_j\ (mod\ m)$ соответственно. В кольце Z_m решения u и v единственны в силу взаимной простоты a_k' и m. Покажем, что $a_iu + a_jv = 0$. Для этого подставим вектор s в ЛОДУ (22):

$$L(s) = a_k'(a_iu + a_jv) + a_id_i + a_jd_j = a_k'(a_iu + a_jv) + 0 = a_k'(a_iu + a_jv) = 0.$$

Поскольку $a_k' \neq 0$ и не является делителем нуля, то $a_iu + a_jv = 0$, что и требовалось показать.

Случай 2. В множестве B существует вектор $s_1 = (0, \ldots, 0, a_i', 0, \ldots, 0, a_k', 0, \ldots, 0)$. Рассмотрим вектор

$$s = ys_1 = (0, \ldots, 0, ya_i', 0, \ldots, 0, d_j, 0, \ldots, 0),$$

где y – единственное решение сравнения $a_k'y \equiv d_j\ (mod\ m)$. Поскольку x и s – решения ЛОДУ (22), то $d - s$ тоже будет решением этого ЛОДУ, т. е. $d - s = (0, \ldots, 0, d_i - a_i'y, 0, \ldots, 0)$ и $L(d - s) = a_k'(d_i - a_i'y) = 0$. Так как $a_k' \neq 0$ и не является делителем нуля, то $d_i = a_i'y$. ■

Теорема 11. *Множество B решений ЛОДУ (22), построенное комбинированием дополнения первого ненулевого коэффициента, удовлетворяющего условию 1 и взятого с отрицательным знаком, с остальными ненулевыми коэффициентами и пополненное векторами канонического базиса, которые соответствуют нулевым коэффициентам ЛОДУ (22), является базисом множества всех решений этого ЛОДУ.*

Сложность алгоритма пропорциональна величине l^3, где $l = \max(t, n)$, $t = \log m$ – количество двоичных разрядов числа m, а n – количество неизвестных в ЛОДУ.

Доказательство проводится индукцией по числу k ненулевых координат в решении ЛОДУ. Пусть $x = (x_1, x_2, \ldots, x_n)$ – произвольное решение ЛОДУ (22).

Базис индукции. Если $k = 1$, то x должен совпадать с одним из векторов канонического базиса (который по построению является элементом B) или совпадать с одним из векторов, у которого ненулевая координата является делителем нуля в кольце Z_m. Первый случай очевиден. Во втором случае вектор имеет вид $(0, \ldots, 0, c_j, 0, \ldots, 0)$. Выберем решение из B, в котором j-я координата a'_k не равна нулю (такое решение должно существовать по построению множества B). Пусть это будет вектор $s_1 = (0, \ldots, 0, a'_i, 0, \ldots, 0, a'_k, 0, \ldots, 0)$. Рассмотрим вектор $z = ys_1$, где y – решение сравнения $a'_k y \equiv c_j \ (mod\ m)$ (такое решение существует и единственно в силу взаимной простоты a'_k и m). Но тогда вектор $z - x$ будет иметь единственную ненулевую координату ya'_i и при подстановке его в $L(x)$ получаем $ya'_i a_k n \equiv 0 \ (mod\ m)$, т. е. число $c_j a'_i = 0 \ (mod\ m)$. Следовательно, $x = z$.

При $k = 2$ справедливость теоремы вытекает из леммы 7.

Шаг индукции. Допустим, что теорема верна для всех $1 \le k < n$ и вектор-решение x имеет n ненулевых координат. Рассмотрим ненулевые координаты x_i, x_j вектора x. Возможны такие случаи.

Случай 1. В B существует вектор канонического базиса s с i-й координатой равной единице. Тогда вектор $y = x - x_i s$ будет иметь $n-1$ ненулевых координат. По предположению индукции для этого вектора существует представление $x - x_i s = d_1 e_1 + \ldots + d_p e_p$, где $e_i \in B$, $i = 1, \ldots, p$. Но тогда $x = x_i s + d_1 e_1 + \ldots + d_p e_p$.

Случай 2. В B существует вектор канонического базиса s с единственной ненулевой j-й координатой равной a, т. е. a – делитель нуля в Z_m. Выберем решение из B, в котором j-я координата a'_k не равна нулю (такое решение должно существовать по построению множества B). Пусть это будет вектор $s_1 = (0, \ldots, 0, a'_i, 0, \ldots, 0, a'_k, 0, \ldots, 0)$. Рассмотрим вектор $z = ds_1$, где d – решение сравнения $a'_k y \equiv c_j \ (mod\ m)$ (такое решение существует и единственно в силу взаимной простоты a'_k и m). Но тогда вектор $x - z$ будет иметь на одну ненулевую координату меньше, чем x. По предположению индукции вектор $x - z$ имеет представление (отрицательные координаты заменяются своими положительными дополнениями)

$$x - z = d_1 e_1 + \ldots + d_p e_p \text{ или } x = ds_1 + d_1 e_1 + \ldots + d_p e_p.$$

Случай 3. В B существует вектор $s = (0, \ldots, 0, a'_j, 0, \ldots, 0, a'_k, 0, \ldots, 0)$ с k-й и j-й ненулевыми координатами. Рассмотрим вектор

$$x - us = (x_1, \ldots, x_{k-1}, x_k - ua'_j, x_{k+1}, \ldots, 0, \ldots, x_n),$$

где u – решение сравнения $a'_k u \equiv x_j \ (mod\ m)$. Построенный вектор имеет $n - 1$ ненулевых координат, и по предположению индукции получаем, что

$$x - us = d_1 e_1 + \ldots + d_p e_p \text{ или } x = us + d_1 e_1 + \ldots + d_p e_p.$$

Случай 4. В B существуют векторы $s_1 = (0, \ldots, 0, a'_i, 0, \ldots, 0, a'_k, 0, \ldots, 0)$ и $s_2 = (0, \ldots, 0, a'_j, 0, \ldots, 0, a'_k, 0, \ldots, 0)$ с i-й, j-й и k-й ненулевыми координатами. Построим вектор

$$s = a'_j s_1 - a'_i s_2 = (0, \ldots, 0, a'_k a'_j, 0, \ldots, 0, -a'_k a'_i, 0, \ldots, 0),$$

у которого после замены $-a'_k a'_i$ его положительным дополнением ненулевыми координатами будут координаты с номерами i и j. Очевидно, что вектор s является решением ЛОДУ (22), и тогда согласно лемме 7 он выражается через векторы из множества B. Пусть $s' = (0, \ldots, 0, b_j, 0, \ldots, 0, b_i, 0, \ldots, 0)$ и $s' = d_1 e_1 + \ldots + d_r e_r$ это представление. Далее доказательство сводится к случаю 3, рассмотренном выше.

При получении оценки сложности элементарными считаются операции сложения и вычитания (арифметическая сложность выполнения которых пропорциональна t). При построении множества TSS для ЛОДУ вычисляется $n-1$ вектор и не больше, чем $n-1$ раз вычисляется НОД двух чисел в кольце Z_m. Построение векторов требует, очевидно, не более чем $n \cdot (n-1)$ шагов, а сложность вычисления НОД с помощью алгоритма Эвклида пропорциональна t^2, где t – разрядность чисел в двоичном представлении, для которых вычисляется НОД [7]. Суммируя все эти величины, получаем l^3, где $l = \max(t, n)$. ■

Следствие 3. *Если модуль m является простым числом, то множество B решений ЛОДУ (22) является базисом множества всех решений этого ЛОДУ.*

Сложность алгоритма пропорциональна величине l^3, где $l = \max(t, n)$, $t = \log m$ – число двоичных разрядов простого числа m, а n – количество неизвестных в ЛОДУ [7].

Действительно, если модуль m – простое число, то условие 1 выполняется автоматически.

Пример 11. Построить базис множества всех решений ЛОДУ $2x_1 + 5x_2 + 7x_3 + 3x_4 + 6x_5 = 0$ в кольце Z_{12}.

Решение. Выбираем ненулевой коэффициент 7, который взаимно прост с модулем 12, заменяем его дополнением -5 и получаем ЛОДУ вида $2x_1 + 5x_2 - 5x_3 + 3x_4 + 6x_5 = 0$. Применяя TSS-метод, получаем такие базисные решения:

$$e_1 = (5, 0, 2, 0, 0),\ e_2 = (0, 1, 1, 0, 0),\ e_3 = (0, 0, 3, 5, 0),\ e_4 = (0, 0, 6, 0, 5).\ \spadesuit$$

Случай ЛНДУ. Пусть дано ЛНДУ

$$a_1 x_1 + \ldots + a_k x_k + \ldots + a_n x_n = b, \tag{23}$$

у которого коэффициент a_k взаимно прост с модулем m. Найдем решение сравнения

$$a_k y \equiv b\ (mod\ m),$$

которое при данных условиях будет единственным. Пусть этим числом будет c, т. е. вектор $x^1 = (0, \ldots, 0, c, 0, \ldots, 0)$ будет решением (23). Применяя TSS-метод к ЛОДУ, которое соответствует (23), находим базис B множества его решений.

Пусть $x^1 = (c_1, c_2, \ldots, c_n)$ – некоторое частное решение (23), найденное описанным выше способом, а $B = \{e_1, e_2, \ldots, e_m\}$ – базис множества решений ЛОДУ

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = 0. \tag{24}$$

Теорема 12. *Произвольное решение ЛНДУ (23) представляется в виде $u = x^1 + \sum_{i=1}^{m} b_i e_i$, где x^1 – частное решение ЛНДУ (23), а $e_1, \ldots, e_m$ – базисные векторы множества решений ЛОДУ (24), которое соответствует ЛНДУ (23).*

Доказательство. Пусть $u = (u_1, u_2, \ldots, u_n)$ – произвольное решение ЛНДУ (23), а $B = \{e_1, e_2, \ldots, e_m\}$ – базис множества решений ЛОДУ, которое соответствует (23). Рассмотрим вектор $y = u - x^1 = (d_1, d_2, \ldots, d_n)$. Если некоторые координаты в векторе y стали отрицательными, то заменим их положительными дополнениями. Вектор y является решением ЛОДУ и, следовательно, представляется в виде неотрицательной линейной комбинации векторов из B, т. е. $y = \sum\limits_{i=1}^{m} b_i e_i$, $e_i \in B$, $i = 1, 2, \ldots, m$. Но тогда $u = x^1 + \sum\limits_{i=1}^{m} b_i e_i$. ■

Пример 12. Найти в кольце F_{12} общее решение ЛНДУ $2x + 3y + 5z + 6u + 4v = 7$.

Решение. Выбираем ненулевой коэффициент $a_3 = 5$, который взаимно прост с 12. Решаем сравнение $5s \equiv 7\ (mod\ 12)$. Решением этого сравнения есть $s = 11$. Тогда вектор $x^1 = (0, 0, 11, 0, 0)$ является искомым частным решением ЛНДУ.

Найдем базис множества решений ЛОДУ $2x + 3y + 5z + 6u + 4v = 0$. С этой целью заменим коэффициент 5 его дополнением -7 и построим базис множества решений ЛОДУ $2x + 3y - 7z + 6u + 4v = 0$. Этими решениями будут векторы

$$e_1 = (7, 0, 2, 0, 0),\ e_2 = (0, 7, 3, 0, 0),\ e_3 = (0, 0, 6, 7, 0),\ e_4 = (0, 0, 4, 0, 7).$$

Следовательно, общее решение данного ЛНДУ будет иметь вид

$$x = x^1 + b_1 e_1 + b_2 e_2 + b_3 e_3 + b_4 e_4.$$

Например, при $b_1 = 2, b_2 = 7, b_3 = b_4 = 0$ получаем $u = (2, 1, 0, 0, 0)$. ♠

Общий случай ЛОДУ над примарными кольцами. Рассмотрим ЛОДУ над примарным кольцом Z_m

$$L(x) = a_1 x_1 + \ldots + a_i x_i + \ldots + a_n x_n = 0, \tag{25}$$

где $a_i, x_i \in Z_m$, $m = p^k, k > 1, k \in \mathcal{N}$, $i = 1, \ldots, n$. Пусть НОД$(a_1, a_2, \ldots, a_n, m) = p^u$, тогда, сокращая (25) на p^u, получаем ЛОДУ

$$b_1 x_1 + b_2 x_2 + \ldots + b_n x_n = 0 \tag{26}$$

над примарным кольцом $Z_{m'}$, где $m' = p^v, v = k - u$. Полученное уравнение обладает тем свойством, что любое решение ЛОДУ (25) будет решением ЛОДУ (26). Обратное утверждение не имеет места. Действительно, пусть b_1 в (26) взаимно прост с модулем m'. Тогда строим TSS этого ЛОДУ, которое в силу теоремы 11 является базисом его множества решений:

$$s_1 = (b_2, c, 0, 0, 0, \ldots, 0), s_2 = (b_3, 0, c, 0, , 0 \ldots, 0),$$
$$s_3 = (b_4, 0, 0, c, 0, \ldots, 0), \ldots, s_{n-1} = (b_n, 0, 0, 0, \ldots, 0, c),$$

где $c = b_1 - p^v$ – дополнение коэффициента b_1. Поскольку кольцо Z_m с делителями нуля, то очевидным решением (25) будет вектор $s_n = (p^v, 0, 0, \ldots, 0)$, который не выражается неотрицательной линейной комбинацией векторов из TSS, так как $c \cdot x \equiv 0\ (mod\ p^v)$ тогда и только тогда, когда $x = p^v$ в силу взаимной простоты c и p^v.

Имеет место

Теорема 13. *TSS уравнения (26), среди коэффициентов которого есть коэффициент удовлетворяющий условию 1, дополненное вектором $s_n = (p^v, 0, 0, \ldots, 0)$, является базисом множества решений ЛОДУ (25).*

Доказательство. Пусть $x = (d_1, d_2, \ldots, d_n)$ – произвольное решение ЛОДУ (25). Построим вектор $y = c_1 s_1 + c_2 s_2 + \ldots + c_{n-1} s_{n-1} = (c_1 b_2 + \ldots + c_{n-1} b_n, d_2, \ldots, d_n)$, т. е. $d_i \equiv c_i \cdot c\ (mod\ m')$, $i = 2, \ldots, n$, $c = b_1 - p^v$, и рассмотрим вектор

$$y - x - c_n s_n = (c_1 b_2 + \ldots + c_{n-1} b_n - d_1', 0, \ldots, 0) = (c_1 b_2 + \ldots, c_{n-1} b_n + d_1'', 0, \ldots, 0),$$

где d_1'' – дополнение $d_1' c_n \equiv d_1''\ (mod\ m')$. Полученный вектор является решением (25), а следовательно, является решением (26). Представим d_1'' в виде $b_1 d$, где $b_1 d \equiv d_1''\ (mod\ p^v)$ (такое представление единственно в силу взаимной простоты b_1 и p^v). Тогда $y - x - c_n s_n = (c_1 b_2 + \ldots + c_{n-1} b_n + b_1 d'', 0, \ldots, 0)$, и после подстановки $y - x$ в ЛОДУ (26) получаем

$$b_1(b_1 d'' + b_2 c_1 + \ldots, b_n c_{n-1}) \equiv 0\ (mod\ p^v).$$

Следовательно, возможны два случая:

а) $b_1 d'' + b_2 c_1 + \ldots, b_n c_{n-1} = g p^l \equiv 0\ (mod\ p^k)$;

б) $b_1 d'' + b_2 c_1 + \ldots, b_n c_{n-1} = g p^l \not\equiv 0\ (mod\ p^k)$.

В случае а) доказывать нечего. В случае б) для окончательного представления вектора x необходимо из вектора $x - y$ вычесть вектор $g s_n$: $x - y - (c_n + g) s_n = 0$ и $x = y + (c_n + g) s_n$. ■

Если для ЛОДУ не выполняется условие 1, то этого всегда можно добиться. Это следует из такого простого утверждения.

Лемма 8. *ЛОДУ (26) удовлетворяет условию 1, т. е. в этом уравнении существует по крайней мере один коэффициент, который взаимно прост с модулем p^k.*

Доказательство. Рассмотрим произвольный ненулевой коэффициент a_i ЛОДУ (25). Если a_i и m взаимно простые, то доказательство не требуется. Если таковых коэффициентов нет, т. е. НОД$(a_1, a_2, \ldots, a_n, m) = p^u,\ u < k$, то сокращая на p^u коэффициенты и модуль, получаем уравнение, эквивалентное исходному, для которого выполняется условие 1. ■

Отсюда вытекает, что ЛОДУ (26) удовлетворяет условию 1. Тогда базис множества решений преобразованного ЛОДУ строится TSS-алгоритмом.

TSS-метод решения СЛОДУ. Из вышеприведенных теорем следует такая процедура построения базиса множества решений СЛОДУ (19) над примарным кольцом Z_m. Она сводится к решению ЛОДУ в примарном кольце Z_m с помощью TSS-метода. Проиллюстрируем это на примерах.

Пример 13. а) Построить базис множества всех решений в кольце Z_8 для СЛОДУ

$$S = \begin{cases} 2x + 3y + 8z + 6u + 4v = 0, \\ 4x + 6y + 2z + 3u + 2v = 0, \\ 2x + 3y + 2z + 2u + 8v = 0. \end{cases}$$

Решение. В результате приведения коэффициентов системы получаем СЛОДУ:

$$S_1 = \begin{cases} L_1 = 2x + 3y + 0z + 6u + 4v = 0, \\ L_2 = 4x + 6y + 2z + 3u + 2v = 0, \\ L_3 = 2x + 3y + 2z + 2u + 0v = 0. \end{cases}$$

Строим базис B СЛОДУ S_1, выбирая коэффициент $a_{12} = 3$ и его дополнение $b = -5$:

$$B_1 = \{e_1 = (5, 2, 0, 0, 0), e_2 = (0, 0, 1, 0, 0), e_3 = (0, 6, 0, 5, 0), e_4 = (0, 4, 0, 0, 5)\}.$$

Значения L_2 на векторах из B_1: 0, 2, 3, 2. Получаем уравнение $0d_1 + 2d_2 + 3d_3 + 2d_2 = 0\ (mod\ 8)$, которое имеет базисные решения (1,0,0), (0,5,2,0), (0,0,2,5). Этим решениям соответствуют базисные векторы

$$B_2 = \{s_1 = 1 \cdot e_1 = (5, 2, 0, 0, 0), s_2 = 5e_2 + 2e_3 = (0, 4, 5, 2, 0), s_3 = 2e_3 + 5e_4 = (0, 0, 0, 2, 1)\}.$$

Значения L_2 на векторах из B_2: 0, 2, 4. Получаем уравнение $0d_1 + 2d_2 + 4d_3 = 0\ (mod\ 8)$, которое имеет решения (1,0,0), (0,2,3) и (0,4,0) (поскольку НОД(2,4) = 2). Этим решениям соответствуют базисные векторы исходной системы S

$$B = \{v_1 = 1 \cdot s_1 = (5,2,0,0,0), v_2 = 4s_2 = (0,0,4,0,0), v_3 = 2s_2 + 3s_3 = (0,0,2,2,3)\}.$$

б) Построить базис множества всех решений в кольце Z_{24} для СЛОДУ

$$S = \begin{cases} 2x + 3y + 8z + 6u + 4v = 0, \\ 4x + 6y + 2z + 3u + 2v = 0, \\ 2x + 3y + 2z + 2u + 8v = 0. \end{cases}$$

Решение. В результате разложения модуля $m = 24 = 3 \cdot 8$ получаем две СЛОДУ:

$$S_1 = \begin{cases} L_{11} = 2x + 0y + 0z + 0u + 1v = 0, \\ L_{12} = 1x + 0y + 2z + 0u + 2v = 0, \\ L_{13} = 2x + 0y + 2z + 2u + 2v = 0 \end{cases}$$

и

$$S_2 = \begin{cases} L_{21} = 2x + 3y + 0z + 6u + 4v = 0, \\ L_{22} = 4x + 6y + 2z + 3u + 2v = 0, \\ L_{23} = 2x + 3y + 2z + 2u + 0v = 0. \end{cases}$$

Решения СЛОДУ S_1 находим в поле F_3, а СЛОДУ S_2 – в примарном кольце Z_8.

Строим базис B_1 СЛОДУ S_1:

$$B_{11} = \{(2,0,1,0,0), (0,1,0,0,0), (0,0,0,1,0), (1,0,0,0,1)\}.$$

Значения L_{12} на векторах из B_{11}: 1,0,0,0. Тогда

$$B_{12} = \{(0,1,0,0,0), (0,0,0,1,0), (1,0,0,0,1)\}.$$

Значения L_{13} на векторах из B_{12}: 0,2,1. Тогда

$$B_1 = B_{13} = \{(0,1,0,0,0), (1,0,0,1,1)\}.$$

Базис B_2 СЛОДУ S_2 был построен выше в пункте а):

$$B_2 = \{(5,2,0,0,0), (0,0,2,2,3)\} \cup \{(0,0,4,0,0)\}.$$

Таким образом окончательно получаем базис множества решений исходной СЛОДУ

$$B = 8 \cdot B_1 \cup 3 \cdot B_2 = \{(0,8,0,0,0), (8,0,0,8,8), (15,6,0,0,0), (0,0,6,6,9), (0,0,12,0,0)\}. \spadesuit$$

В общем случае, если модуль m имеет разложение, содержащее больше двух сомножителей, т. е. $m = p_1^{k_1} p_2^{k_2} \dots p_r^{k_r}$, то получаем r подсистем. Принимая во внимание, что арифметическая сложность выполнения операций сложения и вычитания двоичных чисел в кольце Z_m пропорциональна t (t – максимальная разрядность рассматриваемых двоичных чисел), операций умножения и деления, как и вычисления НОД двух чисел, меньших m пропорциональна t^2, то арифметическая сложность построения базиса множества решений СЛОДУ имеет вид:

– l^3 – решение одного ЛОДУ и решение одного промежуточного ЛОДУ;

– $n^2 l^3$ – вычисление значений и сокращение на НОД $L(x)$.

– $n^2 l^3$ – построение комбинаций векторов, составляющих базис множества решений ЛОДУ ($l = \max(n,t)$).

Таким образом, арифметическая сложность перехода от предыдущего к последующему ЛОДУ в одной подсистеме пропорциональна величине l^5, где $l = \max(n,t)$, $t = \log m$. Такая процедура повторяется r раз и в результате имеем $O(l^6)$, где $l = \max(n,t,r)$. Иными словами имеет место

Теорема 14. *Множество B, построенное TSS-методом, является базисом множества решений СЛОДУ (19). Арифметическая сложность построения B пропорциональна величине $O(l^6)$, где $l = \max(n,t,r)$.*

TSS-**метод решения СЛНДУ**. Построение базиса множества решений СЛНДУ сводится к поиску частного решения СЛНДУ и базиса множества решений приведенной СЛОДУ. Это построение выполняется путем перехода к расширенной СЛОДУ, у которой к исходной СЛОДУ добавляется столбик из свободных членов с дополнительной неизвестной. Построив базис множества решений такой СЛОДУ, выделяем базисные решения, у которых последняя координата (она соответствует дополнительной неизвестной) отлична от нуля. Если такой координаты нет, то исходная СЛНДУ несовместна. В противном случае составляется уравнение вида $c_1z_1 + \ldots + c_rz_r = 1\ (mod\ m)$, где $c_1, \ldots, c_r$ – значения ненулевых координат выделенных векторов. Если это уравнение не имеет решений, то исходная СЛНДУ несовместна, в противном случае по одному из его решений строим частное решение СЛНДУ, которое вместе с базисными решениями СЛОДУ, которая соответствует данной СЛНДУ, будут составлять базис множества всех решений исходной СЛНДУ.

Пример 14. Найти базис множества всех решений в кольце Z_{24} для СЛНДУ

$$S = \begin{cases} 2x + 3y + 8z + 6u = 20, \\ 4x + 6y + 2z + 3u = 22, \\ 2x + 3y + 2z + 2u = 16. \end{cases}$$

Решение. От этой СЛНДУ переходим к расширенной СЛОДУ

$$S = \begin{cases} 2x + 3y + 8z + 6u + 4v = 0, \\ 4x + 6y + 2z + 3u + 2v = 0, \\ 2x + 3y + 2z + 2u + 8v = 0, \end{cases}$$

у которой последний столбик соответствует свободным членам с дополнительной неизвестной v. Базис множества решений данной СЛОДУ был найден в предыдущем примере

$$B = 8 \cdot B_1 \cup 3 \cdot B_2 = \{(0,8,0,0,0), (8,0,0,8,8), (15,6,0,0,0), (0,0,6,6,9), (0,0,12,0,0)\}.$$

Выделяем вектор (8,0,0,8,8) и вектор (0,0,6,6,9) с ненулевыми последними координатами и строим уравнение $8x + 9y = 1\ (mod\ 24)$ по последним координатам выделенных векторов. Это уравнение имеет решение (-1,1) = (23,1), которому соответствует частное решение исходной СЛОДН $x^1 = (16,0,6,22)$. Тогда общее решение исходной СЛНДУ принимает вид:

$$x = x^1 + a(0,8,0,0,0) + b(15,6,0,0,0) + c(0,0,12,0),$$

где $a, b, c \in Z_{24}$ – произвольные постоянные. ♠

Можно предложить и такой способ поиска общего решения СЛНДУ, который представим в виде такой последовательности шагов:

1) $i = 1$;

2) Найти частное решение x^1 ЛНДУ $L_i(x) = b_i$. Если x^1 не существует, то (СТОП: решений нет), иначе на шаг 3);

3) Построить базис $B_i = \{e_{i1}, e_{i2}, \ldots, e_{iw}\}$ ЛОДУ, которое соответствует ЛНДУ $L_i(x) = b_i$;

4) Найти значения $c = L_{i+1}(x^1)$ и $c_1 = L_{i+1}(e_{i1}), \ldots, c_w = L_{i+1}(e_{iw})$, где $e_{ij} \in B_i$, $j = 1, 2, \ldots, w$;

5) Найти частное решение y^1 ЛНДУ

$$c_1y_1 + \ldots + c_wy_w = b_{i+1} - c. \tag{27}$$

Если y^1 не существует, то (СТОП: решений нет), иначе на шаг 6);

6) Построить базис B_i' ЛОДУ, которое соответствует (27);

7) Построить базис B_{i+1} для ЛНДУ $L_{i+1}(x) = b_{i+1}$ исходя из B'_i;
8) Если $i+1 < r$, то ($i = i+1$; на 4)), иначе (СТОП: печать B_{i+1}.
Правильность этой процедуры следует из доказанных выше теорем и лемм.

Пример 15. а) Найти в кольце Z_{12} общее решение СЛНДУ

$$S = \left\{ \begin{array}{l} 2x_1 + 3x_2 + 8x_3 + 6x_4 + 4x_5 = 8 \\ 4x_1 + 3x_2 + 6x_3 + 6x_4 + 8x_5 = 6 \end{array} \right.$$

Решение. Поскольку разложение 12 на простые множители имеет вид $m = 12 = 3 \cdot 4$, то построение общего решения первого ЛНДУ системы S сводится к нахождению его частного решения и построения в кольце Z_{12} базиса ЛОДУ вида

$$2x_1 + 3x_2 + 8x_3 + 6x_4 + 4x_5 = 0.$$

Очевидным решением ЛНДУ является вектор $x^1 = (4, 4, 0, 0, 0)$, а построение базиса множества решений ЛОДУ в кольце Z_{12} сводится к решению двух ЛОДУ соответственно в поле F_3 и в примарном кольце Z_4 вида

$$\begin{array}{l} 2x_1 + 0x_2 + 2x_3 + 0x_4 + 1x_5 = 0, \\ 2x_1 + 3x_2 + 0x_3 + 2x_4 + 0x_5 = 0. \end{array}$$

Базис множества решений первого уравнения составляют векторы

(2,0,1,0,0), (0,1,0,0,0), (0,0,0,1,0), (1,0,0,0,1),

а базис множества решений второго уравнения составляют векторы

(1,2,0,0,0), (0,2,0,1,0), (0,0,1,0,0), (0,0,0,0,1).

Значения второго уравнения на векторах из первого базиса равны: 0, 3, 2, 2, а значения первого уравнения на векторах из второго базиса равны: 2, 0, 2, 1. Следовательно, базис B_1 исходного ЛОДУ включает векторы

(2,0,1,0,0), (0,4,0,0,0), (0,0,0,2,0), (2,0,0,0,2), (3,6,0,0,0), (0,2,0,1,0), (0,0,3,0,0), (0,0,0,0,3).

Тогда общее решение исходного ЛНДУ имеет вид

$$x = x^1 + \sum_{i=1}^{8} a_i e_i,$$

где $e_i \in B_1,\ i = 1, \ldots, 8$.

Подставляя полученные векторы во второе уравнение исходной СЛНДУ, получаем такие значения $L_2(x^1) = 4$, а на остальных векторах соответственно 2,0,0,0,6,0,6,0. Строим уравнение (для упрощения опущены нулевые коэффициенты)

$$2y_1 + 6y_2 + 6y_3 = 6 - 4 = 2 \text{ или } y_1 + 3y_2 + 3y_3 = 1 \text{ в кольце } Z_6.$$

Решениями полученного уравнения будут векторы (3, 0, 5), (3, 5, 0) и частное решение $y^1 = (1, 0, 0)$. Этим векторам соответствуют такие редуцированные векторы:

$x^1 = (0, 0, 1, 0, 0), s_1 = (3, 2, 3, 0, 0), s_2 = (0, 0, 6, 0, 0),\ s_3 = (0, 2, 0, 1, 0), s_4 = (0, 0, 0, 0, 3), s_5 = (2, 0, 0, 0, 2)$.

Таким образом, общее решение исходной СЛНДУ имеет вид

$$x = x^1 + \sum_{i=1}^{5} a_i s_i.$$

б) Найти в кольце Z_{12} общее решение СЛНДУ

$$S = \left\{ \begin{array}{l} 2x_1 + 3x_2 + 8x_3 + 6x_4 + 4x_5 = 8, \\ 4x_1 + 3x_2 + 6x_3 + 6x_4 + 8x_5 = 5. \end{array} \right.$$

Решение. Общее решение первого ЛНДУ было найдено в предыдущем примере:

$$x = x^1 + \sum_{i=1}^{8} a_i e_i,$$

где $x^1 = (4, 4, 0, 0, 0)$, а

$$e_1 = (2,0,1,0,0),\ e_2 = (0,4,0,0,0),\ e_3 = (0,0,0,2,0),\ e_4 = (2,0,0,0,2),$$
$$e_5 = (3,6,0,0,0),\ e_6 = (0,2,0,1,0),\ e_7 = (0,0,3,0,0),\ e_8 = (0,0,0,0,3).$$

Подставляя эти векторы во второе уравнения СЛНДУ, получаем такие значения: $L_2(x^1) = 4$, а на остальных векторах – значения 2, 0, 0, 0, 6, 0, 6, 0. Строим уравнение (для упрощения опущены нулевые коэффициенты) $2y_1 + 6y_2 + 6y_3 = 5 - 4 = 1$. Полученное уравнение не имеет решений, поскольку наибольший общий делитель модуля и коэффициентов равен 2, а 2 не делит свободный член 1. Следовательно, исходная СЛНДУ решений не имеет. ♠

Отметим, что приведенные алгоритмы имеют полиномиальные оценки временной сложности при условии известного разложения модуля на простые множители. Проблема разложения натурального числа на простые множители (которая называется проблемой факторизации) является одной из наиболее важных проблем теории чисел. Имеется несколько алгоритмов ее решения: алгоритмы Полларда, Полларда-Штрассена, решета числового поля [15]. Наиболее эффективным алгоритмом в настоящее время является последний из упомянутых алгоритмов, потому что в отличие от первых двух алгоритмов он ищет большие делители заданного числа. Все эти алгоритмы имеют экспоненциальные оценки временной сложности, наилучшая из которых для заданного числа n имеет вид $O(2^{c\sqrt{\ln n \ln\ln n}})$.

7. Язык линейных уравнений над $\mathcal{N}$

Этот язык является примером NP-полного языка. Описание его свойств требует введения отношения порядка на множестве векторов $\mathcal{N}^q \setminus \{\bar{0}\}$. Это отношение определяется покоординатно: для произвольных $x = (x_1, x_2, \ldots, x_q), y = (y_1, y_2, \ldots, y_q) \in \mathcal{N}^q \setminus \{\bar{0}\}$

$$x \ll y \Leftrightarrow x_i \le y_i\ \forall i \in [1,q].$$

Это отношение является отношением частичного порядка и относительно этого порядка можно говорить о минимальных элементах в множестве $\mathcal{N}^q \setminus \{\bar{0}\}$. Известно, что множество B минимальных элементов множества решений M СЛОДУ относительно этого порядка составляет **базис** множества M и если $|M| > 1$, то базис B всегда существует, конечен и всякий элемент из M представим в виде неотрицательной линейной комбинации векторов из B [18, 19]. Как отмечалось выше, процесс решения СЛНДУ или системы линейных диофантовых неравенств (СЛДН) может быть сведен к решению СЛОДУ. Такое сведение увеличивает размерность СЛОДУ, что снижает эффективность вычислений, но имеются методы сведения, которые не увеличивают размерности пространства [48].

7.1. Критерий совместности СЛОДУ

Критерий совместности СЛОДУ, используемый здесь, и алгоритм его реализации подробно описаны в работах [5, 16], поэтому приведем лишь необходимые сведения.

Пусть дана СЛОДУ

$$S = \left\{ \begin{array}{rcccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & 0, \end{array} \right. \tag{28}$$

где $a_{ij} \in \mathcal{Z}, i = 1,2,\ldots,p,\ j = 1,2,\ldots,q$. Рассмотрим множество векторов канонического базиса $M_0' = \{e_1, \ldots, e_q\}$ и первое уравнение $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q = 0$ системы S. С помощью функции $L_1(x)$ разобъем элементы множества M_0' на такие три группы

$M_1^0 = \{e^0 | L_1(e^0) = 0\},\ M_1^+ = \{e^+ | L_1(e^+) > 0\}$ и $M_1^- = \{e^- | L_1(e^-) < 0\}$. Ясно, что если одно из множеств $M_1^0 \cup M_1^+$ или $M_1^0 \cup M_1^-$ пусто, то уравнение $L_1(x) = 0$ не имеет нетривиальных решений в множестве натуральных чисел. Допустим, что хотя бы два из множеств $M_1^0,\ M_1^+,\ M_1^-$ непусты, тогда рассмотрим множество

$$M_1' = M_1^0 \cup \{e_{ij} | e_{ij} = -L_1(e_i)e_j + L_1(e_j)e_i, e_j \in M_1^+, e_i \in M_1^-\}.$$

Используя функцию $L_2(x)$ разобъем элементы множества M_1' аналогично предыдущему тоже на три группы $M_2^0 = \{e^0 | L_2(e^0) = 0\},\ M_2^+ = \{e^+ | L_2(e^+) > 0\}$ и $M_2^- = \{e^- | L_2(e^-) < 0\}$. Допустим, что хотя бы два из этих множеств непусты, тогда построим множество

$$M_2' = M_2^0 \cup \{e_{ij} | e_{ij} = -L_2(e_i)e_j + L_2(e_j)e_i,\ e_j \in M_2^+,\ e_i \in M_2^-\}.$$

Предположим, что таким способом построено множество M_j' из множеств $M_j^0 = \{e_r^0 | L_j(e_r^0) = 0\},\ M_j^+ = \{e_i^+ | L_j(e_i^+) > 0\}$ и $M_j^- = \{e_s^- | L_j(e_s^-) < 0\}$ с помощью функции $L_j(x)$ и это множество непусто. Непосредственно из этих построений вытекает

Теорема 15. *Элементы множества M_j' являются решениями системы уравнений $L_1(x) = 0 \wedge L_2(x) = 0 \wedge \ldots \wedge\ L_j(x) = 0$.*

Множество M_j', построенное выше, будем называть TSS системы $S' = L_1(x) = 0 \wedge L_2(x) = 0 \wedge \ldots \wedge L_j(x) = 0$.

Пусть $M_j' = \{e_1', \ldots, e_k'\}$ – TSS системы S', а M_j – множество всех ее решений. Имеет место

Теорема 16. *Для всякого вектора $x \in M_j \setminus M_j'$ существует представление в виде неотрицательной линейной комбинации вида*

$$tx = b_1 e_1' + \ldots + b_l e_k', \tag{29}$$

где $t, b_i \in \mathcal{N},\ t \neq 0,\ e_i' \in M_j',\ i = 1, \ldots, k$ [5, 6].

Доказательство теоремы использует следующую лемму.

Лемма 9. *Любая неотрицательная линейная комбинация вида*

$$y = ce_i^+ + de_s^-$$

представляется как неотрицательная линейная комбинация вида $ly = ue_i^+ + ve_{is}^0$ или же как неотрицательная линейная комбинация вида $ly = ue_s^- + ve_{is}^0$, где l, u, v – натуральные числа, $e_s^- \in M_j^-,\ e_i^+ \in M_j^+,\ e_{is}^0 \in M_j'$.

Критерий проверки совместности СЛОДУ формулируется следующим образом.

Теорема 17. *Система $S = L_1(x) = 0 \wedge\ L_2(x) = 0 \wedge \ldots \wedge L_{p-1}(x) = 0 \wedge L_p(x) = 0$ совместна тогда и только тогда, когда $M_p' \neq \emptyset$.*

Заметим, что из теорем 16, 17 следует, что каждый вектор из TSS можно разделить на НОД его координат, если этот НОД отличен от единицы. Это позволяет уменьшить величину координат этих векторов и более эффективно проводить вычисления.

Легко заметить, что TSS зависит от порядка, в котором расположены уравнения системы. Исключение избыточных векторов из этого множества решений базируется на следующей теореме.

Теорема 18. *Пусть S – СЛОДУ вида (28) и M'_p – ее TSS, состоящее из k элементов. Тогда любой вектор x из M'_p такой, что $tx \gg e'_i \in M'_p \setminus \{x\},\ i = 1, 2, \ldots,\ k-1,\ t \in \mathcal{N}$ и $t \neq 0$, имеет представление вида*

$$mx = b_1 e'_1 + b_2 e'_2 + \ldots + b_{k-1} e'_{k-1},$$

где $m \in \mathcal{N},\ m \neq 0,\ b_i \in \mathcal{N},\ e_i \in M'_p,\ i = 1, 2, \ldots, k-1$.

Из приведенной теоремы вытекает следующая простая процедура чистки TSS: вектор x удаляется из TSS, если x больше или его произведение tx больше некоторого из оставшихся векторов TSS. В качестве множителя t можно взять, в частности, максимальную координату векторов текущего TSS.

7.2. Свойства TSS СЛОДУ

Допустим, что СЛОДУ S совместна и $M' = \{e'_1, e'_2, \ldots, e'_k\}$ ее TSS.

Теорема 19 (минимальность) . *Векторы из TSS являются минимальными решениями СЛОДУ S, т.е. являются ее базисными решениями.*

Доказательство. Покажем сначала, что никакой вектор e'_i из M' не представим в виде неотрицательной линейной комбинации остальных векторов из M'.

Допустим что в множестве M' существует вектор e'_i для которого имеет место разложение

$$te'_i = a_1 e'_1 + a_2 e'_2 + \ldots + a_k e'_k.$$

Тогда $te'_i \gg e'_1, \ldots, te'_i \gg e'_{i-1}, te'_i \gg e'_{i+1}, \ldots, te'_i \gg e'_k$. Но в силу теоремы 18 вектор e'_i не попадает в множество M'. Получаем противоречие со способом построения $TSS\ M'$.

Допустим теперь, что существует некоторое решение x данной СЛОДУ S такое, что $x \ll e'_i$, где $e'_i \in M'$. Тогда в силу теоремы 16 имеем такие представления для векторов x и $e'_i - x$:

$$t(e'_i - x) = a_1 e'_1 + a_2 e'_2 + \ldots + a_k e'_k,$$
$$t'x = b_1 e'_1 + b_2 e'_2 + \ldots + b_k e'_k,$$

где $t, t' \neq 0,\ a_i, b_i, t, t' \in \mathcal{N},\ i = 1, 2, \ldots, k.$. Домножив первое из равенств на t', а второе на t и подставив одно в другое, получаем

$$t'te'_i - t'tx = t'a_1 e'_1 + \ldots + t'a_k e'_k,$$

или

$$t'te'_i = (t'a_1 + tb_1)e'_1 + \ldots + (t'a_k + tb_k)e'_k.$$

Но полученное равенство противоречит тому, что $e'_i \in M'$, а это доказывает теорему. ■

Теорема 20. *Пусть $x = (x_1, x_2, \ldots, x_q)$ минимальное решение СЛОДУ S и $M' = \{e'_1 = (\alpha_{11}, \ldots, \alpha_{1q}), e'_2 = (\alpha_{21}, \ldots, \alpha_{2q}), \ldots, e'_k = (\alpha_{k1}, \ldots, \alpha_{kq})\}$ ее TSS. Тогда имеет место неравенство*

$$x' = \max_i\ x_i\ \leq\ k \cdot \max_{i,j}\ \alpha_{ij},$$

где α_{ij} *– координаты векторов* $e'_i \in M',\ i = 1, ..., k,\ j = 1, 2, ..., q.$

Доказательство. Если $x \in M'$, то утверждение теоремы очевидно имеет место. Допустим, что $x \in B \setminus M'$, где B – базис всего множества решений СЛОДУ S. Тогда в силу теорем 16, 18 вектор x имеет представление

$$tx = a_1e'_1 + a_2e'_2 + \ldots + a_ke'_k,$$

где $t > 1,\ a_i, t \in \mathcal{N},\ e'_i \in M',\ i = 1, \ldots, k$. Заметим, что для коэффициентов a_i $(i = 1, 2, \ldots, k)$ имеют место неравенства $a_i < t$. Действительно, если допустить что некоторое $a_i \geq t$, то из равенства

$$tx = a_1e'_1 + \ldots + a_ie'_i + \ldots + a_ke'_k$$

следует равенство

$$t(x - e'_i) = a_1e'_1 + \ldots + a'_ie'_i + \ldots + a_ke'_k,$$

где $a'_i = a_i - t$. Последнее равенство значит, что вектор $x - e'_i$ представляется в виде неотрицательной линейной комбинации векторов из M', но тогда $x \gg e'_i$, что противоречит минимальности вектора x. Записывая равенство

$$tx = a_1e'_1 + a_2e'_2 + \ldots + a_ke'_k,$$

в координатной форме, получаем такую систему уравнений

$$\left\{\begin{array}{lcccccccc} tx_1 & = & a_1\alpha_{11} & + & \ldots & + & a_k\alpha_{k1}, \\ tx_2 & = & a_1\alpha_{12} & + & \ldots & + & a_k\alpha_{k2}, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ tx_q & = & a_1\alpha_{1q} & + & \ldots & + & a_k\alpha_{kq}. \end{array}\right. \tag{30}$$

Из равенств системы (30) вытекает, что для любого $j = 1, 2, \ldots, q$ справедливы неравенства

$$tx_j \leq \alpha(a_1 + a_2 + \ldots + a_k) < \alpha \cdot k \cdot t$$

или после сокращения получаем $x_j < k \cdot \alpha$, где $\alpha = \max\limits_{i,j}\ \alpha_{ij},\ i = 1, \ldots, k,\ j = 1, 2, \ldots, q.$ ■

Пример 16. Система

$$S_1 = \left\{\begin{array}{lcccccccccccc} 5x_1 & + & 0x_2 & + & 3x_3 & + & 7x_4 & - & 4x_5 & + & 0x_6 & + & 0x_7 & = & 0, \\ 1x_1 & + & 2x_2 & + & x_3 & + & 1x_4 & + & 0x_5 & - & 4x_6 & + & 0x_7 & = & 0, \\ 0x_1 & + & 1x_2 & + & 2x_3 & + & 1x_4 & + & 0x_5 & + & 0x_6 & - & 4x_7 & = & 0 \end{array}\right.$$

имеет 10 базисных векторов-решений:

$$e_1 = (4, 0, 0, 0, 5, 1, 0),\ e_2 = (0, 4, 0, 0, 0, 2, 1),\ e_3 = (0, 0, 4, 0, 3, 1, 2),\ e_4 = (0, 0, 0, 4, 7, 1, 1),$$
$$e_5 = (3, 1, 0, 3, 9, 2, 1),\ e_6 = (1, 1, 2, 3, 8, 2, 2),\ e_7 = (2, 2, 0, 2, 6, 2, 1), e_8 = (0, 2, 2, 2, , 5, 2, 2),$$
$$e_9 = (1, 3, 0, 1, 3, 2, 1),\ e_{10} = (2, 0, 2, 0, 4, 1, 1).$$

TSS составляют вектора $e_1,\ e_2,\ e_3,\ e_4$. Максимальное значение координат этих векторов равно 7, максимальное значение координат базисных векторов равно 9 и $9 < 4 \cdot 7 = 28$. ♠

Пусть $TSS(M, S)$ обозначает TSS для заданной СЛОДУ S с начальным TSS M. Если M совпадает с каноническим базисом, то будем писать $TSS(\emptyset, S)$.

Теорема 21 (инкрементальность) . $TSS(\emptyset, S_1 \wedge S_2) = TSS(M_1', S_2) = TSS(M_2', S_1)$, *где* $M_1' = TSS(\emptyset, S_1)$, $M_2' = TSS(\emptyset, S_2)$.

Доказательство. Равенство $TSS(\emptyset, S_1 \wedge S_2) = TSS(M_1', S_2)$ следует из способа построения TSS.

Для доказательства второго равенства положим $M = TSS(M_1', S_2) = \{e_1, e_2, \ldots, e_k\}$ и $M' = TSS(M_2', S_1) = \{e_1,', e_2', \ldots, e_m'\}$. Пусть для определенности $e_1 \in M$, тогда в соответствии с теоремой 16, можем записать

$$te_1 = a_1e_1' + a_2e_2' + \ldots + a_me_m', \tag{31}$$

где $e_i' \in M', i = 1, 2, \ldots, m$. Поскольку e_i' являются решениями системы $S_1 \wedge S_2 = S_2 \wedge S_1$, то существуют $t_1, t_2, \ldots, t_m \in \mathcal{N}$ такие, что

$$\begin{gathered} t_1e_1' = b_{11}e_1 + b_{12}e_2 + \ldots + b_{1k}e_k, \\ t_1e_2' = b_{21}e_1 + b_{22}e_2 + \ldots + b_{2k}e_k, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ t_me_m' = b_{m1}e_1 + b_{m2}e_2 + \ldots + b_{mk}e_k, \end{gathered}$$

где $e_j \in M$, $j = 1, 2, \ldots, k$. Домножив равенство (31) на наименьшее общее кратное чисел $t_1, t_2, \ldots, t_m$ и подставив вместо t_ie_i' $(i = 1, \ldots, m)$ их выражения с вышеприведенной системы в (31), получаем равенство

$$tt'e_1 = c_1e_1 + c_2e_2 + \ldots + c_ke_k \ .$$

Поскольку $e_1 \in M$, то это равенство возможно только тогда, когда $tt' = c_1, c_2 = c_3 = \ldots = c_k = 0$. Но последние равенства редуцируют вышеприведенную систему равенств к виду

$$t_1e_1' = b_{11}e_1,\ t_1e_2' = b_{21}e_1, \ldots,\ t_me_m' = b_{m1}e_1,$$

а это значит, что $e_1 = e_1' = e_2' = \ldots = e_m'$. Это возможно только тогда, когда $e_1 = e_i'$, для некоторого значения индекса i, потому что в противном случае это противоречило бы тому, что элементы в M' все разные. Следовательно, $e_1 = e_i' \in M'$ i $M \subseteq M'$. Справедливость обратного включения вытекает из симметричности M и M'. ■

Следствие 4. *Пусть* S, S_1 *- две СЛОДУ и* $TSS(\emptyset, S) = M$, $TSS(\emptyset, S_1) = M_1$.

a) $TSS(\emptyset, S_1 \wedge S_2) = TSS(\emptyset, S_2 \wedge S_1)$;

b) Если $TSS(M, S_1) = M$, *то* $\mathcal{S}ol(S) \subseteq \mathcal{S}ol(S_1)$, *где* $\mathcal{S}ol(S)$ *- множество всех решений СЛОДУ* S;

c) S_1 *и* S_2 *эквивалентны тогда и только тогда, когда* $M = M_1$.

Из следствия 4 вытекает, что в процессе построения TSS можно определять (и при необходимости удалять) линейно зависимые уравнения системы, а также вычислять ранг матрицы данной системы.

7.3. Сложность проверки совместности СЛОДУ

Нетрудно показать, что в общем случае временная сложность алгоритма определения совместности СЛОДУ экспоненциальная по числу уравнений в системе. Действительно, рассмотрим систему вида

$$S = \begin{cases} -x_1 + x_2 + 0x_3 + 0x_4 + x_5 + 0x_6 + 0x_7 = 0, \\ -x_1 + 0x_2 + x_3 + 0x_4 + 0x_5 + x_6 + 0x_7 = 0, \\ -x_1 + 0x_2 + 0x_3 + x_4 + 0x_5 + 0x_6 + x_7 = 0. \end{cases}$$

TSS данной системы имеет 2^3 векторов. Нетрудно показать, что такого рода системы, т.е. системы состоящие из p уравнений с $2p+1$ неизвестными и построенные путем присоединения к столбцу из -1 размерности p двух единичных квадратных матриц размерности $p \times p$, имеют TSS, состоящее из 2^p элементов. Следует заметить, что данная СЛОДУ соответствует системе неравенств вида

$$S' = \left\{ \begin{array}{lcl} x_1 & \leq & 1, \\ x_2 & \leq & 1, \\ \ldots & \ldots & \ldots \\ x_p & \leq & 1. \end{array} \right.$$

Таким образом, имеет место

Теорема 22. *Сложность алгоритма определения совместности СЛОДУ имеет экспоненциальную сложность по числу уравнений в системе.*

Отметим также, что в случае такого рода систем, TSS совпадает с базисом всего множества решений данной СЛОДУ.

СЛОДУ, совместность которых определяется за полиномиальное время. В связи с вышесказанным, интересно было бы указать класс СЛОДУ, для которых этот алгоритм работает полиномиальное время. Один из таких классов систем дает следующее утверждение.

Теорема 23. *Если матрица СЛОДУ S имеет вид $A = (B|B')$, где A – матрица размерности $p \times (p+q)$, B – матрица размерности $p \times q$, с положительными элементами, а B' – диагональная матрица размерности $p \times p$ с отрицательными элементами на диагонали, то система S всегда совместна и ее TSS вычисляется за время $O(p(p+q))$.*

Доказательство теоремы очевидным образом следует из построения TSS. Действительно, на каждом шаге алгоритма число элементов в TSS не превосходит величины $p+q$. Следовательно, верхняя оценка временной сложности ограничена величиной $p(p+q)$. ■

Пример 17. Определить совместность СЛОДУ

$$S = \left\{ \begin{array}{rcrcrcrcrcrcl} 5x_1 & + & 0x_2 & + & 3x_3 & - & 4x_4 & + & 0x_5 & + & 0x_6 & = & 0, \\ 1x_1 & + & 2x_2 & + & x_3 & + & 0x_4 & - & 3x_5 & + & 0x_6 & = & 0, \\ 0x_1 & + & 1x_2 & + & 2x_3 & + & 0x_4 & + & 0x_5 & - & 2x_6 & = & 0. \end{array} \right.$$

В данной системе

$$B = \begin{pmatrix} 5 & 0 & 3 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \quad B' = \begin{pmatrix} -4 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{pmatrix}.$$

TSS для этой системы состоит из трех векторов

$$(12, 0, 0, 15, 4, 0),\ (0, 6, 0, 0, 4, 3),\ (0, 0, 12, 9, 4, 12).$$

Базис множества решений этой системы состоит из 11 векторов. ♠

7.4. Критерий совместности СЛНДУ

Пусть

$$S = \left\{ \begin{array}{rcrcrcrcl} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & b_1, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & b_p \end{array} \right.$$

– СЛНДУ, где $a_{ij},\ b_i \in \mathcal{Z},\ x_j \in \mathcal{N},\ i = 1,\ldots,p,\ j = 1,\ldots,q$. Перейдем от системы S к системе S':

$$S' = \left\{ \begin{array}{ccccccccccc} L_1(x)' & = & \alpha_{11}x_1 & + & \ldots & + & \alpha_{1q}x_q & = & 0, \\ L_2(x)' & = & \alpha_{21}x_1 & + & \ldots & + & \alpha_{2q}x_q & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ L_{p-1}(x)' & = & \alpha_{p-11}x_1 & + & \ldots & + & \alpha_{p-1q}x_q & = & 0, \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & b_p, \end{array} \right.$$

где $b_p \neq 0$. Этот переход осуществляется с помощью процедуры элиминации свободных членов.

а) Взять i-е уравнение системы S, у которого $b_i \neq 0$; пусть, для определенности, это будет b_p и $b_p > 0$.

б) Для всех $i = 1$ до $p-1$ выполнить

Если $b_i < 0$ то заменить i-е уравнение системы S линейной комбинацией вида

$$-b_i(L_p(x) - b_p) + b_p(L_i(x) - b_i) = -b_iL_p(x) + b_pL_i(x) = 0$$

иначе если $b_i > 0$ то заменить i-е уравнение системы S линейной комбинацией вида

$$b_i(L_p(x) - b_p) + b_p(-L_i(x) + b_i) = b_iL_p(x) - b_pL_i(x) = 0$$

иначе оставить i-е уравнение без изменений.

Обоснованием этой процедуры есть

Теорема 24. *Система S совместна тогда и только тогда, когда совместна система S'.*

Доказательство. Пусть система S совместна, B – базис множества ее решений и $\gamma = (\gamma_1,\ldots,\gamma_q) \in B$, тогда $L_i(\gamma) = b_i, i = 1,2,\ldots,p-1$, и $L_p(\gamma) = b_p$. Но отсюда получаем $L_i'(\gamma) = -b_iL_p(\gamma) + +b_pL_i(\gamma) = b_pb_i - b_pb_i = 0$, т.е. γ – решение системы S'.

Пусть система S' совместна и $\gamma = (\gamma_1,\ldots,\gamma_q)$ ее решение. Поскольку последнее уравнение системы S' такое же как и в системе S, то $L_p'(\gamma) = L_p(\gamma) = b_p$. Но тогда для всех $i = 1,2,\ldots,p-1$ имеем $L_i'(\gamma) = b_p(L_i(\gamma) - b_i) - b_i(L_p(\gamma) - b_p) = b_p(L_i(\gamma) - b_i) - b_i0 = 0$. Отсюда получаем, что $L_i(\gamma) - b_i = 0$ или $L_i(\gamma) = b_i$ (поскольку $b_p \neq 0$), т.е. $\gamma = (\gamma_1,\ldots,\gamma_q)$ – решение системы S. ■

Пусть $M_{p-1}' = \{e_1', e_2', \ldots, e_k'\}$ – TSS подсистемы системы S, состоящей из первых $p-1$ уравнений. Далее, пусть $d_1, d_2, \ldots, d_k$ – значения $L_p(x)$ на векторах $e_1', e_2', \ldots, e_k'$ соответственно. Имеет место

Теорема 25. *Система S' совместна тогда и только тогда, когда уравнение*

$$d_1u_1 + d_2u_2 + \ldots + d_ku_k - tb_p = 0 \tag{32}$$

имеет хотя бы одно решение $(\alpha_1, \alpha_2, \ldots, \alpha_k, \gamma)$ в множестве $\mathcal{N}$ такое, что координаты вектора $v = \alpha_1e_1' + \alpha_2e_2' + \ldots + \alpha_ke_k'$ кратны числу γ, где $e_i' \in M_{p-1}', \gamma \neq 0, \alpha_i, t, \gamma \in \mathcal{N}, i = 1,2,\ldots k$.

Доказательство. Пусть $(\alpha_1, \alpha_2, \ldots, \alpha_k, \gamma)$ решение уравнения (32), удовлетворяющее условиям теоремы, т.е. $d_1\alpha_1 + d_2\alpha_2 + \ldots + d_k\alpha_k = \gamma b_p$. Тогда для вектора $v' = \alpha_1e_1' + \ldots + \alpha_ke_k'$ имеем

$$L_p(v') = \alpha_1 L_p(e'_1) + \ldots + \alpha_k L_p(e'_k) = d_1\alpha_1 + d_2\alpha_2 + \ldots + d_k\alpha_k = \gamma b_p.$$

и в силу того, что координаты вектора v' кратны γ получаем $L_p(1/\gamma v) = b_p$, т.е. вектор $v = 1/\gamma v'$ является решением системы S'.

Обратно, пусть система S' совместна и $v' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_k)$ ее решение. Тогда в силу теоремы 16 вектор v' имеет представление

$$tv' = a_1 e'_1 + \ldots + a_k e'_k,$$

где $e'_i \in M'_{p-1}, t, \neq 0, a_i, t \in \mathcal{N}, i = 1, 2, \ldots, k$. Поскольку вектор v' решение S', то

$$L_p(tv') = tL_p(v') = a_1 L_p(e'_1) + \ldots + a_k L_p(e'_k) = d_1 a_1 + d_2 a_2 + \ldots + d_k a_k = tb_p.$$

Полученное уравнение должно иметь хотя бы одно решение $(\alpha_1, \alpha_2, \ldots, \alpha_k, \gamma)$, т. к. иначе это противоречит тому, что система S' совместна. Более того, должно существовать хотя бы одно решение уравнения (32) такое, что координаты вектора $\alpha_1 e'_1 + \alpha_2 e'_2 + \ldots + \alpha_k e_k$ кратны числу γ, т. к. иначе это противоречит существованию вектора v', являющегося решением S'. ■

Следствие 5 (достаточное условие совместности) . *СЛНДУ S совместна, если уравнение $d_1u_1 + d_2u_2 + \ldots + d_ku_k = b_p$ имеет хотя бы одно решение в множестве $\mathcal{N}$.*

Действительно, если такое уравнение имеет хотя бы одно решение, то уравнение (32) имеет решение, у которого последняя координата равна 1. А это значит, что система S совместна.

Пример 18. a) Выясним, будет ли совместна ниже приведенная система неоднородных уравнений, если за столбец свободных членов принять столбец коэффициентов при неизвестном x_5.

$$S = \left\{ \begin{array}{lcrcrcrcrcrcl} L_1(x) & = & -x_1 & + & 2x_2 & + & 4x_3 & - & 3x_4 & + & x_5 & = & 0, \\ L_2(x) & = & 2x_1 & + & 3x_2 & - & 4x_3 & - & x_4 & + & x_5 & = & 0, \\ L_3(x) & = & 0x_1 & + & x_2 & - & 5x_3 & + & x_4 & + & x_5 & = & 0. \end{array} \right.$$

Приведем систему S к системе S', исключая последний член в уравнениях. Получаем систему

$$S' = \left\{ \begin{array}{lcrcrcrcrcrcl} L'_1(x) & = & -x_1 & + & x_2 & + & 9x_3 & - & 4x_4 & = & 0, \\ L'_2(x) & = & 2x_1 & + & 2x_2 & + & x_3 & - & 2x_4 & = & 0, \\ L'_3(x) & = & 0x_1 & + & x_2 & - & 5x_3 & + & x_4 & = & -1. \end{array} \right.$$

TSS для первого уравнения системы S' имеет вид:

$$M'_1 = \{(1,1,0,0), (9,0,1,0), (0,4,0,1), (0,0,4,9)\}.$$

Значения на этих векторах для второго уравнения системы S' таковы: 4, 19, 6, -14. Получаем TSS

$$M'_2 = \{(14,0,10,19), (0,14,6,17)\}.$$

Значения на этих векторах для последнего уравнения системы S' таковы: -31, 1. Составляем уравнение

$$-31u_1 + u_2 + t = 0.$$

Это уравнение имеет корень (1,30,1) который порождает вектор (14,420,190,529). Этот вектор, как нетрудно убедиться, является решением системы S', т. е. СЛНДУ, соответствующая системе S, совместна.

b) Рассмотрим СЛНДУ

$$S = \left\{ \begin{array}{rcrcrcrcl} 4x_1 & + & 2x_2 & - & 3x_3 & - & 2x_4 & = & 1, \\ 2x_1 & - & x_2 & - & 4x_3 & + & x_4 & = & -5, \\ 0x_1 & + & 2x_2 & + & 4x_3 & + & 0x_4 & = & 9. \end{array} \right.$$

Преобразуем систему S к виду

$$\left\{ \begin{array}{rcrcrcrcl} 22x_1 & + & 9x_2 & - & 19x_3 & - & 9x_4 & = & 0, \\ 18x_1 & + & x_2 & - & 16x_3 & + & 9x_4 & = & 0, \\ 2x_1 & - & x_2 & - & 4x_3 & + & x_4 & = & -5. \end{array} \right.$$

Строим TSS

$$M_1' = \{(19,0,22,0),(0,19,9,0),(9,0,0,22),(0,1,0,1)\}.$$

Значения на этих векторах для второго уравнения системы S' после сокращения на общий делитель 5 таковы: -2, -25, 72, 2. Им соответствует TSS

$$M_2' = \{(63,0,72,2),(0,63,18,25)\}.$$

Значения на этих векторах для последнего уравнения системы S' таковы: -160, -110. Составляем уравнение, сократив коэффициенты на общий делитель 5,

$$32u_1 + 22u_2 - t = 0.$$

Это уравнение имеет решения (1,0,32) и (0,1,22), которые не удовлетворяют условиям теоремы 25 над множеством натуральных чисел $\mathcal{N}$. Следовательно, СЛНДУ S несовместна. В самом деле, в системе S последнее уравнение не имеет корней не только в области натуральных чисел, но и в области целых чисел. ♠

8. Методы построения базиса решений СЛОДУ над $\mathcal{N}$

Приведем методы построения базиса множества всех решений СЛОДУ, которые хорошо себя зарекомендовали в практических приложениях. Сначала рассмотрим случай одного уравнения и метод построения базиса множества всех его решений.

8.1. Метод Фортенбахера

Метод Фортенбахера [45] и его обобщение – метод Контежан–Деви [47] были, по-видимому, первыми алгоритмами, которые не использовали идею сокращения числа неизвестных и уравнений.

Метод Фортенбахера решения ЛОДУ в множестве натуральных чисел $\mathcal{N}$ сводится к такому простому процессу. Пусть дано ЛОДУ:

$$a(x) = a_1x_1 + a_2x_2 + \ldots + a_qx_q = 0, \tag{33}$$

где $a_i \in \mathcal{Z},\ x_i \in \mathcal{N}$ и $e_1 = (1,0,\ldots,0,0), e_2 = (0,\ 1,\ldots,0,0),\ldots,e_q = (0,0,\ldots,0,1)$ – векторы канонического базиса пространства $\mathcal{N}^q$. Идея Фортенбахера состоит в том, чтобы начинать поиск минимальных базисных решений ЛОДУ (33) с канонических векторов $e_1, e_2, \ldots, e_q$. При этом, если некоторый текущий вектор $x = (x_1, \ldots, x_q)$ еще не является решением (33) и

– $a_1x_1 + \ldots + a_qx_q < 0$, то увеличивать значение на 1 того x_j, при котором коэффициент $a_j > 0$;

– $a_1x_1 + \ldots + a_qx_q > 0$, то увеличивать значение на 1 того x_j, при котором коэффициент $a_j < 0$. (Заметим, что если все $a_i > 0$ или все $a_i < 0$, то уравнение (33) имеет только тривиальное решение $\overline{0} = (0,\ldots,0)$).

Вышеприведенное условие **(условие Фортенбахера)** формально записывается так:

• (C_1) увеличить на 1 то x_j, для которого $a(x) \cdot a(e_j) < 0$, где $x = (x_1,\ldots,x_q),\ a(e_j) = a_j,\ e_j = (0,\ldots,0,1,0,\ldots,0)$ – вектор канонического базиса.

Алгоритм Фортенбахера имеет вид:

FORTEN $(a(x))$
Вход: Коэффициенты ЛОДУ $a(x) = 0$.
Выход: Множество B – базис множества решений ЛОДУ $a(x) = 0$.
Метод:

begin
  $P := \{e_1, e_2, \ldots, e_q\}$;
  $B := \emptyset$;
  while $P \neq \emptyset$ do
    $B := B \cup \{x \in P | a(x) = 0\}$;
    $Q := \{x \in P \setminus B | \forall s \in B\ s \not\leq x\}$;
    $P := \{x + e_j | x \in Q \text{ and } a(x) \cdot a(e_j) < 0\}$
  od;
end

Характеристику этого алгоритма дает

Теорема 26. *Алгоритм FORTEN всегда заканчивает свою работу и каждое решение ЛОДУ $a(x) = 0$, найденное этим алгоритмом, является минимальным [45].*

Пример 19. Найти базис множества решений ЛОДУ $-x_1 + x_2 + 2x_3 - 3x_4 = 0$.
На векторах канонического базиса это уравнение принимает такие значения $-1, 1, 2, -3$, а весь процесс решения показан на рис. 8.1.

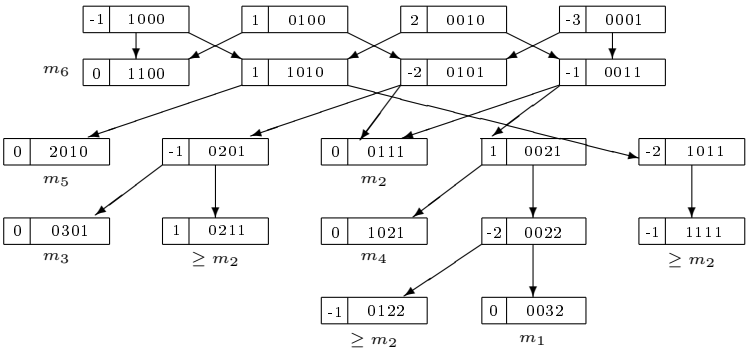


Рис. 8.1. Процесс решения ЛОДУ

В результате получаем такие базисные решения:

$$m_1 = (0, 0, 3, 2),\ m_2 = (0, 1, 1, 1),\ m_3 = (0, 3, 0, 1),\ m_4 = (1, 0, 2, 1),\ m_5 = (2, 0, 1, 0),\ m_6 = (1, 1, 0, 0).\ \spadesuit$$

8.2. Метод Гаусса

Исторически первым методом построения базиса множества решений СЛОДУ, наверное, следует считать метод сокращения числа уравнений Гаусса. Этот метод успешно используется и конкурирует с многими современными методами. Суть его состоит в следующем.

Пусть $S = (L(x) = 0)\ \wedge\ S'$ – некоторая СЛОДУ, где S' подсистема системы S, и пусть $B = \{e_1, e_2, \ldots, e_k\}$ – базис множества решений уравнения $L(x) = 0$. Поскольку любое решение s системы S является решением уравнения $L(x) = 0$, то это решение можно записать в виде такой линейной комбинации

$$s = c_1 e_1 + c_2 e_2 + \ldots + c_k e_k, \tag{34}$$

где $c_i \in \mathcal{N}$, $e_i \in B$, $i = 1, 2, \ldots, k$. Подставив вектор s в уравнения подсистемы S', получим систему с $p - 1$ уравнениями относительно неизвестных $c_1, c_2, \ldots, c_k$. Решая полученную систему и найдя ее базис B', строим все базисные решения системы S с помощью подстановки вместо коэффициентов c_i в выражении (34) их значения, удаляя не минимальные решения из полученного множества. Таким образом, алгоритм Гаусса сводит

процесс решения СЛОДУ к решению одного уравнения.

Пример 20. Рассмотрим систему

$$S = \left\{ \begin{array}{lcrcrcrcl} L_1(x) & = & -x_1 & + & x_2 & + & 2x_3 & -3x_4 & = & 0, \\ S' & = & -x_1 & + & 3x_3 & + & -2x_3 & -x_4 & = & 0. \end{array} \right.$$

Базис множества решений уравнения $L_1(x) = 0$ был найден в предыдущем примере и состоит из таких векторов:

$$e_1 = (0,0,3,2),\ e_2 = (0,1,1,1),\ e_3 = (0,3,0,1),\ e_4 = (1,0,2,1),\ e_5 = (2,0,1,0),\ e_6 = (1,1,0,0).$$

Построив линейную комбинацию

$$s = c_1e_1 + c_2e_2 + c_3e_3 + c_4e_4 + c_5e_5 + c_6e_6$$

и подставив ее в уравнение S', получаем уравнение

$$S'(s) = c_1S'(e_1) + c_2S'(e_2) + c_3S'(e_3) + c_4S'(e_4) + c_5S'(e_5) + c_6S'(e_6) = \\ = -8c_1 + 0e_2 + 8e_3 - 6e_4 - 4e_5 + 2e_6 = 0.$$

Полученное уравнение имеет такие базисные решения:

$$s_1 = (0,1,0,0,0,0),\ s_2 = (1,0,1,0,0,0),\ s_3 = (0,0,1,0,2,0), s_4 = (0,0,0,0,1,2),\\ s_5 = (0,0,0,1,0,3),\ s_6 = (0,0,1,1,1,1),\ s_7 = (1,0,0,0,0,4),\ s_8 = (0,0,2,2,1,0),\\ s_9 = (0,0,1,2,0,2),\ s_{10} = (0,0,2,3,0,1),\ s_{11} = (0,0,3,4,0,0).$$

Этим решениям соответствуют такие решения системы S:

$$e'_1 = (0,1,1,1),\ (0,3,3,3) \geq e'_1,\ (4,3,2,1) \geq e'_1,\ e'_2 = (4,2,1,0),\ (4,3,2,1) \geq e'_1,\\ (4,4,3,2) \geq e'_1,\ (4,4,3,2) \geq e'_1,\ (4,6,5,4) \geq e'_1,\ (4,5,4,3) \geq e'_1,\ (4,7,6,5) \geq e'_1,\ (4,9,8,7) \geq e'_1.$$

Следовательно, система S имеет базис множества решений $B = \{(0,1,1,1),(4,2,1,0)\}$. ♠

8.3. Метод Контежан–Деви

Этот метод, как отмечалось выше, является обобщением метода Фортенбахера. Это обобщение состоит в следующем.

Пусть дана СЛОДУ (28) и $a(x)$ означает вектор-столбик этой СЛОДУ вида:

$$a(x) = \left(\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1q}x_q \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2q}x_q \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ a_{p1}x_1 & + & a_{p2}x_2 & + & \cdots & + & a_{pq}x_q \end{array} \right),$$

где $x = (x_1, x_2, \cdots, x_q)$. При этих обозначениях условие Фортенбахера $(C1)$ обобщается к такому [47]:

• (C_p): увеличивать на 1 то x_j, которое удовлетворяет условию $a(x) * a(e_j) < 0$, где * – операция скалярного произведения векторов.

Алгоритм Контежан–Деви имеет вид:

CONT-DEVIE $(a(x))$
Вход: Матрица системы $a(x) = 0$.
Выход: Множество B – базис множества решений системы $a(x) = 0$.
Метод:
begin
$P := \{e_1, e_2, \ldots, e_q\}$;
$B := \emptyset$;
while $P \neq \emptyset$ do

$B := B \cup \{x \in P | a(x) = 0\}$;
$Q := \{x \in P \setminus B | \forall s \in Bs \not\leq x\}$;
$P := \{x + e_j | x \in Q \text{ and } a(x) * a(e_j) < 0\}$
od;
end

Характеристика этого алгоритму дает

Теорема 27. *Алгоритм CONT-DEVIE всегда завершает свою работу и каждое решение СЛОДУ (28), найденное алгоритмом CONT-DEVIE, является минимальным [47].*

Пример 21. Найти минимальные решения СЛОДУ

$$S = \left\{ \begin{array}{rcrcrcrcl} -x_1 & + & x_2 & + & 2x_3 & - & 3x_4 & = & 0, \\ -x_1 & + & 3x_2 & - & 2x_3 & - & x_4 & = & 0. \end{array} \right.$$

Процесс решения показан ниже на рис. 8.2.

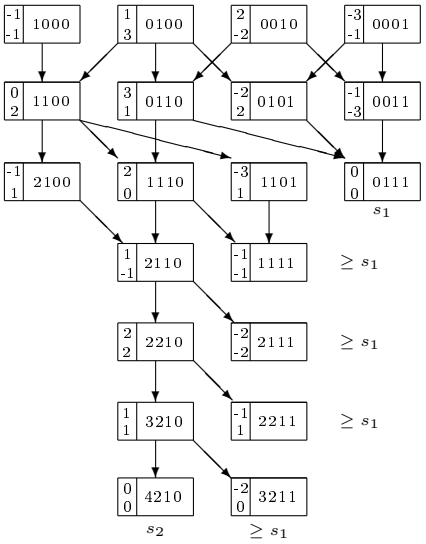


Рис. 8.2. Процесс решения СЛОДУ

Таким образом, минимальными решениями данной системы S являются векторы $s_1 = (0, 1, 1, 1)$ и $s_2 = (4, 2, 1, 0)$.♠

Повышение эффективности алгоритма Контежан–Деви. Из приведенного примера видно, что некоторые векторы в процессе работы алгоритма вычисляются повторно. Эти избыточные вычисления можно удалить, если ввести порядок на вершинах дерева, которое порождается алгоритмом. Рассмотрим один з таких порядков (порядки могут быть разными).

Если вершина x имеет двух разных сыновей $x+e_{j_1}$ и $x+e_{j_2}$, то $x+e_{j_1} <_x x+e_{j_2} \Leftrightarrow j_1 < j_2$.

Правило замораживания компонент состоит в том, что когда $x + e_{j_1} <_x x + e_{j_2}$, то j_1-та компонента замораживается в поддереве, корнем которого является вершина $x + e_{j_2}$: она не может увеличиваться даже тогда, когда условие (C_p) выполняется!

Пример 22. Рассмотрим систему из предыдущего примера. Первые векторы – векторы канонического базиса – упорядочены относительно порядка $<_0$. Замороженные компоненты подчеркнуты.

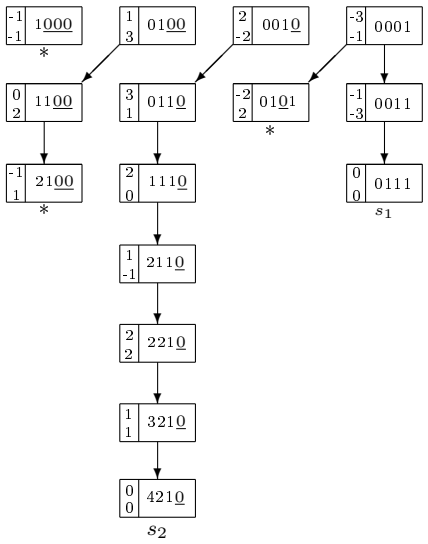


Рис. 8.3. Процесс решения СЛОДУ с замораживанием

Таким образом, этим алгоритмом было порождено на 5 вершин меньше, чем предыдущим. ♠

Некоторые методы во время поиска решений СЛНДУ и СЛОДУ сводятся к определению границ, в рамках которых находятся все базисные решения данной системы уравнений, а потом применяются методы вычисления этих решений в этих границах. Методы Потье и Доменджуда, приводимые ниже, как раз и работают по такой схеме.

8.4. Методы Потье

Первый метод Потье. Л. Потье [51] предложил условие наращивания компонент во время поиска векторов-решений СЛОДУ, отличное от условия C_p:

• (PC_p): увеличивать x_j на 1, если для всех l l-ая компонента $a(x_1,\dots,x_{j-1},x_j+1,x_{j+1},\dots,x_q)$ принадлежит интервалу

$$[-\sum_{l=1}^{q}\max\{a_{il},0\},\sum_{l=1}^{q}\max\{-a_{il},0\}].$$

Имеет место

Теорема 28. *Пусть S – СЛОДУ вида (28) и $m=(m_1,m_2,\dots,m_q)$ – ее минимальное решение. Тогда*

$$\sum_{l=1}^{q}m_l\le\prod_{i=1}^{p}(\sum_{j=1}^{q}|a_{ij}|+1).$$

Имеется более точная оценка решений СЛОДУ в терминах ранга матрицы системы S [51].

Теорема 29. *Пусть A – матрица СЛОДУ S размерности $p\times q$ ранга r. Каждое минимальное решение $m=(m_1,m_2,\dots,m_q)$ системы S удовлетворяет неравенству*

$$\max_{1\le j\le q}|m_j|\le(p-r)\left(\frac{\sum\limits_{i,j}|c_{ij}|}{r}\right)^r.$$

Второй метод Потье. Этот метод поиска базиса множества решений СЛОДУ состоит в применении известного алгоритма Бухбергера, который выполняет построение базиса идеала в кольце многочленов (базиса Грьобнера). Алгоритм Бухбергера описан во многих работах и его характеристику можно найти, например, в [4]. Поэтому этот алгоритм здесь не рассматривается. Следует заметить, что такой способ построения базиса решений СЛОДУ на практике себя не оправдал, поскольку его выполнение имеет большие накладные расходы (большое количество промежуточных вычислений), которые снижают его эффективность [47, 51].

8.5. Метод Доменджуда

Этот метод базируется на таких двух шагах: сначала строится минимальное порождающее множество минимальных решений, а потом с помощью комбинаций полученных решений строится все множество базисных решений [49]. Алгоритм, реализующий этот метод, состоит из двух алгоритмов, которые реализуют указанные шаги. Пред применением этих алгоритмов необходимо убедиться в том, что

а) матрица A (размерности $p \times q$) системы S имеет ранг, равный p (система S не имеет линейно зависимых уравнений);

б) $p \leq q+1$ (система S имеет по крайней мере одно решение над Z^q).

Обоснованием первого шага и алгоритма его выполнения является

Теорема 30. *Пусть m – минимальное решение из минимального порождающего множества решений S. Тогда существует подмножество $\{i_1, i_2, \ldots, i_{p+1}\}$ из множества $\{1, 2, \ldots, q\}$ и натуральное число k такие, что*

$$km = \begin{vmatrix} e_{i_1} & e_{i_2} & \ldots & e_{i_{p+1}} \\ a(e_{i_1}) & a(e_{i_2}) & \ldots & a(e_{i_{p+1}}) \end{vmatrix}, \tag{35}$$

где прямые скобки означают детерминант.

Таким образом, процесс построения всех минимальных решений минимального порождающего множества решений сводится к вычислению детерминантов вида (35) для всех подмножеств $\{i_1, i_2, \ldots, i_{p+1}\}$ из множества $\{1, 2, \ldots, q\}$ и поиска всех ненулевых векторов, координаты которых одновременно или все положительные или все отрицательные. Потом найденные векторы должны быть поделены на НОД своих компонент.

Второй шаг алгоритма обосновывает

Теорема 31. *Все минимальные решения S являются линейными комбинациями некоторых линейно независимых минимальных решений из минимального порождающего множества решений.*

Ограничения на величину коэффициентов таких линейных комбинаций вытекают из таких преобразований.

Пусть a – вектор, который является некоторой линейной комбинацией, о которой говорилось в теореме 31, и M – матрица, столбики которой являются линейно независимыми векторами из минимального множества образующих. Тогда существует матрица U, такая что $det(U) = \pm 1$ и

$$UM = \begin{pmatrix} T \\ 0 \end{pmatrix},$$

где T – верхняя треугольная матрица с положительными диагональными коэффициентами (существование такой матрицы можно показать с помощью алгоритма последовательного сокращения неизвестных Гаусса). Вектор $M * a$ имеет натуральные координаты тогда и только тогда, когда $T * a$ имеет такие же координаты, т. е. тогда и только тогда, когда $a = T^{-1}x$, где x – некоторый вектор с целыми координатами. Тогда вектор $a = T^{-1}x - [T^{-1}x] = 1/d[T^{-1}x]_d$, где $d = det(T)$, $[T^{-1}x]$, $[T^{-1}x]_d$ – векторы целых и дробных частей соответственно, полученные в результате деления координат вектора $T^{-1}x$ на d. Поскольку T и $det(T)$ известны, то необходимо рассмотреть только конечное число векторов x, таких, что их компоненты лежат в интервале $\{0, \ldots, d\}$. С этого конечного множества векторов x находят искомое множество векторов a.

Пример 23. Пусть дано систему S, которая рассматривалась в предыдущих примерах,

$$S = \left\{ \begin{array}{rcrcrcrcl} -x_1 & + & x_2 & + & 2x_3 & - & 3x_4 & = & 0, \\ -x_1 & + & 3x_2 & - & 2x_3 & - & x_4 & = & 0. \end{array} \right.$$

Множество индексов для этой системы состоит из четырех элементов $\{1, 2, 3, 4\}$ (так как $q = 4$), а подмножества этого множества состоят из трех элементов (та как $p = 2$). Число разных подмножеств буде $C_4^3 = C_4^1 = 4$:

$$\{1, 2, 3\}, \quad \{1, 2, 4\}, \quad \{1, 3, 4\}, \quad \{2, 3, 4\}.$$

Для первого из подмножеств имеем

$$\left| \begin{array}{ccc} e_1 & e_2 & e_3 \\ a(e_1) & a(e_2) & a(e_3) \end{array} \right| = \left| \begin{array}{ccc} e_1 & e_2 & e_3 \\ -1 & 1 & 2 \\ -1 & 3 & -2 \end{array} \right| = -8e_1 - 4e_2 - 2e_3 = -2(4, 2, 1, 0).$$

Для второго из подмножеств имеем

$$\left| \begin{array}{ccc} e_1 & e_2 & e_4 \\ a(e_1) & a(e_2) & a(e_4) \end{array} \right| = \left| \begin{array}{ccc} e_1 & e_2 & e_4 \\ -1 & 1 & -3 \\ -1 & 3 & -1 \end{array} \right| = 8e_1 + 2e_2 - 2e_4,$$

что не является решением, поскольку координаты полученного вектора имеют разные знаки.

Для третьего из подмножеств имеем

$$\left| \begin{array}{ccc} e_1 & e_3 & e_4 \\ a(e_1) & a(e_3) & a(e_4) \end{array} \right| = \left| \begin{array}{ccc} e_1 & e_3 & e_4 \\ -1 & 2 & -3 \\ -1 & -2 & -1 \end{array} \right| = -8e_1 + 2e_3 - 4e_4,$$

что тоже не является решением, поскольку координаты полученного вектора имеют разные знаки.

Для четвертого из подмножеств имеем

$$\left| \begin{array}{ccc} e_2 & e_3 & e_4 \\ a(e_2) & a(e_3) & a(e_4) \end{array} \right| = \left| \begin{array}{ccc} e_2 & e_3 & e_4 \\ 1 & 2 & -3 \\ 3 & -2 & -1 \end{array} \right| = -8e_2 - 8e_3 - 8e_4 = -8(0, 1, 1, 1).$$

Объединяя первое и четвертое решения, получаем такое минимальное порождающее множество решений, которое в данном случае (как следует из рассмотренных выше примеров) и составляет базис $B = \{(0, 1, 1, 1), \ (4, 2, 1, 0)\}$ множества решений системы S. ♠

Как следует из TSS-алгоритма и алгоритма Доменджуда, с помощью минимального порождающего множества решений можно охарактеризовать проблему совместности СЛОДУ.

8.6. Критерий совместности СЛОДН

При рассмотрении СЛОДУ и характеристике ее множества решений отмечалось, что это множество имеет конечный базис, который составляют ее минимальные решения. В случае

СЛОДН множество всех ее решений тоже имеет конечный базис, но этот базис, вообще говоря, состоит не только из ее минимальных решений. Простейшим подтверждением этого является пример СЛОДН, которая имеет единственное неравенство $x_1 - x_2 \geq 0$. Для этого неравенства базис множества всех решений состоит из векторов (1,0) и (1,1), где вектор (1,1) $>$ (1,0) не является минимальным. В связи с этим возникает вопрос: как построить базис и минимальное порождающее множество решений для данной СЛОДН? Ответ на это вопрос вытекает из возможности сведения проблемы решения СЛОДН к проблеме решения СЛОДУ путем введения дополнительных неизвестных.

Анализ TSS-алгоритма показал, что построение TSS для СЛОДН можно выполнить без введения дополнительных неизвестных. Перейдем к рассмотрению результатов этого анализа в соответствии с [5].

Пусть

$$S = \left\{ \begin{array}{lcccccccc} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & \geq & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & \geq & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & \geq & 0 \end{array} \right. \tag{36}$$

– СЛОДН. Как и в случае СЛОДУ с помощью функции $L_1(x)$, множество векторов канонического базиса можно розбить на три группы:

$$M_1^0 = \{e|L_1(e) = 0\},\ M_1^+ = \{e|L_1(e) > 0\},\ M_1^- = \{e|L_1(e) < 0\}.$$

Рассмотрим множество

$$M_1' = M_1^0 \cup M_1^+ \cup \{y_{ij}^0 | y_{ij}^0 = -L_1(e_i)e_j + L_1(e_j)e_i, e_j \in M_0^+, e_i \in M_0^-\}.$$

Используя функцию $L_2(x)$, разобъем элементы множества M_1' аналогично предыдущему тоже на три группы

$$M_2^0 = \{e|L_2(e) = 0\},\ M_2^+ = \{e|L_2(e) > 0\},\ M_2^- = \{e|L_2(e) < 0\}$$

и построим множество

$$M_2' = M_2^0 \cup M_2^+ \cup \{y_{ij}^0 | y_{ij}^0 = -L_2(e_i)e_j + L_2(e_j)e_i, e_j \in M_1^+, e_i \in M_1^-\}.$$

Пусть таким способом построено множество

$$M_i' = M_i^0 \cup M_i^+ \cup \{y_{kj}^0 | y_{kj}^0 = -L_i(e_k)e_j + L_i(e_j)e_k, e_j \in M_i^+,\ e_k \in M_i^-\}$$

с помощью функции $L_i(x)$ и это множество непустое. Имеет место

Теорема 32. *Векторы из TSS M_i' составляют множество, с помощью которого произвольное решение x подсистемы $S' = L_1(x) \geq 0 \wedge\ \wedge L_2(x) \geq 0 \wedge \ldots \wedge L_i(x) \geq 0$ системы S можно представить в виде неотрицательной линейной комбинации*

$$kx = a_1e_1 + \ldots + a_te_t + b_1e_1' + \ldots + b_re_r' + d_1y_{i1}^0 + \ldots + d_ly_{il}^0, \tag{37}$$

где $e_k \in M_i^0,\ e_j' \in M_i^+,\ y_{ij} \in \{y_{ij}^0 | y_{ij}^0 = -L_i(e_i)e_j + L_i(e_j)e_i,\ e_j \in M_i^+,\ e_k \in M_i^-\}$.

Доказательство. Заметим, что произвольный вектор из M_{i+1}' является решением системы S'. Следовательно, для доказательства теоремы необходимо показать, что любое решение системы S' можно представить в виде (37). Доказательство индукцией по числу неравенств в системе S'.

Базис индукции имеет место на основании леммы 9. Действительно, поскольку любое решение x' неравенства $L_1(x) \geq 0$ можно записать в виде неотрицательной линейной комбинации через векторы канонического базиса M_0', то применяя лемму 9 столько раз, сколько нужно, удалим векторы множества M_1^-, которые входят в запись x'. Получаем представление вектора x' в виде (37).

Пусть утверждение теоремы справедливо для любой подсистемы S_i системы S, которая состоит из первых $i < p$ неравенств, и покажем, что оно справедливо и для подсистемы S_{i+1} системы S.

Пусть x – решение системы S_{i+1}, тогда согласно предположению индукции вектор x можно представить в виде

$$k'x = a_1'e_1' + \ldots + a_t'e_t' + b_1'e_1'' + \ldots + b_r'e_r'' + d_1'y_{i1}^0 + \ldots + d_s'y_{is}^0,$$

где $e_j' \in M_i^+$, $e_k'' \in M_i^-$, $y_{il}^0 \in M_i^0$, $j = 1, \ldots, t, k = 1, \ldots, r, l = 1, \ldots, s$, $k' \in \mathcal{N}$. Если все числа b_k' равны нулю, то все доказано. Предположим, что среди указанных чисел имеются строго положительные. Тогда среди чисел a_i' должны быть числа тоже строго положительные, поскольку в противном случае

$$L_{i+1}(k'x) = k'L_{i+1}(x) = b_1'L_{i+1}(e_1'') + \ldots + b_r'L_{i+1}(e_r'') + d_1'L_{i+1}(y_{i1}^0) + \ldots + d_s'L_{i+1}(y_{is}^0) < 0,$$

что противоречит тому, что x – решение системы S_{i+1}.

Удаляя векторы $e_k'' \in M_i^-$ из представления вектора $k'x$ путем применения леммы 9, приводим вектор $k'x$ к виду (37). ■

Отсюда вытекает такой критерий проверки совместности СЛОДН над множеством натуральных чисел $\mathcal{N}$.

Теорема 33. *СЛОДН $S' = L_1(x) \geq 0 \wedge L_2(x) \geq 0 \wedge \ldots \wedge L_p(x) \geq 0$ совместна тогда и только тогда, когда ее TSS $M_p' \neq \emptyset$.*

Пример 24. Пусть дана система

$$S = \left\{ \begin{array}{lllllllllll} L_1(x) & = & -3x_1 & - & 4x_2 & + & 5x_3 & - & 6x_4 & \geq & 0, \\ L_2(x) & = & 2x_1 & + & 3x_2 & - & 3x_3 & + & x_4 & \geq & 0. \end{array} \right.$$

Для функции $L_1(x)$ имеем такие множества

$$M_1^0 = \emptyset,\ M_1^- = \{(1,0,0,0), (0,1,0,0), (0,0,0,1)\},$$
$$M_1^+ = \{(0,0,1,0)\}.$$

Значения функции $L_1(x)$ на этих векторах таковы: -3, -4, -6, 5. Следовательно, TSS M_1' для $L_1(x) \geq 0$ имеет вид:

$$M_1' = \{(0,0,1,0), (5,0,3,0), (0,5,4,0), (0,0,6,5)\}.$$

Значения на полученных векторах для $L_2(x)$ таковы: -3, 1, 3, -13. Следовательно, TSS состоит из векторов:

$$M_2' = \{(3,0,2,0), (5,0,3,0), (0,5,4,0), (0,1,1,0),$$
$$(13,0,9,1), (0,13,14,3)\}.$$

Поскольку полученное множество непусто, то система S совместна. ♠

8.7. Проверка совместности без введения дополнительных неизвестных

Анализ процесса построения TSS для СЛОДН показывает, что при построении базиса множества всех решений СЛОДН путем сведения к СЛОДУ можно избежать явного

введения дополнительных неизвестных. Это возможно благодаря специфике СЛОДУ, которая соответствует данной СЛОДН. Объясним идею этого построения с помощью примера.

Пример 25. Построить TSS для СЛОДН

$$S = \left\{ \begin{array}{rcl} -2x_1 + 1x_2 - 1x_3 + 1x_4 + 0x_5 & \geq & 0, \\ 1x_1 - 2x_2 - 3x_3 + 2x_4 + 1x_5 & \geq & 0. \end{array} \right.$$

Рассмотрим СЛОДУ S', которая соответствует данной СЛОДН:

$$S = \left\{ \begin{array}{rcl} -2x_1 + 1x_2 - 1x_3 + 1x_4 + 0x_5 - 1x_6 + 0x_7 & = & 0, \\ 1x_1 - 2x_2 - 3x_3 + 2x_4 + 1x_5 + 0x_6 - 1x_7 & = & 0, \end{array} \right.$$

где последние два столбика соответствуют дополнительным неизвестным x_6 и x_7.

TSS для первого уравнения СЛОДУ S' состоит из таких векторов-решений:

$$s_1 = (1,2,0,0,0,0,0),\ s_2 = (1,0,0,2,0,0,0),\ s_3 = (0,1,1,0,0,0,0),$$
$$s_4 = (0,0,1,1,0,0,0), s_5 = (0,0,0,0,1,0,0),\ s_6 = (0,1,0,0,0,1,0),$$
$$s_7 = (0,0,0,1,0,1,0),\ s_8 = (0,0,0,0,0,0,1).$$

При подстановке этих векторов во второе уравнение СЛОДУ S' получаем такие значения: $L_2(s_1) = -3, L_2(s_2) = 5, L_2(s_3) = -5,\ L_2(s_4) = -1, L_2(s_5) = 1, L_2(s_6) = -2, L_2(s_7) = = 2, L_2(s_8) = -1$. Комбинируя эти значения в соответствии с TSS-алгоритмом, находим такие векторы:

$$m_1 = (0,1,0,1,0|2,0),\ m_2 = (0,0,0,1,0|1,2),\ m_3 = (0,0,0,0,1|0,1),$$
$$m_4 = (1,0,0,2,0|0,5),\ m_5 = (0,1,0,0,2|1,0),\ m_6 = (2,5,0,4,0|5,0),$$
$$m_7 = (0,0,1,1,1|0,0),\ m_8 = (0,0,2,3,0|1,0),\ m_9 = (1,0,5,7,0|0,0),$$
$$m_{10} = (1,1,1,2,0|0,0),\ m_{11} = (0,1,1,0,5|0,0),\ m_{12} = (0,2,2,5,0|5,0),$$
$$m_{13} = (4,5,0,3,0|0,0),\ m_{14} = (1,2,0,0,3|0,0),\ m_{15} = (2,4,0,3,0|3,0).$$

В процессе чистки удаляем вектор m_6, так как $m_6 > m_1$, и векторы m_{12} и m_{15} по той же причине. Вектор m_{10} удаляется, поскольку $5m_{10} = m_9 + m_{13}$. Следовательно, TSS СЛОДУ S' включает векторы:

$$m_1 = (0,1,0,1,0|2,0),\ m_2 = (0,0,0,1,0|1,2),\ m_3 = (0,0,0,0,1|0,1),$$
$$m_4 = (1,0,0,2,0|0,5),\ m_5 = (0,1,0,0,2|1,0),\ m_7 = (0,0,1,1,1|0,0),$$
$$m_8 = (0,0,2,3,0|1,0),\ m_9 = (1,0,5,7,0|0,0),\ m_{11} = (0,1,1,0,5|0,0),$$
$$m_{13} = (4,5,0,3,0|0,0),\ m_{14} = (1,2,0,0,3|0,0).$$

Обратим внимание на последние две координаты полученных векторов (эти координаты отделены вертикальной риской). Нетрудно понять, что эти координаты равны $L_1(m'_i)$ и $L_2(m'_i)$, где m'_i – вектор, который включает первые q координат вектора $m_i,\ i = 1,2,\ldots,15$. Это обстоятельство дает возможность не вводить явно дополнительные неизвестные, а в процессе чистки при необходимости вычислять координати, которые требуюся для сравнения векторов.

Если вернуться к СЛОДН S, то TSS для $L_1(x) \geq 0$ имеет вид:

$$s'_1 = (1,2,0,0,0),\ s'_2 = (1,0,0,2,0),\ s'_3 = (0,1,1,0,0),\ s'_4 = (0,0,1,1,0),$$
$$s'_5 = (0,0,0,0,1),\ s'_6 = (0,1,0,0,0),\ s'_7 = (0,0,0,1,0).$$

При подстановке этих векторов в $L_2(x) \geq 0$, получаем такие значения: -3, 5, -5, -1, 1, -2, 2. Комбинируя векторы $s'_1 - s'_7$ в соответствии с TSS-алгоритмом и добавляя векторы s'_2, s'_5 и s'_7, получаем такие векторы:

$$m'_1 = (0,1,0,1,0),\ m'_2 = (0,0,0,1,0),\ m'_3 = (0,0,0,0,1),$$
$$m'_4 = (1,0,0,2,0),\ m'_5 = (0,1,0,0,2),\ m'_6 = (2,5,0,4,0),$$
$$m'_7 = (0,0,1,1,1),\ m'_8 = (0,0,2,3,0),\ m'_9 = (1,0,5,7,0),$$
$$m'_{10} = (1,1,1,2,0),\ m'_{11} = (0,1,1,0,5),\ m'_{12} = (0,2,2,5,0),$$
$$m'_{13} = (4,5,0,3,0),\ m'_{14} = (1,2,0,0,3),\ m'_{15} = (2,4,0,3,0).$$

В процессе чистки находим, что $m'_6 > m'_1$. Для того, чтобы принять решение об удалении m'_6 из TSS, вычисляем $L_1(m'_6) = 5,\ L_2(m'_6) = 0$ и $L_2(m'_1) = 2,\ L_2(m'_1) = 0$. Так как вектор $(5,\ 0\ ) > (2,\ 0)$, то m'_6 удаляется из TSS. Аналогично удаляются и векторы m'_{12} и m'_{15}. ♠

8.8. Краткая характеристика методов

Рассмотренные методы и алгоритмы не сравниваются между собой и каждый из них следует рассматривать как альтернативный метод решения одной и той же задачи. Действительно, условия (C_p) и (PC_p) в методах Контежан–Деви и Потье не сравниваются, поскольку каждое из них не вытекает из другого. Алгоритм Доменджуда принципиально отличается от алгоритмов Контежан–Деви, Потье и TSS тем, что эти алгоритмы, как и алгоритм Гаусса, плохо распараллеливаются, а алгоритм Доменджуда распараллеливается.

Алгоритм Гаусса, кроме того, что он по своей сути последовательный, имеет и другие недостатки:

– он использует алгоритм решения одного диофантового уравнения (например, алгоритм Фортенбахера) столько раз, сколько уравнений в системе;

– число переменных в новой системе пропорционально мощности базиса решений предыдущего уравнения, которое в худшем случае пропорционально експоненте от значений коэффициентов этого уравнения;

– значения коэффициентов в новой системе быстро возрастают;

– полученное множество решений включает не только минимальные решения, и поэтому необходимо проводить дополнительную работу с целью построения множества минимальных решений.

Алгоритм Контежан–Деви, хотя и не имеет большинства недостатков, присущих алгоритму Гаусса, но стает малоэффективным, когда коэффициенты в уравнениях имеют большие нормы, хотя на практике этот алгоритм зарекомендовал себя с наилучшей стороны [51].

Практика показала, что алгоритм Доменджуда не наилучший, поскольку на многих тестовых примерах он работал не лучше других алгоритмов [47]. Основным недостатком алгоритма Доменджуда является невозможность его инкрементального использования, в то время как алгоритмы Контежан–Деви, Потье и TSS допускают такое использование. Больше того, данные алгоритмы могуь контроллировать и ограничивать генерацию компонент с помощью самих же компонент.

Следовательно, эти алгоритмы применимы к решению не только СЛОДУ, но и к решению СЛНДУ и решению систем линейных неравенств.

Подход к определению совместности СЛОДУ с помощью построения TSS сравнивается только с алгоритмом Доменджуда и эксперименты показывают, что TSS и минимальное порождающее множество совпадают. Однако TSS-алгоритм обладает свойством инкрементальности, а алгоритм Доменджуда этим свойством не обладает. Если в алгоритме Доменджуда заменить алгоритм, реализующий первый шаг, TSS-алгоритмом, то можно получить новый алгоритм, который уже будет иметь свойство инкрементальности. Кроме этого, алгоритм построения TSS

– не требует никаких предварительных преобразований или проверок системы;

– может определять (и в случае необходимости удалять) линейно зависимые уравнения в системе;

– может вычислять ранг матрицы системы;

– применим к системам линейных однородных и неоднородных неравенств.

Эксперименты показали, что TSS-алгоритм достаточно хорошо работает с системами, у которых разреженные матрицы. Если матрица системы не разрежена и число уравнений в системе больше половины числа ее неизвестных, то целесообразно матрицу такой системы предварительно диагонализировать.

9. Язык линейных уравнений над кольцом $\mathcal{Z}$

Рассмотрим модификацию TSS-алгоритма для решения систем линейных диофантовых уравнений над кольцом целых чисел $\mathcal{Z}$ [11].

9.1. TSS-метод решения ЛОДУ

Пусть дано ЛОДУ

$$L(x) = a_1x_1 + \ldots + a_ix_i + \ldots + a_qx_q = 0, \tag{38}$$

где $a_i, x_i \in \mathcal{Z},\ i = 1, \ldots, q$.

Не ограничивая общности, предположим, что в функции $L(x)$ первым ненулевым коэффициентом будет a_1 и $a_1 > 0$. Построим множество векторов

$$B = \{e_1 = (-a_2, a_1, 0, \ldots, 0), e_2 = (-a_3, 0, a_1, 0, \ldots, 0), e_{q-1} = (-a_q, 0, 0, \ldots, 0, a_1)\} \cup M_0,$$

где $M_0 = \{e_r : L(e_r) = 0\}$, причем, если для некоторого $a_i \neq 0$ НОД$(a_i, a_1) \neq 1$, то сократим координаты такого вектора на этот НОД. Выбранный ненулевой коэффициент a_1 будем называть **основным**. Таким образом, можно считать, что все векторы в множестве B таковы, что a_i и a_1 взаимно просты. Иными словами, множество B, как и раньше, строится путем комбинирования первого ненулевого коэффициента с остальными ненулевыми коэффициентами, взятыми с противоположными знаками, и пополнения векторами канонического базиса, которые соответствуют нулевым коэффициентам ЛОДУ (38). Построенное таким образом TSS будем называть **предбазисом**. Очевидно, что векторы из множества B являются решениями ЛОДУ (38), а само множество B – замкнуто относительно сложения, вычитания и умножения на элемент из кольца $\mathcal{Z}$. Пусть M – множество всех решений ЛОДУ (38).

Лемма 10. *Пусть $x = (c_1, c_2, \ldots, c_q)$ – некоторое решение ЛОДУ (38), тогда если $x \notin B$, то x представляется в виде линейной комбинации вида*

$$a_1x = c_2e_1 + c_3e_2 + \ldots + c_qe_{q-1},$$

где $e_i \in B, i = 1, \ldots, q-1$.

Доказательство. Если $x = (c_1, \ldots, c_q) \in M$, то вектор

$$c_2e_1 + c_3e_2 + \ldots + c_qe_{q-1} = (-c_2a_2 - c_3a_3 - \ldots - c_qa_q, c_2a_1, \ldots, c_qa_1) = \\ = (c_1a_1, c_2a_1, \ldots, c_qa_1) = a_1(c_1, c_2, \ldots, c_q) = a_1x,$$

где $a_1c_1 = -a_2c_2 - a_3c_3 - \ldots - a_qc_q$ в силу того, что x – решение ЛОДУ (38).

Заметим, что если некоторый вектор e_j из B является вектором канонического базиса и j-я координата вектора x равна c_j, то в представлении вектора x вектор e_j входит с коэффициентом a_1c_j. ■.

Из доказанной леммы вытекает такое полезное следствие.

Следствие 6. *Если среди коэффициентов ЛОДУ имеется хотя бы один коэффициент равный 1, то множество B является базисом множества всех решений ЛОДУ.*

Действительно, тогда элементы множества B имеют вид

$$\{e_1 = (-a_2, 1, 0, \ldots, 0), e_2 = (-a_3, 0, 1, 0, \ldots, 0), e_{q-1} = (-a_q, 0, 0, \ldots, 0, 1)\} \cup M_0,$$

т.е. в разложении произвольного решения x по векторам из множества B основной коэффициент согласно лемме 10 будет равным единице. А это и означает, что множество B – базис. ■

Пример 26. Построить TSS ЛОДУ

$$L(x) \quad = \quad 3x_1 + y - z + 2u + v \quad = \quad 0.$$

Предбазис или TSS этого ЛОДУ имеет вид

$$e_1 = (-1, 3, 0, 0, 0), e_2 = (1, 0, 3, 0, 0), e_3 = (-2, 0, 0, 3, 0), e_4 = (-1, 0, 0, 0, 3).$$

Решения ЛОДУ $x_1 = (0, 2, 3, 0, 1), x_2 = (1, 1, 0, -2, 0)$ имеют представления $3x_1 = 2e_1 + 3e_2 + e_4,\ 3x_2 = e_1 - 2e_3$. Базисом множества всех решений для этого ЛОДУ будет множество

$$B = \{e_1 = (1, -3, 0, 0, 0), e_2 = (0, 1, 1, 0, 0), e_3 = (0, -2, 0, 1, 0), e_4 = (0, -1, 0, 0, 1)\}.$$

В этом базисе векторы x_1 и x_2 имеют представление: $x_1 = 3e_2 + e_4,\ x_2 = e_1 - 2e_3$. ♠

9.2. TSS-метод решения СЛОДУ

Рассмотрим СЛОДУ

$$S = \left\{ \begin{array}{rcrcrcrcr} L_1(x) & = & a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0, \\ L_2(x) & = & a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & 0, \end{array} \right. \tag{39}$$

где $a_{ij}, x_i \in \mathcal{Z},\ i = 1, \ldots, p,\ j = 1, \ldots, q$.

Построим предбазис $B_1 = \{e_{11}, e_{12}, \ldots, e_{1q-1}\}$ для первого уравнения $L_1(x) = 0$ и вычислим значения $L_2(e_{1i}) = b_i$, где $e_{1i} \in B_1, b_i \in \mathcal{Z}$. Составим уравнение

$$b_1y_1 + \ldots + b_iy_i + \ldots + b_{q-1}y_{q-1} = 0, \tag{40}$$

и построим для него предбазис $B_1' = \{s_1, \ldots, s_{q-2}\}$. Векторам s_i из B_i' соответствуют векторы-решения $B_2 = \{e_{21}, \ldots, e_{2q-2}\}$ СЛОДУ $L_1(x) = 0 \wedge L_2(x) = 0$.

Лемма 11. *Множество векторов B_2 составляет предбазис СЛОДУ $L_1(x) = 0 \wedge L_2(x) = 0$, т. е. любое решение x этой СЛОДУ имеет представление $kx = l_1e_{21} + \ldots + l_{q-2}e_{2q-2}$, где $e_{2i} \in B_2, l_i \in \mathcal{Z},\ i = 1, \ldots, q-2$.*

Доказательство. Пусть x – произвольное решение СЛОДУ $L_1(x) = 0 \wedge L_2(x) = 0$. Поскольку x – решение $L_1(x) = 0$, то в силу леммы 10 x можно представить в виде

$$dx = a_1e_{11} + \ldots + a_{q-1}e_{1q-1},$$

где $e_{1i} \in B_1, a_i \in \mathcal{Z},\ i = 1, \ldots, q-1$. Тогда, в силу того, что x – решение $L_2(x) = 0$, получаем

$$L_2(dx) = a_1b_1 + \ldots + a_{q-1}b_{q-1} = 0,$$

где $b_j = L_2(e_{1j}),\ j = 1, \ldots, q-1$. Следовательно, вектор $a = (a_1, \ldots, a_{q-1})$ – решение ЛОДУ (40) и в силу леммы 10 получаем

$$ka = d_1 s_1 + \ldots + d_{q-2} s_{q-2},$$

где $s_i \in B'_1, d_i \in \mathcal{Z},\ i = 1, \ldots, q-2$, а k – основной коэффициент этого ЛОДУ. Отсюда следует, что

$$kdx = d_1 e_{21} + \ldots + d_{q-2} e_{2q-2},$$

где $e_{2i} \in B_2, i = 1, \ldots, q-2$. ■.

С помощью математической индукции непосредственно из лемм 10 и 11 следует справедливость такой теоремы.

Теорема 34. *TSS СЛОДУ (39) B, построенное вышеописанным способом, является предбазисом множества всех решений этой СЛОДУ.*

Пример 27. Найдем предбазис для СЛОДУ

$$S = \left\{ \begin{array}{lcccl} L_1(x) & = & 3x_1 + x_2 - x_3 + 2x_4 + x_5 & = & 0, \\ L_2(x) & = & 2x_1 + 3x_2 + 0x_3 - x_4 + 2x_5 & = & 0. \end{array} \right.$$

Предбазис для первого уравнения был построен выше в примере 28:

$$B_1 = \{e_{11} = (1, -3, 0, 0, 0), e_{12} = (0, 1, 1, 0, 0), e_{13} = (0, -2, 0, 1, 0), e_{14} = (0, -1, 0, 0, 1)\}.$$

Значения $L_2(x)$ на этих векторах равны соответственно -7, 3, -7, -1. Составляем уравнение $-7y_1 + 3y_2 - 7y_3 - y_4 = 0$ и строим предбазис множества решений этого ЛОДУ:

$$B'_1 = \{s_1 = (3, 7, 0, 0), s_2 = (-1, 0, 1, 0), s_3 = (-1, 0, 0, 7)\}.$$

Этим векторам соответствует TSS (предбазис)

$$B_2 = \{e_{21} = (3, -2, 7, 0, 0), e_{22} = (-1, 1, 0, 1, 0), e_{23} = (-1, -4, 0, 0, 7)\}.$$

Если при построении предбазиса уравнения $-7y_1 + 3y_2 - 7y_3 - y_4 = 0$ комбинирование вести по последнему значению (т. е. по -1), то получаем такой базис множества всех решений данной СЛОДУ

$$\{e_{21} = (1, 4, 0, 0, -7), e_{22} = (0, -2, 1, 0, 3), e_{23} = (0, 5, 0, 1, -7)\}.\ \spadesuit$$

Принимая во внимание следствие 6, результат теоремы 34 можно усилить путем введения избыточности в предбазис. Это можно сделать на основании того, что значения коэффициентов в уравнении $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = 0$ взаимно просты. Благодаря этому всегда можно добиться того, чтобы среди значений $L_1(x)$ была единица. Не ограничивая общности, будем предполагать, что НОД$(a_{11}, a_{12}, a_{13}) = 1$, т. е. первые три коэффициенты взаимно просты в $L_1(x)$. Тогда существуют числа d_1, d_2, d_3 такие, что на векторе $y = (d_1, d_2, d_3, 0, \ldots, 0)$ значение $L_1(y) = 1$. Добившись этого, вычислим значения $L_1(x)$ на векторах канонического базиса. Построим предбазис B_1, комбинируя вектор y с остальными векторами для получения предбазиса. Заметим, что векторы из B_1 имеют вид

$$e'_1 = -a_{11}y + e_1, e'_2 = -a_{12}y + e_2, e'_3 = -a_{13}y + e_3, e'_4 = -a_{14}y + e_4, \ldots, e'_n = -a_{1n}y + e_n, \quad (41)$$

где e_i – векторы канонического базиса, a_{ij} – коэффициенты в уравнении $L_1(x) = 0$. В координатной форме векторы e'_i выглядят следующим образом:

$$\begin{array}{c} e'_1 = (-a_{11}d_1 + 1, -a_{11}d_2, -a_{11}d_3, 0, \ldots, 0), \\ e'_2 = (-a_{12}d_1, -a_{12}d_2 + 1, -a_{12}d_3, 0, \ldots, 0), \\ e'_3 = (-a_{13}d_1, -a_{13}d_2, -a_{13}d_3 + 1, 0, \ldots, 0), \\ e'_4 = (-a_{14}d_1, -a_{14}d_2, -a_{14}d_3, 1, \ldots, 0), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ e'_n = (-a_{1n}d_1, -a_{1n}d_2, -a_{1n}d_3, 0, \ldots, 1). \end{array}$$

Имеет место

Теорема 35. *TSS ЛОДУ (38) B_1, построенное вышеописанным способом, является базисом множества всех решений этого ЛОДУ $L_1(x) = 0$.*

Сложность построения базиса пропорциональна величине l^3, где l – максимальное из чисел m и q, где q – количество неизвестных в ЛОДУ, а m – максимальная длина двоичного представления коэффициентов ЛОДУ.

Доказательство. Пусть $x = (c_1, c_2, \ldots, c_q)$ – решение ЛОДУ $L_1(x) = 0$. Тогда вектор x имеет представление

$$\begin{gathered} x = c_1e'_1 + c_2e'_2 + c_3e'_3 + c_4e'_4 + \ldots + c_qe'_q = \\ = ([-c_1a_{11}d_1 + c_1 - c_2a_{12}d_1 - c_3a_{13}d_1 - c_4a_{14}d_1 - \ldots - c_qa_{1q}d_1], \\ [-c_1a_{11}d_2 - c_2a_{12}d_2 + c_2 - c_3a_{13}d_2 - c_4a_{14}d_2 - \ldots - c_qa_{1q}d_2], \\ [-c_1a_{11}d_3 - c_2a_{12}d_3 - c_3a_{13}d_3 + c_3 - c_4a_{14}d_3 - \ldots - c_qa_{1q}d_3], c_4, \ldots, c_q) = \\ = (c_1, c_2, c_3, c_4, \ldots, c_q) \end{gathered}$$

в силу того, что $L(x) = a_{11}c_1 + a_{12}c_2 + \ldots + a_{1q}c_q = 0$.

Сложность данного алгоритма определяется сложностью расширенного алгоритма Эвклида, находящего вместе с НОД и линейную комбинацию, представляющую этот НОД. Известно (см. [15]), что эта сложность выражается величиной $O(m \log m)$, где m – длина двоичной записи максимального из коэффициентов ЛОДУ. Этот алгоритм применяется не более q раз и тогда имеем оценку $O(mq \log m)$. Построение базиса B_1 требует не более q^3 операций. Следовательно, общая оценка временной сложности алгоритма выражается величиной $O(l^3)$, где $l = \max(m, q)$. ■.

Из этой теоремы вытекает такое следствие.

Следствие 7. *Временная сложность построения базиса множество всех решений СЛОДУ вида (39) пропорциональна величине $O(pl^3)$, где p – количество уравнений СЛОДУ, а $l = \max(m, p)$.*

Заметим, что первые три вектора e'_1, e'_2, e'_3 в представлении (41) линейно зависимы. Действительно, в силу того, что $a_{11}d_1 + a_{12}d_2 + a_{13}d_3 = 1$, то

$$d_1e'_1 + d_2e'_2 + d_3e'_3 = 0.$$

Действительно, используя координатное представление векторов, имеем

$$\begin{gathered} d_1e'_1 + d_2e'_2 + d_3e'_3 = (a_{12}d_1d_2 + a_{13}d_1d_3, -a_{11}d_1d_2, -a_{11}d_3, 0, \ldots, 0) + \\ +(-a_{12}d_1d_2, a_{11}d_1d_2 + a_{13}d_2d_3, -a12d_2d_3, 0 \ldots, 0) + \\ +(-a_{13}d_1d_3, -a_{13}d_2d_3, a_{11}d_1d_3 + a_{12}d_2d_3, 0 \ldots, 0) = 0. \end{gathered}$$

Таким образом один из векторов e'_1, e'_2, e'_3 можно удалить из полученного базиса решений.

9.3. TSS-метод решения СЛНДУ

Пусть S – СЛНДУ вида (39) и $b_p \neq 0$. Выполняя элиминацию свободных членов в первых $p-1$ уравнениях преобразуем исходную СЛНДУ к виду

$$S' = \left\{ \begin{array}{rcccccccc} L'_1(x) & = & a'_{11}x_1 & + & \ldots & + & a'_{1q}x_q & = & 0, \\ L'_2(x) & = & a'_{21}x_1 & + & \ldots & + & a'_{2q}x_q & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ L'_{p-1}(x) & = & a'_{p-11}x_1 & + & \ldots & + & a'_{p-1q}x_q & = & 0, \\ L_p(x) & = & a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & b_p. \end{array} \right. \quad (42)$$

Построим базис множества решений СЛОДУ, состоящей из $p-1$ первых уравнений системы (42). Пусть это будут векторы $\{s_1,\ldots,s_k\}$. Найдем значения $L_p(s_j)=a_j, j=1,\ldots,k$. Для этих значений верна

Теорема 36. *СЛНДУ вида (42) (а вместе с ней и СЛНДУ (19)) совместна тогда и только тогда, когда ЛНДУ $a_1y_1+a_2y_2+\ldots+a_ky_k=b_p$ имеет хотя бы одно решение в множестве целых чисел.*

Доказательство. Если уравнение $a_1y_1+a_2y_2+\ldots+a_ky_k=b_p$ имеет решение $(c_1,c_2,\ldots,c_k)$, то очевидно, что вектор $s=c_1s_1+c_2s_2+\ldots+c_ks_k$ – решение СЛНДУ.

Если СЛНДУ совместна и $s=(k_1,k_2,\ldots,k_n)$ ее решение, то представим s в виде линейной комбинации базисных векторов подсистемы, состоящей из первых $p-1$ однородных уравнений системы (42), т. е.

$$s=c_1s_1+c_2s_2+\ldots+c_ks_k.$$

Тогда $L_p(s)=c_1a_1+c_2a_2+\ldots+c_ks_k=b_q$ должно иметь ходя бы одно решение, поскольку s решение СЛНДУ. ■.

Известно, что общее решение СЛНДУ имеет вид $y=x+\sum\limits_{i=1}^{k}a_ix_i$, где x – частное решение СЛНДУ, x_i – базисные решения приведенной СЛОДУ, a_i – произвольные целые числа, а k – количество базисных решений. Таким образом, для полного решения СЛНДУ необходимо построить базис множества решений ее СЛОДУ и найти одно из решений СЛНДУ. Поиск такого решения, как следует из вышеизложенного, сводится к поиску решения уравнения $a_1y_1+a_2y_2+\ldots+a_ky_k=b_p$. Это решение можно найти, например, методом наименьшего коэффициента.

Пример 28. Проверить на совместность СЛНДУ

$$S=\left\{\begin{array}{rcrcrcrcrcrcr} L_1(x) & = & 2x_1 & - & 3x_2 & + & x_3 & + & x_4 & + & 0x_5 & = & 1,\\ L_2(x) & = & 3x_1 & + & x_2 & + & x_3 & + & 0x_4 & - & x_5 & = & -2. \end{array}\right.$$

Преобразованная СЛНДУ имеет вид

$$S'=\left\{\begin{array}{rcrcrcrcrcrcr} L_1'(x) & = & 7x_1 & - & 5x_2 & + & 3x_3 & + & 2x_4 & - & x_5 & = & 0,\\ L_2(x) & = & 3x_1 & + & x_2 & + & x_3 & + & 0x_4 & - & x_5 & = & -2. \end{array}\right.$$

Базис ЛОДУ $L_1(x)'=0$ составляют векторы (здесь не нужно вычислять НОД коэффициентов, поскольку имеется коэффициент равный 1):

$$(1,0,0,0,7),(0,1,0,0,-5),(0,0,1,0,3),(0,0,0,1,2).$$

Значения $L_2(x)$ на этих векторах равны -4, 6, -2, -2. Наибольший общий делитель этих значений равен 2 и является делителем свободного члена $b_2=-2$. Следовательно, СЛНДУ имеет решение, т. е. совместна. Частным решением СЛНДУ является вектор $x^1=(0,0,0,1,2)$, а решениями СЛОДУ, которая соответствует данной СЛНДУ, являются векторы $x_1=(1,0,0,-2,3), x_2=(0,0,1,-1,1), x-3=(0,1,0,3,1)$.

Если дана система

$$S'=\left\{\begin{array}{rcrcrcrcrcrcr} L_1'(x) & = & 7x_1 & - & 5x_2 & + & 3x_3 & + & 2x_4 & - & x_5 & = & 0,\\ L_2(x) & = & 3x_1 & + & x_2 & + & x_3 & + & 0x_4 & - & x_5 & = & -3, \end{array}\right.$$

то она решений не имеет в кольце целых чисел, поскольку НОД(-4, 6, -2, -2) = 2 не делит свободный член -3 и поэтому уравнение $-4x+6y-2z-2u=-3$ не имеет решений.

10 Язык линейных уравнений над множеством {0,1}

Рассмотрим класс СЛОДУ вида

$$S = \left\{ \begin{array}{ccccccccccccccc} 1x_1 & + & 0x_2 & \ldots & + & 0x_q & 1x_{q+1} & 0x_{q+2} & \ldots & 0x_{2q} & -1x_{2q+1} & = & 0, \\ 0x_1 & + & 1x_2 & \ldots & + & 0x_q & 0x_{q+1} & 1x_{q+2} & \ldots & 0x_{2q} & -1x_{2q+1} & = & 0, \\ \ldots & & \ldots & \ldots & & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0x_1 & + & 0x_2 & \ldots & + & 1x_q & 0x_{q+1} & 0x_{q+2} & \ldots & 1x_{2q} & -1x_{2q+1} & = & 0. \end{array} \right. \quad (43)$$

Строение такой СЛОДУ очевидно: к двум единичным матрицам размерности $q \times q$, записанных одна за другой, добавляется вектор-столбик, состоящий из -1 (см. раздел 7.3).

Непосредственное применение TSS-алгоритма показывает, что сгенерированное им множество решений совпадает с базисом множества всех решений данной СЛОДУ и состоит из 2^q векторов. Эти векторы имеют следующие интересные для нас свойства :

a) первые q координат и вторые q координат дополняют друг друга, т. е. если i-я координата первой части равна 1 (0), то $q+i$-я координата второй части равна 0 (1), а последняя координата такого решения всегда равна 1;

b) величины первых q компонент возрастают от 0 до 2^q-1, т. е. включают все возможные комбинации 0 и 1.

Для примера, рассмотрим систему

$$S = \left\{ \begin{array}{ccccccccccccccccc} 1x_1 & + & 0x_2 & + & 0x_3 & + & 0x_4 & + & 1x_5 & + & 0x_6 & + & 0x_7 & + & 0x_8 & -1x_9 & = & 0, \\ 0x_1 & + & 1x_2 & + & 0x_3 & + & 0x_4 & + & 0x_5 & + & 1x_6 & + & 0x_7 & + & 0x_8 & -1x_9 & = & 0, \\ 0x_1 & + & 0x_2 & + & 1x_3 & + & 0x_4 & + & 0x_5 & + & 0x_6 & + & 1x_7 & + & 0x_8 & -1x_9 & = & 0, \\ 0x_1 & + & 0x_2 & + & 0x_3 & + & 1x_4 & + & 0x_5 & + & 0x_6 & + & 0x_7 & + & 1x_8 & -1x_9 & = & 0. \end{array} \right.$$

TSS данной системы совпадает с базисом множества всех решений и состоит из таких 16 векторов, группы координат которых отделены пробелами:

$$x_1 = (1,1,1,1,\ 0,0,0,0,\ 1),\ x_2 = (0,1,1,1,\ 1,0,0,0,\ 1), x_3 = (1,0,1,1,\ 0,1,0,0,\ 1), x_4 = (0,0,1,1,\ 1,1,0,0,\ 1),$$
$$x_5 = (1,1,0,1,\ 0,0,1,0,\ 1), x_6 = (0,1,0,1,\ 1,0,1,0,\ 1), x_7 = (1,0,0,1,\ 0,1,1,0,\ 1),\ x_8 = (0,0,0,1,\ 1,1,1,0,\ 1),$$
$$x_9 = (1,1,1,0,\ 0,0,0,1,\ 1), x_{10} = (0,1,1,0,\ 1,0,0,1,\ 1), x_{11} = (1,0,1,0,\ 0,1,0,1,\ 1), x_{12} = (0,0,1,0,\ 1,1,0,1,\ 1),$$
$$x_{13} = (1,1,0,0,\ 0,0,1,1,\ 1), x_{14} = (0,1,0,0,\ 1,0,1,1,\ 1), x_{15} = (1,0,0,0,\ 0,1,1,1,\ 1), x_{16} = (0,0,0,0,\ 1,1,1,1,\ 1).$$

Отмеченные свойства множества решений для такого вида СЛОДУ позволяют построить достаточно экономный по памяти алгоритм построения базиса множества всех решений для общего вида СЛОДУ над множеством $\{0,1\}$.

Идея построения этого алгоритма вытекает из свойства инкрементальности TSS и свойства a), приведенного выше. Она состоит в реализации полного перебора вариантов и не требует введения дополнительных переменных.

Рассмотрим систему уравнений следующего вида

$$S'' = \left\{ \begin{array}{cccccccccc} S, & & & & & & & & & \\ 1 & 0 & \ldots & 0 & 1 & \ldots & 0 & -1 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ 0 & 0 & \ldots & 1 & 0 & \ldots & 1 & -1 & = & 0. \end{array} \right.$$

Из свойства инкрементальности алгоритма построения TSS следует, что TSS данной системы совпадает с TSS системы

$$S_1 = \left\{ \begin{array}{cccccccccc} 1 & 0 & \ldots & 0 & 1 & \ldots & 0 & -1 & = & 0, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & \\ 0 & 0 & \ldots & 1 & 0 & \ldots & 1 & -1 & = & 0, \\ S. & & & & & & & & & \end{array} \right.$$

Отсюда следует, что вычислять TSS для подсистемы, состоящей из уравнений предшествующих системе S (назовем ее верхней), нет необходимости, т. к. это множество известно. С другой стороны, поскольку уравнения подсистемы S дополняются нулями, то только первые q координат TSS верхней подсистемы влияют на окончательное TSS всей системы в целом. Поскольку закон строения этих координат известен из свойства a), отмеченного выше, то при вычислении TSS нет необходимости вводить дополнительные переменные и запоминать все TSS векторы верхней подсистемы. TSS всей системы можно вычислять перебирая последовательно все варианты векторов TSS верхней подсистемы, запоминая только базисные решения. Причем при этом переборе можно использовать различные порядки.

Основываясь на этих рассуждениях и используя естественный порядок булевских векторов, приведем структуры данных алгоритма и сам алгоритм в этих структурах.

Пусть x – q-мерный булевский вектор, в котором хранится текущее значение вектора-решения СЛОДУ S, B означает базис множества всех решений S в множестве $\{0,1\}$, $b(j)$ – вектора-столбцы матрицы A системы S. При этих структурах данных алгоритм принимает вид:

$$TSS-01(b_1,b_2,\ldots,b_q,p)$$

Вход. $b_1,b_2,\ldots,b_q$ – столбцы матрицы A системы $A\cdot x=0$, а p – их размерность.
Выход. B – базис множества всех решений системы $A\cdot x=0$ в множестве $\{0,1\}$.
Метод.

begin
$B=\emptyset$; $x=0$;
for $i=1$ *to* 2^{q-1} *do*
$x=x+1$; $y=0$;
for $j=1$ *to* q *do*
if $x(j)=1$ *then* $y=y+b(j)$
od
if $y=0$ *then add* x *to* B
od
end

Характеристику временной сложности данного алгоритма дает следующее утверждение.

Теорема 37 *Временная сложность алгоритма TSS-01 пропорциональна величине* $pq\cdot 2^{q-1}$, *а память, используемая алгоритмом, пропорциональна величине* $\max(pq,|B|q)$.

Доказательство непосредственно следует из вида алгоритма $TSS-01$.

Заметим, что алгоритм применим к любой числовой области, так как он совершенно не зависит от того, какому числовому множеству принадлежат коэффициенты системы. Кроме того, алгоритм применим без каких либо модификаций к неоднородным системам.

Свойство инкрементальности TSS-алгоритма позволяет построить алгоритм решения СЛОДУ в множестве $\{0,1\}$ и доказать что он строит базис всего множества решений. Однако, эксперименты показали, что время генерации базиса для системы S'' намного лучше, чем время генерации базиса для системы, в которой верхняя и нижняя подсистемы поменялись местами. Сказанное подтверждается таким примером СЛОДУ:

$$\left\{\begin{array}{rrrrrrrrrrrrrrrrrcr}
0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
-1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
1 & 0 & -1 & 1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & = & 0.
\end{array}\right.$$

TSS-алгоритм генерирует такое множество векторов-решений системы

$$s_1 = (11101011000101001),\ s_2 = (01111110100000011),\ s_3 = (10002021122202012),$$
$$s_4 = (00012120222101022),\ s_5 = (20211000020112222),\ s_6 = (02210100200121222),$$
$$s_7 = (12200001100222212),\ s_8 = (00000000111111111),\ s_9 = (05300112502554435),$$
$$s_{10} = (30213021031203123),\ s_{11} = (40434120040103244),\ s_{12} = (02003142424413024),$$
$$s_{13} = (03102132302312013)$$

за время 0 милисекунд (ОС Vista, CPU Duo T2080). Если поменять местами подсистемы, т. е. применить TSS-алгоритм к СЛОДУ

$$\left\{\begin{array}{rrrrrrrrrrrrrrrrrcr}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & = & 0,\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & = & 0,\\
0 & 1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
-1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,\\
1 & 0 & -1 & 1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0,
\end{array}\right.$$

то тоже множество решений TSS-алгоритм генерирует за 220 милисекунд.

Дальнейший анализ этой ситуации показал, что можно использовать эту возможность и создать другой алгоритм построения базиса, который требует введения только двух дополнительных переменных (а не p дополнительных переменных). Приведем краткое описание этого алгоритма из работы [10].

Пусть дана СЛОДУ

$$S = \left\{\begin{array}{rcccccc}
a_{11}x_1 & + & \ldots & + & a_{1q}x_q & = & 0,\\
a_{21}x_1 & + & \ldots & + & a_{2q}x_q & = & 0\\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots\\
a_{p1}x_1 & + & \ldots & + & a_{pq}x_q & = & 0,
\end{array}\right. \tag{44}$$

где $a_{ij}, b_j \in \mathcal{Z}$ (кольцо целых чисел), $i = 1, 2, \ldots, p, j = 1, 2, \ldots, q$. Требуется найти базис множества решений данной СЛОДУ в множестве $\{0, 1\}$.

Работа описываемого ниже алгоритма сводится к выполнению следующих шагов.

1) Для заданной СЛОДУ S строим ее TSS с помощью TSS-алгоритма. Пусть в результате этого построения получены вектора-решения

$$(c_{11}, c_{12}, \ldots, c_{1q}),$$
$$(c_{21}, c_{22}, \ldots, c_{2q}),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(c_{k1}, c_{k2}, \ldots, c_{kq}).$$

Если TSS содержит вектора, координаты которых состоят из 0 и 1, то заносим эти вектора в базис и исключаем их из TSS.

2) Добавляем к векторам, полученным на шаге 1, одну нулевую дополнительную координату и два дополнительных вектора вида $(0, 0, \ldots, 0|1)$ и $(0, 0, \ldots, 0|0)$, т. е. получаем такую систему векторов:

$$\begin{array}{c}(c_{11}, c_{12}, \ldots, c_{1q}|0),\\ (c_{21}, c_{22}, \ldots, c_{2q}|0),\\ \ldots\ldots\ldots\ldots\ldots\ldots\\ (c_{k1}, c_{k2}, \ldots, c_{kq}|0),\\ (0, 0, 0, \ldots, 0, 0|1),\\ (0, 0, 0, \ldots, 0, 0|0).\end{array}$$

3) Вычисления базиса на этих векторах осуществляется путем выполнения q шагов следующего содержания:

a) $i := 1$;

b) от i-го столбца вычитаем почленно последний, добавленный вектор-столбец;

c) последнему (добавленному) вектору приписывается значение 1.

В результате, например, выполнения первого шага получается такая система векторов и значений:

$$\begin{array}{l|l}(c_{11}, c_{12}, \ldots, c_{1q}|0) & (c_{11}, c_{12}, \ldots, c_{1q}|0) \mid c_{11}\\ (c_{21}, c_{22}, \ldots, c_{2q}|0) & (c_{21}, c_{22}, \ldots, c_{2q}|0) \mid c_{21}\\ \ldots\ldots\ldots\ldots\ldots\ldots & \Rightarrow \quad \ldots\ldots\ldots\ldots\ldots\ldots \mid \ldots\\ (c_{k1}, c_{k2}, \ldots, c_{kq}|0) & (c_{k1}, c_{k2}, \ldots, c_{kq}|0) \mid c_{k1}\\ (0, 0, 0, \ldots, 0, 0|1) & (0, 0, 0, \ldots, 0, 0|1) \mid \text{-1},\\ (0, 0, 0, \ldots, 0, 0|0) & (0, 0, 0, \ldots, 0, 0|0) \mid 1.\end{array}$$

4) Комбинируем вектора так, как они комбинируются в TSS-алгоритме, используя для этого последний столбик значений (он отделен вертикальными линиями). Если полученное TSS содержит вектора, координаты которых состоят из 0 и 1, то заносим эти вектора в базис в соответствии с правилами включения и исключаем их из TSS.

Чистка текущего TSS выполняется в два этапа (этап а) и этап б)):

а) сравниваются вектора из текущего базиса (если он непуст) и текущего TSS и если вектор из базиса больше вектора из TSS, то он заменяется в базисе меньшим и удаляется из текущего TSS, иначе если вектор из TSS больше вектора из базиса то он удаляется из TSS, а потом

б) сравниваются вектора из TSS между собой. Если один из векторов больше другого, то вычисляются дополнения этих векторов относительно последней координаты обеих векторов и если полученные вектора тоже находятся в том же отношении, что и основные вектора, то больший вектор удаляется из текущего TSS. Следует заметить, что дополнение нет необходимость строить явно. Действительно, если на некотором шаге j, $j \in [2, q]$ значения d и d', стоящие в предпоследнем столбце (они находятся между вертикальными черточками) и относительно которых вычисляются дополнения векторов $x = (a_1, a_2, \ldots, a_q)$ и $y = (b_1, b_2, \ldots, b_q)$, то при $a_i \geq b_i$ для $i = 1, 2, ..., j$ дополнения будут находиться в том же отношении, если $d - a_i \geq d' - b_i$ или $d - d' \geq a_i - b_i$ для $i = 1, 2, .., j$. Следовательно, если

$x \geq y$, то при сравнении можно использовать разности $c = d - d'$ и $c_i = a_i - b_i$. Если $c \geq c_i$ для всех $i \in [1, j]$, то вектор x удаляется из текущего TSS.

В результате выполнения этого шага получаем такие вектора:

$$(c_{11}, c_{12}, \ldots, c_{1q}|c_{11}),$$
$$(c_{21}, c_{22}, \ldots, c_{2q}|c_{21}),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(c_{k1}, c_{k2}, \ldots, c_{kq}|c_{k1}),$$
$$(0, 0, 0, \ldots, 0, 0|1).$$

5) Добавляем к полученным векторам нулевой вектор и вычисляем значения следующим образом:

a) $i := i + 1$;

b) почленно вычитаем от i-го столбца значения координат последнего столбца;

c) последнему вектору приписываем значение 1.

В результате получаем такую систему векторов, например, $i = 2$:

$$\left|\begin{array}{ll} (c_{11}, c_{12}, \ldots, c_{1q}|c_{11}) & c_{12} - c_{11} \\ (c_{21}, c_{22}, \ldots, c_{2q}|c_{21}) & c_{22} - c_{21} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots \\ (c_{k1}, c_{k2}, \ldots, c_{kq}|c_{k1}) & c_{k2} - c_{k1} \\ (0, 0, 0, \ldots, 0, 0|1) & \text{-1}, \\ (0, 0, 0, \ldots, 0, 0|0) & 1. \end{array}\right|$$

6) Если $i < q$, то ($i := i + 1$; Перейти к выполнению шага 4) иначе закончить работу.

Полученное в результате множество решений содержит все базисные вектора-решения в множестве $\{0, 1\}$, если удалить вектора, содержащие координаты большие 1. Обоснование корректности данного алгоритма вытекает из его свойства инкрементальности [48].

Пример работы алгоритма. Рассмотрим СЛОДУ вида

$$S = \left\{\begin{array}{rcrcrcrcrcrcrcl} -1x_1 & + & 0x_2 & + & 0x_3 & + & 0x_4 & + & 0x_5 & + & 1x_6 & + & 0x_7 & = & 0, \\ 0x_1 & + & 1x_2 & + & 0x_3 & + & 0x_4 & + & 1x_5 & + & 0x_6 & - & 1x_7 & = & 0, \\ 0x_1 & + & 0x_2 & + & 1x_3 & + & 1x_4 & - & 1x_5 & - & 2x_6 & + & 1x_7 & = & 0. \end{array}\right.$$

TSS данной системы состоит из таких четырех векторов-решений:

$$s_1 = (0, 0, 0, 0, 1, 0, 1),\ s_2 = (1, 0, 2, 0, 0, 1, 0), s_3 = (1, 0, 0, 2, 0, 1, 0),\ s_4 = (1, 2, 0, 0, 0, 1, 2).$$

Вектор $s_1 = (0, 0, 0, 0, 1, 0, 1)$ сразу заносим в базис и в дальнейшем он используется при сравнении в случае пополнения базиса и при чистке множества решений, полученного на очередном шаге вычислений.

На первом шаге ($i = 1$) работы алгоритма получаем такое множество векторов и соответствующих им значений (значения представлены в последнем столбике):

(1,2,0,0,0,1,2 | 0)	| 1 |
(1,0,2,0,0,1,0 | 0)	| 1 |
(1,0,0,2,0,1,0 | 0)	| 1 |
(0,0,0,0,0,0,0 | 1)	| -1|
(0,0,0,0,0,0,0 | 0)	| 1 |

На последующих шагах алгоритма получаем следующую последовательность векторов и соответствующих им значений (для i = 2):

(1,2,0,0,0,1,2 | 1)	| 1|
(1,0,2,0,0,1,0 | 1)	|-1|
(1,0,0,2,0,1,0 | 1)	|-1|
(0,0,0,0,0,0,0 | 1)	|-1|
(0,0,0,0,0,0,0 | 0)	|1 |

Для $i = 3$:

(1,1,1,0,0,1,1 | 1)	заносим в базис	
(1,1,0,1,0,1,1 | 1)	заносим в базис	
(1,2,0,0,0,1,2 | 2)	|-2|	
(1,0,2,0,0,1,0 | 1)	| 1|	
(1,0,0,2,0,1,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 0)	|1 |	

Для $i = 4$:

(3,2,4,0,0,3,2 | 4)	удаляем > (1,1,1,0,0,1,1)	
(1,0,1,1,0,1,0 | 1)	заносим в базис	
(1,2,0,0,0,1,2 | 2)	|-2|	
(1,0,2,0,0,1,0 | 2)	|-2|	
(1,0,0,2,0,1,0 | 1)	| 1|	
(0,0,0,0,0,0,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 0)	|1 |	

Для $i = 5$:

(3,2,0,4,0,3,2 | 4)	удаляем > (1,1,0,1,0,1,0)	
(3,0,2,4,0,3,0 | 4)	удаляем > (1,0,1,1,0,1,0)	
(1,2,0,0,0,1,2 | 2)	|-2|	
(1,0,2,0,0,1,0 | 2)	|-2|	
(1,0,0,2,0,1,0 | 2)	|-2|	
(0,0,0,0,0,0,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 0)	|1 |	

Для $i = 6$:

(1,2,0,0,0,1,2 | 2)	|-1|	
(1,0,2,0,0,1,0 | 2)	|-1|	
(1,0,0,2,0,1,0 | 2)	|-1|	
(0,0,0,0,0,0,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 0)	|1 |	

Для $i = 7$:

(1,2,0,0,0,1,2 | 2)	| 0|	
(1,0,2,0,0,1,0 | 2)	|-2|	
(1,0,0,2,0,1,0 | 2)	|-2|	
(0,0,0,0,0,0,0 | 1)	|-1|	
(0,0,0,0,0,0,0 | 0)	|1 |	

Окончательно получаем такие вектора:

(1,2,0,0,0,1,2 | 2)
(1,0,2,0,0,1,0 | 2)
(1,0,0,2,0,1,0 | 2)
(0,0,0,0,0,0,0 | 1)
(0,0,0,0,0,0,0 | 0)

Удаляя два последних вектора и последний столбик, окончательно получаем такие вектора

$$(1,2,0,0,0,1,2),\ (1,0,2,0,0,1,0),\ (1,0,0,2,0,1,0).$$

Отбрасывая вектора, содержащие координаты большие 1 (в данном случае это векторы из TSS), получаем такой базис множества решений данной СЛОДУ:

$$(0,0,0,0,1,0,1),\ (1,1,1,0,0,1,1),\ (1,1,0,1,0,1,1),\ (1,0,1,1,0,1,0).$$

11. Автоматные методы решения ЛОДУ

Методы, которые будут рассмотрены в этом разделе, разрабатывались для решения ЛОДУ и СЛОДУ над множеством натуральных чисел. Для того, чтобы рассмотреть эти методы, необходимо привести некоторые определения из теории конечных автоматов без выходов (так называемых X-автоматов) и языков, слова которых акцептируются этими автоматами.

11.1 Конечные X-автоматы и регулярные языки

Конечное множество $X = \{x_1, x_2, \ldots, x_n\}$ попарно разных элементов называется **алфавитом**. Элементы, составляющие алфавит, называются **символами** или **буквами**. **Словом** в алфавите X называется конечная последовательность p его символов, выписанных один за другим, т. е. $p = y_1 y_2 \ldots y_k$, где $y_j \in X,\ j = 1, 2, \ldots, k,\ k \in \mathcal{N}$.

Длиной $l(p)$ слова $p \in F(X)$ называется количество символов алфавита X, которые составляют это слово. Множество всех слов конечной длины в алфавите X обозначают $F(X)$. К этому множеству слов добавляется одно специальное слово, которое называется *пустым словом* и по определению не имеющее ни одного символа алфавита. Пустое слово обозначается e. На множества $F(X)$ вводится операция конкатенации слов: для слов $p = y_1 \ldots y_k$ и $q = z_1 \ldots z_l$ в алфавите X их конкатенацией называется слово $pq = y_1 \ldots y_k z_1 \ldots z_l$, полученное в результате приписывания символов слова q к символам слова p. Очевидно, что такая операция удовлетворяет закону ассоциативности и пустое слово играет роль единицы относительно операции конкатенации.

Произвольное подмножество $L \subseteq F(X)$ множества слов в алфавите X называется *языком* или *событием* в этом алфавите. Пустое множество слов называется **пустым языком** или **невозможным событием**.

Определение 17. *Конечным инициальным X-автоматом (KA) или автоматом без выходов в алфавите X называется пятерка $\mathcal{A} = (A, X, f, a_0, F)$, где A – конечное множество, элементы которого называются состояниями, X – конечный алфавит, символы которого называются входными символами автомата, $f : A \times X \to A$ – функция переходов, $a_0 \in A$ – начальное состояние, $F \subseteq A$ – множество заключительных или акцептирующих состояний.*

Функция переходов X-автомата может быть как частичной, так и полностью определенной. Конечный автомат, у которого функция переходов частичная, называется *частичным*, а автомат, у которого функция переходов полностью определенная, называется *полным*.

Функция переходов расширяется на все множество слов во входном алфавите X с помощью такого рекуррентного соотношения:

$$f(a, e) = a, \quad f(a, p) = f(f(a, x), p'),$$

где $p = xp',\ x \in X$.

Говорят, что **слово** $p \in F(X)$ **акцептируется автоматом** $\mathcal{A} = (A, X, f, a_0, F)$, если $f(a_0, p) \in F$, т. е. если это слово переводит автомат $\mathcal{A}$ из начального состояния в одно из заключительных состояний. В противном случае говорят, что слово **не акцептируется** этим автоматом. Язык L во входном алфавите X **акцептируется автоматом** $\mathcal{A}$, если каждое слово этого языка акцептируется автоматом $\mathcal{A}$, т. е. $(\forall p \in L)\ f(a_0, p) \in F$.

На множестве языков в алфавите X рассматриваются операции, среди которых основными являются три операции, определяемые следующим образом.

Пусть L_1 и L_2 – некоторые языки в алфавите X.

Дизъюнкцией $L_1 \vee L_2$ языков L_1 и L_2 называется теоретико-множественное объединение множеств L_1 и L_2, т. е. $L_1 \vee L_2 = L_1 \cup L_2$.

Произведением $L_1 \cdot L_2$ языков L_1 и L_2 называется язык $L = \{pq | p \in L_1 \wedge q \in L_2\}$, т. е. язык всех возможных конкатенаций слов языка L_1 со словами языка L_2.

Итерацией $\{L_1\}$ языка L_1 называется язык $L = \{p_1 p_2 \ldots p_k | p_i \in L_1,\ i = 1, 2, \ldots, k\}$.

Одноэлементные языки $x_1, x_2, \ldots, x_n, e$, где e – пустое слово, а $x_i \in X, i = 1, 2, \ldots, n$, называются **элементарными**. Язык, полученный в результате применения конечного чис-

ла операций дизъюнкции, произведения и итерации к элементарным языкам, называется **регулярным языком**, а его представление через элементарные языки и три основных операции – регулярным выражением этого языка.

Кроме этих трех основных операций на регулярных языках рассматриваются и другие теоретико-множественные операции – пересечение и дополнение.

Пересечением языков L_1 и L_2 называется теоретико-множественное пересечение множеств L_1 и L_2.

Дополнением языка $L \subseteq F(X)$ называется язык $\bar{L} = F(X) \setminus L$.

Хорошо известно, что класс языков, которые акцептируются КА, замкнут относительно теоретико-множественных операций объединения, пересечения и дополнения, т. е. является булевой алгеброй. Более того, для КА имеет место

Теорема 38. *Язык акцептируется конечным X-автоматом тогда и только тогда, когда он регулярный язык* [41].

Для языков, которые принадлежат классу регулярных языков, существуют алгоритмы, которые строят по регулярному выражению языка соответствующий конечный X-автомат (алгоритм синтеза) и по конечному X-автомату регулярное выражение языка, слова которого акцептирует этот автомат (алгоритм анализа).

Рассмотрим автоматные подходы к решению линейных ограничений.

11.2 Автоматная интерпретация алгоритма Фортенбахера

Метод Фортенбахера решения одного ЛОДУ, который рассматривался в предыдущих разделах, можно сформулировать в терминах конечных частичных X-автоматов. То факт, что такой автомат существует, вытекает из условия Фортенбахера: сумма чисел $a(x)$ и $a(e_i)$, которые удовлетворяют условию (C_1), всегда строго меньше по модулю от большего из этих чисел. Это значит, что алгоритм Фортенбахера генерирует процесс, который заканчивается через конечное число шагов. Приведем формальные определения этого процесса.

Пусть $a(x) = a_1x_1 + a_2x_2 + \ldots + a_qx_q = 0$ – ЛОДУ, $Q = \{0\}$ – одноэлементное множество, а $X = \{e_1, e_2, \ldots, e_q\}$ – алфавит, элементами которого являются векторы канонического базиса. Определим X-автомат $\mathcal{A} = (Q, X, f, \{0\}, \{0\})$ таким образом:

$$f(m, e_i) = \begin{cases} a(e_i), & \text{если} \quad m = 0, \\ m, & \text{если} \quad a(e_i) = 0, \\ m + a(e_i), & \text{если} \quad m \cdot a(e_i) < 0, \\ *, & \text{иначе,} \end{cases}$$

где $m \in Q$ (целое число), $e_i \in X$, $i = 1, \ldots, q$, а символ $*$ означает неопределенность функции переходов f.

Если процесс генерации состояний автомата $\mathcal{A}$ замкнется, то строим базис множества решений ЛОДУ $a(x) = 0$. Для этого нужно найти все слова p в алфавите X, которые соответствуют всем простым цикличным путям из начального состояния 0 в это же состояние 0, построить векторы-решения, соответствующие этим словам, и удалить одинаковые векторы (одинаковые векторы появляются из-за коммутативности операции сложения). Построенный таким образом X-автомат будет частичным детерминированным автоматом и множество решений ЛОДУ определяется этим автоматом однозначно. Отсюда вытекает

Теорема 39. *Множество решений уравнения $a(x) = 0$ является регулярным языком. Базис множества решений этого уравнения состоит из слов в алфавите X, которые*

соответствуют всем простым циклическим путям в автомате $\mathcal{A}$ *из состояния* 0 *в состояние* 0.

Алгоритм построения автомата $\mathcal{A}$ *всегда заканчивает свою работу. Число состояний автомата* $\mathcal{A}$*, порождаемых алгоритмом, ограничено величиной* $\sum_{j=1}^{q} |a_{ij}| + 1$.

Пример 29. Пусть дано $a(x) = 2x_1 + 0x_2 - x_3 + 3x_4 = 0$ ЛОДУ. Автомат, который акцептирует все решения данного уравнения, строим в соответствии с вышеприведенными правилами. Единственным начальным состоянием является состояние 0, оно же и единственное заключительное состояние. Дальше последовательно генерируем состояния:

$$f(0,e_1) = 2, f(0,e_2) = 0, f(0,e_3) = -1, f(0,e_4) = 3, f(2,e_2) = 2, f(2,e_3) = 1,$$
$$f(-1,e_1) = 1, f(-1,e_2) = -1, f(-1,e_4) = 2, f(1,e_2) = 1, f(1,e_3) = 0, f(3,e_2) = 3, f(3,e_3) = 2.$$

В результате получаем такой автомат:

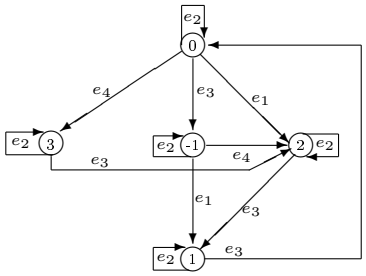


Рис. 11.2.1. Автомат $\mathcal{A}$, который акцептирует решения ЛОДУ ♠

Применение алгоритма анализа КА. Для построения базиса множества всех решений ЛОДУ можно применить алгоритм анализа КА, который использует метод решения систем линейных уравнений в алгебре регулярных языков. Продемонстрируем этот метод для ЛОДУ с предыдущего примера.

Пример 30. Дано ЛОДУ $a(x) = 2x_1 + 0x_2 - x_3 + 3x_4 = 0$, которому соответствует КА, граф переходов которого показан на рис. 11.2.1. Найти базис множества решений данного ЛОДУ.

Для построения базиса множества решений данного ЛОДУ применяем алгоритм анализа КА, описанный в [13]. Этому автомату соответствует такая праволинейная система уравнений в алгебре регулярных языков:

$$S = \begin{cases} S_0 &= e_1S_2 \vee e_3S_{-1} \vee e_4S_3 \vee e_2S_0 \vee e, \\ S_{-1} &= e_2S_{-1} \vee e_4S_2 \vee e_1S_1, \\ S_1 &= e_2S_1 \vee e_3S_0, \\ S_2 &= e_2S_2 \vee e_3S_1, \\ S_3 &= e_2S_3 \vee e_3S_2. \end{cases}$$

Поскольку акцептирующим состоянием является состояние 0, то нас интересует язык S_0 этой системы. Решая последние четыре уравнения, находим

$$\begin{aligned} S_{-1} &= \{e_2\}(e_4S_2 \vee e_1S_1), \\ S_1 &= \{e_2\}e_3S_0, \\ S_2 &= \{e_2\}e_3S_1, \\ S_3 &= \{e_2\}e_3S_2. \end{aligned}$$

Подставляя правые части выражения для S_1 у выражения для S_2 и S_{-1}, а потом полученное выражение для S_2 в остальные выражения, получаем такие выражения:

$$\begin{aligned} S_{-1} &= (\{e_2\}(e_4\{e_2\}e_3\{e_2\}e_3 \vee e_1\{e_2\}e_3)S_0, \\ S_1 &= \{e_2\}e_3S_0, \\ S_2 &= \{e_2\}e_3\{e_2\}e_3S_0, \\ S_3 &= \{e_2\}e_3\{e_2\}e_3\{e_2\}e_3S_0. \end{aligned}$$

Теперь найденные выражения подставляем в первое выражение для S_0

$$S_0 = (e_1\{e_2\}e_3\{e_2\}e_3 \vee e_3\{e_2\}e_4\{e_2\}e_3\{e_2\}e_3 \vee e_1\{e_2\}e_3 \vee e_4\{e_2\}e_3\{e_2\}e_3\{e_2\}e_3 \vee e_2)S_0 \vee e$$

и решаем полученное уравнение относительно S_0:

$$S_0 = \{e_1\{e_2\}e_3\{e_2\}e_3 \vee e_3\{e_2\}e_4\{e_2\}e_3\{e_2\}e_3 \vee e_3e_1\{e_2\}e_3 \vee e_4\{e_2\}e_3\{e_2\}e_3\{e_2\}e_3 \vee e_2\}.$$

Заменим в полученном выражении итерационные скобки дизъюнкцией пустого слова и выражения, которое находится в этих итерационных скобках, и, применяя закон дистрибутивности, умножим, а потом отбросим знак дизъюнкции и пустое слово. В результате получим такое множество слов

$$e_2,\ e_1e_2e_3e_2e_3,\ e_3e_2e_4e_2e_3e_2e_3,\ e_3e_1e_2e_3,\ e_4e_2e_3e_2e_3e_2e_3,$$

из которого удалены одинаковые слова. Поскольку $e_2 = (0,1,0,0)$ – решение ЛОДУ, то остальные слова упрощаются путем удаления символа e_2 из остальных слов. Следовательно, данное множество редуцируется к такому виду:

$$e_2,\ e_1e_3e_3,\ e_3e_4e_3e_3.$$

А этому множеству слов соответствует такое множество базисных решений ЛОДУ:

$$m_1 = (0,1,0,0), m_2 = (1,0,2,0), m_3 = (0,0,3,1).\spadesuit$$

Следует отметить, что данный метод можно обобщить и применить к неоднородным линейным диофантовым уравнениям и системам линейных однородных диофантовых неравенств.

Рассмотрим случай неравенства, которое является отрицанием равенства, т. е. неравенства вида

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \neq 0, \tag{45}$$

где $a_i \in \mathcal{Z}, x_i \in \mathcal{N}, i = 1,2,\ldots,n$. Без ограничения общности будем считать, что НОД$(a_1,a_2,\ldots,a_n) = 1$.

Способ решения такого типа неравенств вытекает из теоремы 39. Действительно, поскольку множество решений ЛОДУ

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0, \tag{46}$$

является регулярным языком, то дополнение этого языка, элементы которого являются решениями (45), тоже будет регулярным языком, который акцептируется тем же автоматом. Только в этом автомате заключительные состояния становятся незаключительными, а незаключительные – заключительными. Следовательно можно применить описанные выше способы построения базиса множества решений неравенств вида (45). Рассмотрим детали.

Чтобы применить автомат к построению базиса множества решений неравенства (45), его необходимо преобразовать в полный автомат. Это преобразование состоит в том, что к множеству состояний добавляется дополнительное “дьявольское состояние” d, в которое определяются все неопределенные переходы в исходном автомате и зацыкливаются все переходы из d в d, т.е. $(\forall x \in X)\ f(d,x) = d$.

Например, преобразованный таким образом автомат из рис. 11.2.1 принимает вид.

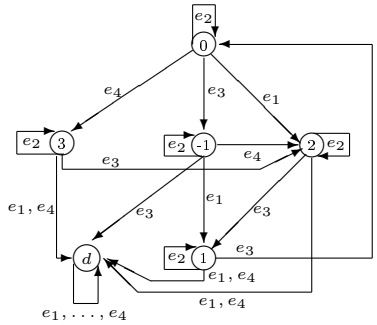


Рис. 11.2.2. Полный автомат, для автомата $\mathcal{A}$

Автомат, акцептирующий множество решений неравенства (45) имеет все состояния акцептирующими, за исключением начального состояния. Применяя метод решения, описанный выше, находим базис множества всех решений данного неравенства.

Пример 31. Рассмотрим неравенство $2x_1+0x_2-x_3+3x_4\neq 0$. Найдем базис множества его решений, применяя алгоритм анализа КА (этот автомат показан на рис. 11.2.2). Поскольку акцептирующими состояниями являются 1,2,3,-1 и d, то нас интересуют языки S_1, S_2, S_3, S_{-1} и S_d, которые включают все решения данного неравенства.

Составляем систему уравнений

$$S=\begin{cases} S_0 &= e_1S_2\vee e_3S_{-1}\vee e_4S_3\vee e_2S_0,\\ S_{-1} &= e_2S_{-1}\vee e_4S_2\vee e_1S_1\vee s_3S_d\vee e,\\ S_1 &= e_2S_1\vee e_3S_0\vee (e_1\vee e_4)S_d\vee e,\\ S_2 &= e_2S_2\vee e_3S_1\vee (e_1\vee e_4)S_d\vee e,\\ S_3 &= e_2S_3\vee e_3S_2\vee (e_1\vee e_4)S_d\vee e,\\ S_d &= ((e_1\vee e_2\vee e_3\vee e_4)S_d\vee e. \end{cases}$$

Решениями данной системы (т.е. решениями, которые соответствуют кратчайшим путям из начального состояния в заключительные состояния) будут векторы e_1, e_4, e_3e_1 и e_3. Первые векторы соответствуют положительным значениям неравенства 2, 3, 1 соответственно, а второй вектор – отрицательному значению -1 (эти решения можно увидеть непосредственно по автомату). Таким образом, общее решение данного неравенства принимает вид:

$$x=(c_1e_1+c_2e_4+c_3e_3e_1)+\sum_{j=1}^{3}d_jv_j\cup c_4e_3\sum_{j=1}^{3}d_jv_j,$$

где $v_1=(1,0,2,0), v_2=(0,1,0,0), v_3=(0,0,3,1)$ – базисные решения ЛОДУ, которые соответствуют данному неравенству, а коэффициенты c_i, d_j – произвольные неотрицательные целые числа, $i=1,2,3,4, j=1,2,3$.

Решения приведенного неравенства можно было бы искать путем решения двух неравенств $2x_1+0x_2-x_3+3x_4>0$ и $2x_1+0x_2-x_3+3x_4<0$, используя вышеописанные методы. Однако, автоматный способ более лаконичный, хотя его сложность остается высокой и возрастает с увеличением числа неизвестных. ♠

Из приведенного примера следует, что все векторы канонического базиса, из которых достижимы заключительные состояния, будут принадлежать базису множества решений (как векторы, соответствующие кратчайшим путям из начального состояния в заключительные) и из остальных состояний в "дьяволское состояние" d будут вести переходы под действием тех же векторов, то вводить состояние d нет необходимости. Это хорошо видно из автомата, приведенного на рис. 12.2.2. Если состояние d исключить из множества заключительных состояний, то система уравнений в алгебре регулярных языков для автомата упрощается и принимает вид:

$$S=\begin{cases} S_0 &= e_1S_2\vee e_3S_{-1}\vee e_4S_3\vee e_2S_0,\\ S_{-1} &= e_2S_{-1}\vee e_4S_2\vee e_1S_1\vee e,\\ S_1 &= e_2S_1\vee e_3S_0\vee e,\\ S_2 &= e_2S_2\vee e_3S_1\vee e,\\ S_3 &= e_2S_3\vee e_3S_2\vee e. \end{cases}$$

Рассмотрим теперь задачу проверки выполнимости системы ограничений над множеством натуральных чисел $\mathcal{N}$ вида

$$S=\begin{cases} L_1(x) &= a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n &= 0,\\ \ldots &\ldots\ldots\ldots\ldots\ldots\ldots\ldots &\ldots\\ L_{s-1}(x) &= a_{s-11}x_1+a_{s-12}x_2+\ldots+a_{s-1n}x_n &= 0,\\ L_s(x) &= a_{s1}x_1+a_{s2}x_2+\ldots+a_{sn}x_n &\neq 0. \end{cases}$$

Построим базис множества решений $B_1=\{b_1,b_2,\ldots,b_k\}$ подсистемы, состоящей из первых $s-1$ уравнений, и базис множества решений $B_2=\{c_1,c_2,\ldots,c_l\}$ СЛОДУ S'

$$S'=\begin{cases} L_1(x) &= a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n &= 0,\\ \ldots &\ldots\ldots\ldots\ldots\ldots\ldots\ldots &\ldots\\ L_{s-1}(x) &= a_{s-11}x_1+a_{s-12}x_2+\ldots+a_{s-1n}x_n &= 0,\\ L_s(x) &= a_{s1}x_1+a_{s2}x_2+\ldots+a_{sn}x_n &= 0. \end{cases}$$

Рассматривая базисные решения b_i из B_1 как канонические векторы для неравенства $L_s(x) \neq 0$, построим вышеописанным способом решения этого неравенства $y_1, y_2, \ldots, y_r$ со значениями на них $L_s(y_i) > 0$ и $z_1, z_2, \ldots, z_m$ со значениями на них $L_s(z_j) < 0$. Тогда общее решение системы S запишется в виде

$$x = \sum_{i=1}^{r} d_i y_i + \sum_{j=1}^{l} a_j c_j \cup \sum_{p=1}^{m} n_p z_p + \sum_{q=1}^{l} a_q c_q,$$

где $d_i, n_q \in \mathcal{N}$ – произвольные числа одновременно не равные нулю.

Пример 32. Найти общее решение системы

$$S = \left\{ \begin{array}{lcrcl} L_1(x) & = & -x_1 + x_2 + 2x_3 - 3x_4 & = & 0, \\ L_2(x) & = & -x_1 + 3x_2 - 2x_3 - x_4 & = & 0, \\ L_3(x) & = & 2x_1 + 0x_2 - x_3 + 3x_4 & \neq & 0. \end{array} \right.$$

Базис множества всех решений СЛОДУ, состоящей из первыхъ двух уравнений, был найден в примере 21: $B_1 = \{b_1 = (0,1,1,1), b_2 = (4,2,1,0)\}$. Система S' несовместна ($B_2 = \emptyset$), а на базисных векторах из B_1 функция L_3 принимает значения 2 и 7 соответственно. Следовательно, общее решение системы ограничений S принимает вид

$$x = d_1 b_1 + d_2 b_2,$$

где $d_1, d_2 \in \mathcal{N}$ – произвольные не равные одновременно нулю числа. ♠

Автоматный подход к решению СЛОДУ, СЛНДУ, ЛОДУ, ЛНДУ и ЛНДН рассматривали Ромеуф [44] и Комон [46].

11.3 Метод Ромеуфа

Используя конечные X-автоматы, Ромеуф обобщил метод Фортенбахера на случай СЛНДУ, которая состоит из двух уравнений, и получил эффективные алгоритмы решения такого типа систем [44].

Пусть $X = \{e_1, e_2, \ldots, e_q\}$ – алфавит, элементами которого являются вектори канонического базиса, и

$$S = \left\{ \begin{array}{lcrcl} a_{11}x_1 & + & a_{12}x_2 & +\ldots+ & a_{1q}x_q & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & +\ldots+ & a_{2q}x_q & = & b_2 \end{array} \right.$$

– СЛНДУ. Обозначим

$$a_{1q+1} = -b_1, \quad a_{2q+1} = -b_2, \quad l_1 = \max_{a_{1i} \leq 0} |a_{1i}|, \quad l_2 = \max_{a_{1i} \geq 0} a_{1i},$$
$$p_1 = \max_{a_{2i} \leq 0} |a_{2i}|, \quad p_2 = \max_{a_{2i} \geq 0} a_{2i}, \quad \forall i \in [1, q]\ a_i = (a_{1i}, a_{2i}) \text{ и } b = (b_1, b_2).$$

Введем в рассмотрение пары $a_i = (a_{1i}, a_{2i})$ для всех $i = 1, 2, \ldots, q$ и множество

$$Q_0 = ([-l_1, l_2] \times ([-p_1, p_2]) \cup ((-l_1, l_2] \times (p_2, 2p_2]) \cup ([-2l_1, -l_1) \times (-p_1, p_2]) \cup$$
$$\cup ([-l_1, l_2) \times [-2p_1, -p_1)) \cup ((l_2, 2l_2] \times [-p_1, p_2)),$$

которое является объединением декартовых произведений целочисленных интервалов (закрытых и полуоткрытых).

Рассмотрим X-автомат $\mathcal{A} = (Q, X, q_0 = \{-b\}, f, F = \{0\})$, функция переходов которого определяется таким образом:

$$f(q, e_i) = q + a_i,$$

где $q, q+a_i \in Q \subset \mathcal{Z}^2, Q_0 \subset Q, e_i \in X$, а $\mathcal{Z}$ – множество целых чисел.

Основное свойство такого автомата вытекает из следующей теоремы.

Теорема 40. *Автомат $\mathcal{A}$ акцептирует все решения СЛНДУ S.*

На основе этой теоремы было построено три алгоритма:

– алгоритм вычисления одного минимального решения СЛНДУ;

– алгоритм вычисления всех минимальных решений СЛНДУ;

– алгоритм вычисления базиса множества всех решений СЛОДУ.

Первый алгоритм очевидным образом вытекает из вышеприведенной теоремы 40. Действительно, если автомат $\mathcal{A}$ акцептирует все решения данной СЛНДУ, то решению минимальной длины в автомате будет соответствовать кратчайший простой путь из начального состояния q_0 в заключительное состояние 0.

Второй алгоритм также является очевидным следствием теоремы 40. Действительно, если $c = (c_1, c_2, \ldots, c_q)$ – минимальное решение СЛНДУ S, то решению c в автомате $\mathcal{A}$ соответствует простой путь из начального состояния q_0 в заключительное состояние 0 (слова “простой путь” означают, что никакое состояние на этом пути не появляется дважды).

Менее очевидным есть следующий результат. Пусть

$$L = (l_1 + l_2 + 1)(p_1 + p_2 + 1) + 2(l_1 + l_2)(p_1 + p_2),$$

тогда имеет место

Теорема 41. $\max\limits_{c \in M} |c| \le L - 1$, *где M – множество минимальных решений СЛНДУ S.*

Третий алгоритм также очевиден и вытекает из теоремы 40. Действительно, минимальному решению СЛОДУ соответствует простой путь минимальной длины с начального состояния $q_0 = 0$ в заключительное состояние 0. Ограничение на длину базисного решения c дает

Теорема 42. $\max\limits_{c \in B} |c| \le L$, *где B - базис множества всех решений СЛОДУ S.*

Характеристика сложности приведена только для первого алгоритма и выражается величиной $O(ql^2)$, где $l = \max(l_1, l_2, p_1, p_2)$. Для остальных алгоритмов в работе Ромеуфа такие характеристики не приводятся.

Пример построения минимального решения. Найти минимальное решение нижеприведенной СЛНДУ методом Ромеуфа.

$$S = \begin{cases} -x_1 + x_2 + 2x_3 = 3, \\ -x_1 + 3x_2 - 2x_3 = 1. \end{cases}$$

Решение. По системе S находим

$$l_1 = 1, \quad l_2 = 2, \quad p_1 = 2, \quad p_2 = 3, \quad b = (-3, -1).$$

Следовательно, множество Q_0 принимает вид:

$$Q_0 = ([-1, 2] \times [-2, 3] \cup (-1, 2] \times (3, 6] \cup [-2, -1) \times (-2, 3] \cup [-1, 2) \times [-4, -2) \cup (2, 4] \times [-2, 3)).$$

Таким образом, множество Q включает такие пары:

$$Q = \{(-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-1,3), (0,-2), (0,-1), (0,0),$$
$$(0,1), (0,2), (0,3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (1,3), (2,-2), (2,-1), (2,0),$$
$$(2,1), (2,2), (2,3), (0,4), (0,5), (0,6), (1,4), (1,5), (1,6), (2,4), (2,5), (2,6), (-2,-1),$$
$$(-2,0), (-2,1), (-2,2), (-2,3), (-1,-4), (-1,-3), (0,-4), (0,-3), (1,-4), (1,-3),$$
$$(3,-2), (3,-1), (3,0), (3,1), (3,2), (4,-2), (4,-1), (4,0), (4,1), (4,2)\}.$$

Отсюда получаем такой фрагмент автомата:

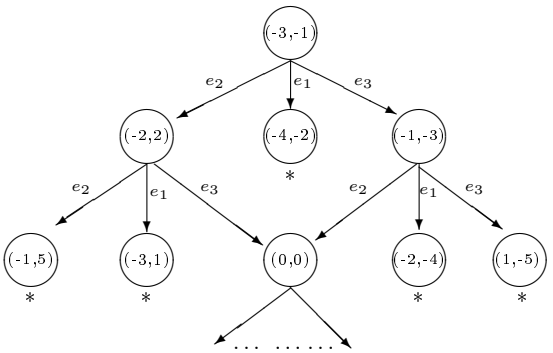


Рис. 11.2.3. Фрагмент автомата для СЛНДУ S

Поскольку пары (-3,1), (-1,5), (-2,-4), (1,-5) и (-2,-4) не принадлежат интервалам множества Q_0, а потому и к множеству Q, то процесс построения искомого решения заканчивается. С этого фрагмента очевидным образом вытекает, что единственным решением СЛНДУ S является $c = e_2 + e_3 = (0, 1, 1)$.♠

11.4 Метод Комона

Алгоритм реализации метода Комона [46] использует двоичное представление натуральных чисел, т. е. использует слова в алфавите $\{0, 1\}$. Например, число 13 в таком представлении имеет вид двоичного слова 1011, а пара (13, 6) выглядит так

$$\begin{matrix} 1011 \\ 0110 \end{matrix}.$$

Символами входного алфавита автомата являются поразрядные двоичные векторы, с которых и строится двоичное представление натуральных чисел. Так для пары (13, 6), символами алфавита будут векторы-столбики

$$x_1 = \begin{matrix} 1 \\ 0 \end{matrix}, \quad x_2 = \begin{matrix} 0 \\ 1 \end{matrix}, \quad x_3 = \begin{matrix} 1 \\ 1 \end{matrix}, \quad x_4 = \begin{matrix} 1 \\ 0 \end{matrix}.$$

Пользуясь этим представлением, Комон предложил алгоритм, который для произвольного линейного неоднородного диофантового уравнения (ЛНДУ)

$$a(x) = a_1x_1 + a_2x_2 + \ldots + a_qx_q = b \tag{47}$$

строит конечный X-автомат, акцептирующий решения данного ЛНДУ. Этот алгоритм имеет вид:

Automat
Вход: коэффициенты $a_1, a_2, \ldots, a_q, b$ уравнения $a(x) = b$.
Выход: автомат $\mathcal{A}$, акцептирующий все решения $a(x) = b$.
Метод:
begin
$a_0 = \{b\}$; $A_1 = \{b\}$;
$A = \emptyset$; $f = \emptyset$;
while $A_1 \neq \emptyset$ *do*;
choose $c \in A_1$;

$A_1 = A_1 \setminus \{c\}$;
$A = A \cup \{c\}$;
let S *be the solutions modulo* 2 *of* $a_1x_1 + \ldots + a_qx_q = c$;
for each $(c_1, c_2, \ldots, c_q) \in S$ *do*;
$d = \frac{c - a_1c_1 - \ldots - a_qc_q}{2}$;
if $d \notin A \cup A_1$ *then* $A_1 = A_1 \cup \{d\}$; *fi*
$f = f \cup \{c, s, d\}$ *where* $s \in \{0,1\}^q$ *encodes* $(c_1, c_2, \ldots, c_q)$;
od
od
if $0 \in A$ *then* $F = \{0\}$ *else* $F = \emptyset$;
end

Пример работы алгоритма Комона. Построить конечный X-автомат, который акцептирует решения ЛНДУ

$$x + 2y - 3z = 1.$$

Решение. Вначале имеем $a_0 = 1,\ A_1 = \{1\},\ c = 1$. Найдем решения по модулю 2 уравнения $x + 2y - 3z = 1$, т. е. строим множество S. Базис множества решений этого уравнения легко найти с помощью TSS-алгоритма (см. раздел 6.1.1) и его составляют векторы $(1,0,1,0),\ (1,0,0,1),\ (0,1,0,0)$. Для построения множества S всех решений по модулю 2, которые нужны в алгоритме Комона, необходимо построить все комбинации базисных векторов и выбрать те, у которых последняя координата равняется 1. Такими векторами будут:

$$(1,0,0,1),\ (0,0,1,1),\ (1,1,0,1),\ (0,1,1,1).$$

Отбрасывая последнюю координату в этих векторах, окончательно получаем такое множество решений:

$$S = \{x_1^T = (0,0,1),\ x_2^T = (1,0,0),\ x_3^T = (0,1,1),\ x_4^T = (1,1,0)\}.$$

Далее, согласно алгоритму *Automat*, получаем последовательно для

$$x_1^T = (0,0,1) \to d = \tfrac{1-0-0+3}{2} = 2; \quad x_2^T = (1,0,0) \to d = \tfrac{1-1-0+0}{2} = 0;$$
$$x_3^T = (0,1,1) \to d = \tfrac{1-0-2+3}{2} = 1; \quad x_4^T = (1,1,0) \to d = \tfrac{1-1-2+0}{2} = -1.$$

В результате выполнения первой итерации цикла *while* получаем новые значения множеств $A = \{1\},\ A_1 = \{0, 2, -1\}$ и такие переходы в автомате:

$$1 \xrightarrow{x_1} 2,\ 1 \xrightarrow{x_2} 0,\ 1 \xrightarrow{x_3} 1,\ 1 \xrightarrow{x_4} -1.$$

Повторяем итерацию, поскольку множество A_1 непусто. Выбираем состояние $c = 0$. Ищем решения уравнения $x + 2y - 3z = 0$ описанным выше способом. Множество решений в этом случае состоит из векторов:

$$S = \{x_5^T = (0,0,0),\ x_6^T = (0,1,0),\ x_7^T = (1,0,1),\ x_8^T = (1,1,1)\}.$$

Согласно алгоритму, получаем последовательно для

$$x_5^T = (0,0,0) \to\ d = \tfrac{0-0-0+0}{2} = 0; \quad x_6^T = (0,1,0) \to\ d = \tfrac{0-0-2+0}{2} = -1;$$
$$x_7^T = (1,0,1) \to\ d = \tfrac{0-1-0+3}{2} = 1; \quad x_8^T = (1,1,1) \to\ d = \tfrac{0-1-2+3}{2} = 0.$$

В результате выполнения первой итерации цикла *while* получаем новые значения множеств $A = \{1, 0\},\ A_1 = \{2, -1\}$ и такие переходы в автомате:

$$0 \xrightarrow{x_5, x_8} 0,\ 0 \xrightarrow{x_6} -1,\ 0 \xrightarrow{x_7} 1.$$

Повторяем итерацию, так как множество A_1 непусто. Выбираем состояние $c = 2$ и находим множество решений уравнения $x + 2y - 3z - 2 = 0$. Это множество составляют векторы-решения уравнения $x + 0y - z = 0$, которые уже были найдены на предыдущей итерации. Тогда

$$x_5^T = (0,0,0) \to\ d = \tfrac{2-0-0+0}{2} = 1; \quad x_6^T = (0,1,0) \to\ d = \tfrac{2-0-2+0}{2} = 0;$$
$$x_7^T = (1,0,1) \to\ d = \tfrac{2-1-0+3}{2} = 2; \quad x_8^T = (1,1,1) \to\ d = \tfrac{2-1-2+3}{2} = 1.$$

В результате выполнения данной итерации получаем $A = \{1,\ 0, 2\},\ A_1 = \{-1\}$ и новые переходы:

$$2 \xrightarrow{x_5, x_8} 1,\ 2 \xrightarrow{x_6} 0,\ 2 \xrightarrow{x_7} 2.$$

Поскольку $A_1 \neq \emptyset$, то выбираем состояние $c = -1$ и повторяем итерацию. Получаем такую последовательность вычислений:

$$x_1^T = (0,0,1) \to\ d = \tfrac{-1-0-0+3}{2} = 1; \quad x_2^T = (1,0,0) \to\ d = \tfrac{-1-1-0+0}{2} = -1;$$
$$x_3^T = (0,1,1) \to\ d = \tfrac{-1-0-2+3}{2} = 0; \quad x_4^T = (1,1,0) \to\ d = \tfrac{-1-1-2+0}{2} = -2.$$

В результате находим, $A = \{1,0,2,-1\},\ A_1 = \{-2\}$ и новые переходы:

$$-1 \xrightarrow{x_1} 1,\ -1 \xrightarrow{x_2} -1,\ -1 \xrightarrow{x_3} 0,\ -1 \xrightarrow{x_4} -2.$$

Та как $A_1 \neq \emptyset$, то выбираем состояние $c = -2$ и повторяем итерацию. Получаем такую последовательность вычислений:

$$x_5^T = (0,0,0) \to\ d = \tfrac{-2-0-0+0}{2} = -1; \quad x_6^T = (0,1,0) \to\ d = \tfrac{-2-0-2+0}{2} = -2;$$
$$x_7^T = (1,0,1) \to\ d = \tfrac{-2-1-0+3}{2} = 0; \quad x_8^T = (1,1,1) \to\ d = \tfrac{-2-1-2+3}{2} = -1.$$

В результате выполнения данной итерации имеем $A = \{1,0,2,-1,-2\},\ A_1 = \emptyset$ и новые переходы:

$$-2 \xrightarrow{x_5,x_8} -1,\ -2 \xrightarrow{x_6} -2,\ -2 \xrightarrow{x_7} 0.$$

На этом работа алгоритма заканчивается (поскольку $A_1 = \emptyset$) и окончательно получаем такой автомат $A = (\{1,0,2,-1,-2\},\ X = \{x_1^T = (0,0,1), x_2^T = (1,0,0), x_3^T = (0,1,1), x_4^T = (1,1,0),\ x_5^T = (0,0,0), x_6^T = (0,1,0), x_7^T = (1,0,1), x_8^T = (1,1,1)\}, f, a_0 = 1,\ F = \{0\})$, функция переходов которого представлена ниже на рисунке графом переходов:

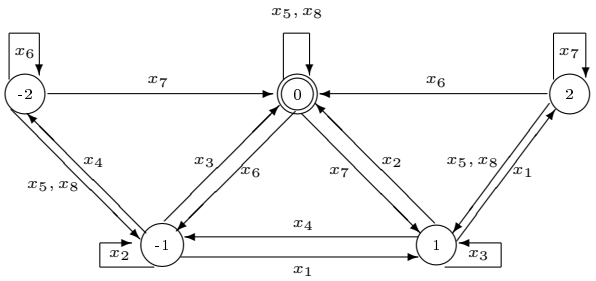


Рис. 11.2.4. Автомат, акцептирующий решения ЛНДУ $x + 2y - 3z = 1$♠

Правильность алгоритма Комона вытекает из такой теоремы [46].

Теорема 43. *Алгоритм* **Automat** *завершает свою работу и строит автомат, количество состояний которого ограничено величиной* $\max(|b|, |a_1|+|a_2|+\ldots+|a_q|)$ *и который акцептирует все решения уравнения* $a_1x_1+ +a_2x_2 + \ldots + a_qx_q = b$.

Алгоритм Комона можно применять и к решению линейных неоднородных диофантовых неравенств (ЛНДН). Отличие состоит в том, что вычисления состояния (числа) d ведется по формуле

$$d = \lfloor \tfrac{c-a_1c_1-a_2c_2-\ldots-a_qc_q}{2} \rfloor,$$

где $\lfloor x \rfloor$ – наибольшее целое число, не превосходящее x, а множество заключительных состояний $F = \{c | c \geq 0\}$.

Пример построения КА для ЛНДН алгоритмом Комона. Построить автомат, который акцептирует решения ЛНДН $2x - y \leq -1$.

Решение. Вначале имеем $A = \{-1\},\ A_1 = \{-1\}$ и $c = -1$. Строим решения уравнения $2x - y + u = -1$ по модулю 2 (u - дополнительная неизвестная). Получаем такое множество решений:

$$(0,0,1,1),\ (0,1,0,1),\ (1,0,1,1),\ (1,1,0,1)\}.$$

Отбрасывая последние две координаты в этих решениях, окончательно получаем такое множество решений данного неравенства:

$$S = \{x_1 = (0,0),\ x_2 = (0,1),\ x_3 = (1,0),\ x_4 = (1,1)\}.$$

Строим переходы из состояния -1:

$$x_1^T = (0,0) \to\ d = \lfloor \tfrac{-1-0+0}{2} \rfloor = -1; \quad x_2^T = (0,1) \to\ d = \lfloor \tfrac{-1-0+1}{2} \rfloor = 0;$$
$$x_3^T = (1,0) \to\ d = \lfloor \tfrac{-1-2+0}{2} \rfloor = -2; \quad x_4^T = (1,1) \to\ d = \lfloor \tfrac{-1-2+1}{2} \rfloor = -1.$$

В результате получаем, $A = \{-1, 0, -2\}$, $A_1 = \{0, -2\}$ и новые переходы:

$$-1 \xrightarrow{x_1, x_4} -1,\ -1 \xrightarrow{x_2} 0,\ -1 \xrightarrow{x_3} -2.$$

Опуская дальнейшие подробности, приведем окончательный вариант автомата $\mathcal{A} = (\{-1, 0,\ 1\}, X = \{x_1, x_2, x_3, x_4\}, f, -1, \{0\})$, который акцептирует решения ЛНДН $2x - y \le -1$.

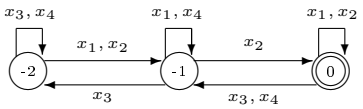


Рис. 11.2.5. Автомат, акцептирующий решения ЛНДН $2x - y \le -1$.♠

12. Таблица сложностей алгоритмов

В заключение приведем итоговую таблицу сложностей алгоритмов решения CSP в дискретных и непрерывных областях:

Область	Сложность CSP	Ссылки
$\mathcal{D}, \mathcal{Q}$	P	[52, 12]
$\mathcal{N}$	NP	[52, 7]
$\{0,1\}$	NP	[52, 10]
Z_m	Сложность проблемы факторизации	[9]
F_p	P	[8]
$\mathcal{Z}$	P	[11]

.

В кольце вычетов Z_m при известном разложении модуля m на простые множители сложность решения CSP принадлежит классу полиномиальной сложности.

Заметим также, что приведенные оценки временных сложностей алгоритмов можно уточнять, если прослеживать все детали процесса вычислений, происходящего в TSS-алгоритме. В данной работе мы ограничиваемся установлением того, что приводим только верхние оценки (т. е. сложность в наихудшем случае) этих алгоритмов.

Отметим также, что при малых значениях модуля p сложностью вычисления НОД в полях и кольцах вычетов можно пренебречь и тогда оценка алгоритмов решения систем в таких полях упрощается. Так, например, в поле F_2, которое часто встречается в приложениях, необходимость вычисления НОД вообще отпадает, поэтому сложность решения СЛОДУ и СЛНДУ в таком поле становится пропорциональна величине sn^2, где s – число уравнений, а n – число неизвестных в системе.

13. Приложения алгоритмов

Рассмотрим некоторые приложения алгоритмов решения систем линейных ограничений.

13.1. Арифметика Пресбургера

В арифметики Пресбургера (АП) используется алфавит переменных $X = \{x, y, \ldots, z, x_1, y_2, \ldots\}$, две константы 0 и 1 (как нульарные операции) и бинарная операция сложения $+$.

Определение 18. *Основным термом АП называется*
– любая переменная из A и константы 0, 1;
– если t_1 и t_2 основные термы АП, то $t_1 + t_2$ – основной терм АП.

Например, $x + y + y + z + z + z + 1 + 1$ – основной терм. При представлении основных термов выполняются обычные сокращения, принятые в арифметике. Так терм $x + y + y + z + z + z + 1 + 1$ записывается в сокращенном виде как $x + 2y + 3z + 2$.

Определение 19. *Атомарной формулой АП называется равенство или неравенство между двумя основными термами, т. е. выражения $t = t'$ или $t \leq t'$, где t, t' – основные термы АП.*

Например, выражения $x + 2y = 3z + 1$ и $2x + 1 \leq y$ является атомарными формулами АП.

Определение 20. *Формулой АП называется*
– любая атомарная формула АП;
– если C и C' формулы АП, то $C \vee C'$, $C \wedge C'$ – формулы АП;
– если C формула АП, то $\neg C$, $\exists x\ C$ и $\forall x\ C$ – формулы АП.

Свободные и связанные переменные у формулах АП определяются стандартным способом: переменная x называется связанной в формуле C, если она находится в области действия квантора существования или общности. В противном случае переменная называется свободной.

Для множества свободных переменных справедливы такие равенства:
$FV(C \vee C') = FV(C) \cup FV(C')$;
$FV(\exists x\ C) = FV(C) \setminus \{x\} = FV(\forall x\ C)$, где $FV(A)$ означает множество свободных переменных формулы A.

Областью интерпретации АП является множество натуральных чисел $\mathcal{N}$, в котором символы 0, 1, +, =, $\leq$ имеют свой обычный смысл.

Решением формулы $C(x_1, x_2, \ldots, x_q)$ называется вектор $c = = (c_1, c_2, \ldots, c_q) \in \mathcal{N}^q$, который удовлетворяет формуле C (т. е. формула $C(c_1, c_2, \ldots, c_q)$ истинна). Например, вектор (0,2,1) является решением формулы $x + 2y = 3z + 1$. Если для данной формулы АП $C(x_1, x_2, \ldots, x_q)$ существует решение, то говорят, что эта формула выполняется.

13.2. Проверка выполнимости формул

Пусть $C(x_1, x_2, \ldots, x_q)$ – атомарная формула АП, свободные переменные которой принадлежат множеству $X = \{x_1, x_2, \ldots, x_q\}$.

Представление решений формулы $C(x_1, x_2, \ldots, x_q, y)$. Рассмотрим случай, когда имеем автомат $\mathcal{A}_{C(x_1,x_2,\ldots,x_q)}$, который акцептирует решения формулы $C(x_1, x_2, \ldots, x_q)$, а нужно построить автомат, акцептирующий решения формулы $C(x_1, x_2, \ldots, x_q, y)$. Это значит, что в формуле $C(x_1, x_2, \ldots, x_q)$ появляется еще одно слагаемое вида $c'y$. Для этого воспользуемся алгоритмом Комона.

Введение одной дополнительной переменной ведет к увеличению числа состояний и числа символов во входном алфавите (это число удваивается в сравнении с числом символов во входном алфавите автомата $\mathcal{A}_{C(x_1,x_2,\ldots,x_q)}$). Если используется алгоритм Комона, то нетрудно понять, что структура автомата $\mathcal{A}_{C(x_1,x_2,\ldots,x_q)}$ не изменится и он входит в автомат $\mathcal{A}_{C(x_1,x_2,\ldots,x_q,y)}$, а изменяются только символы входного алфавита и появляются новые переходы в существующем автомате в новые состояния. Это следует из того, что новые состояния генерируются при помощи формулы

$$d = \frac{c - a_1c_1 - a_2c_2 - \ldots - a_qc_q - a_{q+1}c_{q+1}}{2},$$

где c_{q+1} - коэффициент, стоящий при новой свободной переменной y.

Пример 33. Проиллюстрируем сказанное на примере ЛНДУ $x + 2y + u = 3z + 1$, где u – новая переменная. Для ЛНДУ $x + 2y = 3z + 1$ соответствующий автомат $\mathcal{A} = (A = \{0, 1, 2, -1, -2\}, X = \{x_1, x_2, \ldots, x_8\}, f, 1, \{0\})$ показан на рис. 11.2.4, но он приводится ниже для удобства чтения.

Убедимся в том, что процесс генерации состояний для значения коэффициента $a_{q+1} = 0$ (т. е. в данном случае коэффициента при переменной u) не приводит к появлению новых состояний в существующем автомате $\mathcal{A}_{x+2y+0u=3z+1}$. Сделаем это только для одного состояния $c = 1$, т. е. построим множество всех решений ЛНДУ $x + 2y - 3z + 0u = 1$.

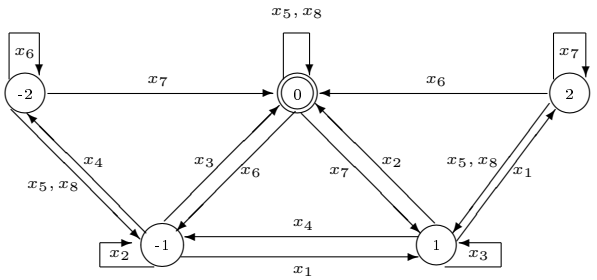


Рис. 13.1.1. Автомат, акцептирующий решения ЛНДУ $x + 2y = 3z + 1$

Применяя метод построения всех решений ЛНДУ, описанный выше, получаем такое множество S всех решений данного уравнения:

$$x_1^T = (0,0,1,0), \quad x_2^T = (1,0,0,0), \quad x_3^T = (0,1,1,0), \quad x_4^T = (1,1,0,0),$$
$$x_5^T = (0,0,1,1), \quad x_6^T = (1,0,0,1), \quad x_7^T = (0,1,1,1), \quad x_8^T = (1,1,0,1).$$

$$x_1^T = (0,0,1,0) \to d = \tfrac{1-0-0+3-0}{2} = 2; \quad x_2^T = (1,0,0,0) \to d = \tfrac{1-1-0+0-0}{2} = 0;$$
$$x_3^T = (0,1,1,0) \to d = \tfrac{1-0-2+3-0}{2} = 1; \quad x_4^T = (1,1,0,0) \to d = \tfrac{1-1-2+0-0}{2} = -1;$$
$$x_5^T = (0,0,1,1) \to d = \tfrac{1-0-0+3-0}{2} = 2; \quad x_6^T = (1,0,0,1) \to d = \tfrac{1-1-0+0-0}{2} = 0;$$
$$x_7^T = (0,1,1,1) \to d = \tfrac{1-0-2+3-0}{2} = 1; \quad x_8^T = (1,1,0,1) \to d = \tfrac{1-1-2+0-0}{2} = -1.$$

В результате выполнения следующих итераций циклу *while* для значения коэффициента $a_{q+1} = 0$ новых состояний не появится.

Рассмотрим процес генерации для значения коэффициента $a_{q+1} = 1$, т. е. найдем все множество S решений для уравнения $x + 2y + u = 3z + 1$. В данном случае множество всех решений состоит из таких векторов:

$$x_9^T = (0,0,0,1), \quad x_{10}^T = (0,1,0,1), \quad x_{11}^T = (1,0,1,1), \quad x_{12}^T = (1,1,1,1),$$
$$x_4^T = (1,1,0,0), \quad x_3^T = (0,1,1,0), \quad x_1^T = (0,0,1,0), \quad x_2^T = (1,0,0,0).$$

$$x_9^T = (0,0,0,1) \to d = \tfrac{1-0-0+0-1}{2} = 0; \quad x_{10}^T = (0,1,0,1) \to d = \tfrac{1-0-2+0-1}{2} = -1$$
$$x_{11}^T = (1,0,1,1) \to d = \tfrac{1-1-0+3-1}{2} = 1; \quad x_{12}^T = (1,1,1,1) \to d = \tfrac{1-1-2+3-1}{2} = 0;$$
$$x_4^T = (1,1,0,0) \to d = \tfrac{1-1-2+0-0}{2} = -1; \quad x_3^T = (0,1,1,0) \to d = \tfrac{1-0-2+3-1}{2} = 1;$$
$$x_1^T = (0,0,1,0) \to d = \tfrac{1-0-0+3-1}{2} = 2; \quad x_2^T = (1,0,0,0) \to d = \tfrac{1-1-0+0-0}{2} = 0.$$

Новых состояний не появилось, но появились новые переходы:

$$1 \xrightarrow{x_2,x_9,x_{12}} 0,\ 1 \xrightarrow{x_4,x_8,x_{10}} -1,\ 1 \xrightarrow{x_3,x_7,x_{11}} 1,\ 1 \xrightarrow{x_1,x_5} 2.$$

Не вдаваясь в подробности вычислений новых переходов, приведем эти переходы (при необходимости можно самому убедиться в их правильности):

$$0 \xrightarrow{x_{13}} 1,\ 0 \xrightarrow{x_{14}} 0,\ 0 \xrightarrow{x_{15}} -1,\ 0 \xrightarrow{x_{16}} -2,$$

$$2 \xrightarrow{x_{13}} 2,\ 2 \xrightarrow{x_{14}} 1,\ 2 \xrightarrow{x_{15}} 0,\ 2 \xrightarrow{x_{16}} -1,$$

$$-1 \xrightarrow{x_9} -1,\ -1 \xrightarrow{x_{10}} -2,\ -1 \xrightarrow{x_{11}} 0,\ -1 \xrightarrow{x_{12}} -1,$$

$$-2 \xrightarrow{x_{13}} 0,\ -2 \xrightarrow{x_{14}} -1,\ -2 \xrightarrow{x_{15}} -2,\ -2 \xrightarrow{x_{16}} -3,$$

$$-3 \xrightarrow{x_9} -2,\ -3 \xrightarrow{x_{10}} -3,\ -3 \xrightarrow{x_{11}} -1,\ -3 \xrightarrow{x_{12}} -2.$$

Как видим, новое состояние -3 появляется при генерации переходов для состояния -2. Окончательный вид искомого автомата приведен ниже на рисунке.

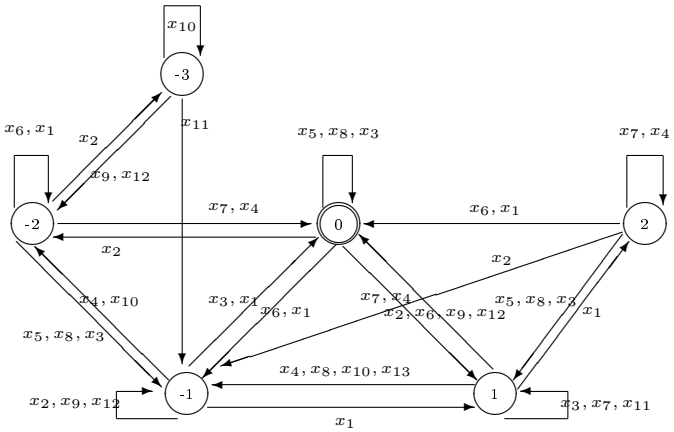


Рис. 13.1.2. Автомат, акцептирующий решения ЛНДУ $x + 2y + u = 3z + 1$ ♠

Рассмотрим еще два вида автоматов, которые акцептируют решения формул вида $\neg C(x_1, x_2, \ldots, x_q)$ и $\exists x_i C(x_1, x_2, \ldots, x_q)$.

Представление решений формулы $\neg C(x_1, x_2, \ldots, x_q)$. Автомат, который акцептирует решения формулы $\neg C(x_1, x_2, \ldots, x_q)$, строится по автомату $\mathcal{A}_{C(x_1,x_2,\ldots,x_q)}$ исходя из такого замечания. Если язык $L \subseteq F(X)$ акцептируется автоматом $\mathcal{A}_C$, то дополнение этого языка $F(X) \setminus L$ акцептируется автоматом $\mathcal{A}'_C$, который отличается от автомата $\mathcal{A}_C$ только множеством заключительных состояний, которое равно дополнению множества заключительных состояний автомата $\mathcal{A}_C$. Следовательно, никаких дополнительных построений выполнять нет надобности, если построен автомат $\mathcal{A}_C$.

Представление решений формулы $\exists x_i C(x_1, x_2, \ldots, x_q)$. Заметим, что формула $\forall x C$ логически эквивалентна формуле $\neg\exists x \neg C$, поэтому рассмотрим представление только для формулы $\exists x C$. Если автомат $\mathcal{A}_{C(x_1,x_2,\ldots,x_q)}$, акцептирующий решения формулы $C(x_1, x_2, \ldots, x_q)$, построен, то построение автомата $\mathcal{A}_{\exists x_i\ C(x_1,x_2,\ldots,x_q)}$ сводится к тому, что координата x_i может быть удалена из множества символов входного алфавита этого автомата.

Функция переходов автомата, акцептирующего решения формулы $\exists x\ (x + 2y = 3z + 1)$ показана ниже на рисунке, а сам автомат принимает вид: $\mathcal{A} = (\{1, 0, 2, -1, -2\}, X = \{x_1^T = (0, 1), x_2^T = (0, 0), x_3^T = (1, 1), x_4^T = (1, 0), f, 1, F = \{0\})$.

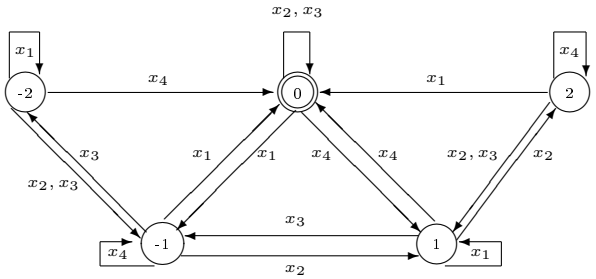


Рис. 13.1.3. Автомат $\mathcal{A}$, акцептирующий решения $\exists z\ x + 2y = 3z + 1$

Заметим, что в общем случае автомат, полученный таким способом не будет детерми-

нированным. Поэтому, после построения автомата, в случае необходимости, к нему нужно применить алгоритм детерминизации X-автоматов.

13.3 Унификация в теориях первого порядка

Проблема ассоциативно-коммутативной унификации (AK-унификации) возникает в задачах автоматизации поиска доказательств теорем в логике предикатов первого порядка. Задача построения алгоритмов AK-унификации сводится к задаче построения базиса решений СЛОДУ и СЛНДУ в множестве натуральных чисел. Вся терминология, определения и результаты, описываемые в этом подразделе, взяты из работы [21].

13.3.1. Общие сведения из теории унификации

Пусть Ω – сигнатура функциональных символов фиксированной арности, V – счетное множество переменных. Множество Ω-термов над алфавитом V обозначается через $T(\Omega, V)$, а $E \subseteq T(\Omega, V) \times T(\Omega, V)$ означает множество тождественных соотношений на множестве $T(\Omega, V)$, которое определяет эквациональную теорию $=_E$. Множество соотношений E называется представлением теории $=_E$. Как известно, отношение $=_E$ является отношением конгруэнтности, а фактор-алгебра по отношению $=_E$ – $T(\Omega, V)/ =_E$ – представляет E-свободную алгебру с множеством образующих V.

Пример 34. Пусть сигнатура Ω состоит из единственного бинарного функционального символа f. Множество тождеств $A = \{f(x, f(y, z)) = f(f(x, y), z)\}$ определяет теорию полугрупп. Очевидно, что $=_A$-классы могут рассматриваться как слова в алфавите V, и A-свободная алгебра $T(\Omega, V)/ =_A$ изоморфна свободной полугруппе $F(V)$ слов в алфавите V. ♠

Напомним, что **подстановкой** называется такое отображение $\theta : V \to T(\Omega, V)$, что $\theta(x) \neq x$ для конечного числа элементов x из V. Поскольку $T(\Omega, V)$ – свободная Ω-алгебра с образующими V, то отображение θ единственным способом продолжается до гомоморфизма (эндоморфизма) $\theta : T(\Omega, V) \to T(\Omega, V)$ таким образом:

$$\theta(t(x_1, \ldots, x_n)) = t(\theta(x_1), \ldots, \theta(x_n)).$$

Исходя из определения подстановки, ее можно задавать в виде:

$$\theta = \{x_1 \to t_1, \ldots, x_k \to t_k\}.$$

С каждой подстановкой $\theta = \{x_1 \to t_1, ..., x_k \to t_k\}$ связываются такие множества: пусть $Var(t_1, \ldots, t_k)$ означает множество переменных, которые входят в термы $t_1, \ldots, t_k$, тогда $DOM(\theta) = \{x \in V | \theta(x) \neq x\} = \{x_1, \ldots, x_k\}$;

$COD(\theta) = \{\theta(x) | x \in DOM(\theta)\} = \{t_1, \ldots, t_k\}$;

$VCOD(\theta) = V(COD(\theta)) = \{x \in V \mid x \in Var(t_1, \ldots, t_k)\}$;

$CCOD(\theta) = \{c_{i_j} | c_{i_j}$ – множество констант, входящих в $t_i, i = 1, 2, \ldots, k\}$.

Если $VCOD(\theta) = \emptyset$, то подстановка θ называется *основной (ground)* подстановкой.

Произведением $\sigma * \tau$ двух подстановок σ и τ называется суперпозиция двух отображений, т. е. $\sigma * \tau(t) = \sigma(\tau(t))$.

Через ε и Λ обозначим соответственно тождественную и нигде не определенную подстановки, которые удовлетворяют таким соотношениям: $\sigma * \varepsilon = \varepsilon * \sigma = \sigma$ и $\Lambda * \sigma = \sigma * \Lambda = \Lambda$ для любой подстановки σ.

Проблема унификации термов s и t в данной E-теории состоит в том, чтобы найти такую подстановку θ, что $\theta(s) =_E \theta(t)$. Если такая подстановка θ существует, то она называется E-унификатором термов s и t. Множество всех E-унификаторов термов s и t обозначают через $U_E(s,t)$.

Примеры унификаторов в эквациональных теориях. Пусть $\Omega = \{f, a\}$, где f – бинарный функциональный символ, a – константа (нульарный функциональный символ) и $s = f(x,\ a), t = f(a,y)$ – термы из $T(\Omega, V)$, где $V = \{x, y\}$.

а) $E = \emptyset$. В этом случае $\theta = \{x \to a,\ y \to a\}$ – единственный $\emptyset$-унификатор термов s и t.

б) $E = K = \{f(x,y) = f(y,x)\}$. Очевидно, что θ также является K-унификатором для s и t. Но поскольку символ f коммутативный, то существует другой K-унификатор $\sigma = \{x \to y\}$. В самом деле, вследствие коммутативности имеем $s = f(x,a)$ и $t = f(a,y) = f(y,a)$ и $\sigma = \{x \to y\}$ дает нам такое тождество: $\sigma(s) = f(y,a) = f(a,y) = f(y,a) = \sigma(t)$. ♠

Когда даны две подстановки $\sigma = \{x_1 \to t_1, \ldots\ldots, x_k \to t_k\}$ и $\theta = \{y_1 \to q_1, \ldots, y_m \to t_m\}$, то их *произведение* $\sigma * \theta$ *находим по таким правилам:*

а) строим множество

$$\{x_1 \to \theta(t_1), \ldots, x_k \to \theta(t_k); y_1 \to q_1, \ldots, y_m \to q_m\},$$

б) из этого множества удаляем пары такого вида

б1) $x_i \to y_j = \theta(t_j)$ *i* $y_j \to x_i$;

б2) и элементы $y_i \to q_i$, если $y_i \in \{x_1, \ldots, x_k\}$.

Например, если θ и σ – подстановки с предыдущего примера и $\theta_1 = \{y \to a\}$, то

$$\theta = \sigma * \theta_1 = \{x \to \theta_1(y); y \to a\} = \{x \to a, y \to a\} = \sigma * \{y \to a\}.$$

В большинстве применений нет необходимости иметь все E-унификаторы, а достаточно иметь полное множество E-унификаторов. Полное множество E-унификаторов определяется с помощью отношения квазипорядка $\leq_E [W]$ на множестве подстановок, где W – конечное множество переменных. Этот квазипорядок определяется таким образом:

$$\sigma \leq_E \theta[W] \Leftrightarrow \exists \lambda : \theta(x) =_E \lambda(\sigma(x)),\ \forall\ x \in W,$$

где λ – некоторая подходящая подстановка.

Если $\sigma \leq_E \theta[W]$, то говорят, что σ является более общим унификатором, чем θ, на множестве W или что унификатор θ является примером унификатора σ. Очевидно, что когда σ является E-унификатором термов s и t, то θ тоже E-унификатор этих термов.

Подстановки σ и τ называются E-эквивалентными ($\sigma =_E \tau[W]$) на W, если $\sigma \leq_E \tau[W] \wedge \tau \leq_E \sigma[W]$.

Полным множеством $cU_E(s,t)$ E-унификаторов термов s и t называется множество унификаторов, которое удовлетворяет таким условиям:

1 : $cU_E(s,t) \subseteq U_E(s,t)$ **(корректность)**;

2 :$\forall \theta \in U_E(s,t) \exists \sigma \in cU_E(s,t) : \sigma \leq_E \theta[W]$, где W – множество переменных, входящих в s и t **(полнота)**.

Понятно, что с точки зрения эффективности это множество должно быть настолько малым, насколько это возможно. Тогда интерес представляет минимальное полное множество $\mu U_E(s,t)$ E-унификаторов для s и t, которое удовлетворяет дополнительному условию

3 : $\forall \sigma, \theta \in \mu U_E(s,t)(\sigma \leq_E \theta[W])$, следует $\sigma = \theta$ **(минимальность)**.

Пример 35. Пусть термы s и t такие как в примере 33, но E-теория теперь будет A-теорией, т. е. $A = \{f(x, f(y,z)) = f(f(x,y),z)\}$.

Подстановки $\sigma = \{x \to a, y \to a\}$ и $\tau = \{x \to f(a,z), y \to f(z,a)\}$ являются A-унификаторами термов s и t и, как легко видеть, множество $\{\sigma, \tau\}$ – полное множество A-унификаторов термов s и t. Добавим, что σ и τ не сравнимы между собой, и это означает что $\{\sigma, \tau\}$ – минимальное полное множество A-унификаторов для s и t.

Если рассматривается A-унификацию для термов $f(x,a)$ и $f(a,x)$, то подстановки $\sigma_1 = \{x \to a\}, \sigma_2 = \{x \to f(a,a)\}, \sigma_3 = \{x \to f(a, f(a,a))\}, \ldots$ дают бесконечное минимальное полное множество A-унификаторов для этих термов. ♠

13.3.2. AK1-унификация

Пусть $T(\Omega, V)$ – коммутативный моноид. Это значит, что когда $V = \{v_1, v_2, \ldots\}$ – множество переменных и $C = \{c_1, c_2, \ldots\}$ – множество констант, то $\Omega = \{\cdot, e\} \cup C$, где $\cdot$ – бинарная операция умножения, а e – нульарная операция – единица моноида и

$$E = \{xy = yx, x(yz) = (xy)z, xe = x\}.$$

Следовательно, имеем $AK1$-теорию и проблема $AK1$-унификации сводится к поиску минимального полного множества унификаторов термов

$$s = v_1^{p_1} v_2^{p_2} \ldots v_k^{p_k} c_1^{q_1} \ldots c_i^{q_i},\ t = v_{k+1}^{p_{k+1}} v_{k+2}^{p_{k+2}} \ldots v_M^{p_M} c_{i+1}^{q_{i+1}} \ldots c_L^{q_L},$$

где $v_i \in V, c_j \in C, p_i, q_j \in \mathcal{N}, i = 1,2,\ldots,M, j = 1,2,\ldots,L$. Прежде чем приступить к описанию алгоритма $AK1$-унификации, рассмотрим основные его детали на примере.

Пример 36. Пусть $s = x^2yac$ і $t = zb^2c$. Допустим, что $\sigma = \{x \to vb, y \to u, z \to uv^2a\}$, где u, v – новые переменные, является унификатором термов s и t. Тогда, применяя подстановку σ к данным термам, имеем такие равенства для переменных u, v и констант a и b соответственно: $\sigma(s) = v^2b^2uac = \sigma(t) = uv^2ab^2c$, где соответствующие показатели степеней при переменных u и v являются решениями таких уравнений:

$$2m_1 + n_1 - k_1 = 0 \tag{48}$$

для переменных,

$$2m_3 + n_3 - k_3 = 1 \tag{49}$$

для константы a,

$$2m_4 + n_4 - k_4 = 2 \tag{50}$$

для константы b, а для константы c уравнения нет, поскольку любой унификатор термов $s' = x^2ya$ и $t' = zb^2$ является также унификатором термов s и t.

Заметим, что для переменных (независимо от их количества) получаем одинаковые уравнения, а для констант эти уравнения разные, хотя матрица систем одна и та же.

Решений уравнений (48) – (50) в множестве натуральных чисел может и не существовать, но если они существуют, то на множестве этих решений вводят покоординатный частичный порядок

$$(a_1, \cdots, a_q) \le (b_1, \cdots, b_q) \Leftrightarrow (\forall i \in [1,q]\ a_i \le b_i,$$

а базис множества решений составляют минимальные решения относительно этого порядка. В данном случае базис множества решений уравнения (48) составляют такие векторы:

$$m_1 = (0,1,1),\ m_2 = (1,0,2).$$

Базисом множества решений уравнений (49) и (50) являются векторы:

$$S_a = \{n_1 = (0,0,1)\},\ S_b = \{n_2 = (1,0,0), n_3 = (0,2,0)\}.$$

Комбинируя векторы из S_a и S_b с векторами m_1 и m_2, находим унификаторы, которые соответствуют минимальному множеству $AK1$-унификаторов термов s и t.

$$S_a \times S_b = \{[(0,0,1),(0,2,0)]; [(0,0,1),(1,0,0)]\} = \{(n_1,n_2); (n_1,n_3)\};$$

$[m_1, m_2, n_1, n_2] =$

	x	y	z
u_1	0	1	1
v_1	1	0	2
a	0	0	1
b	1	0	0

$[m_1, m_2, n_1, n_3] =$

	x	y	z
u_2	0	1	1
v_2	1	0	2
a	0	0	1
b	0	2	0

Отсюда получаем такое множество унификаторов: $\mu U_{AK1}(s,t) = \{\sigma_1 = \{x \to v_1 b, y \to u_1, z \to u_1 v_1^2 a\}, \sigma_2 = \{x \to v_2, y \to u_2 b^2, z \to u_2 v_2^2 a\}\}$. ♠

Рассмотрим теперь общий случай $AK1$-унификации. Для термов s и t, вид которых был приведен выше,

1) составляем уравнение вида

$$p_1 m_1 + p_2 m_2 + \ldots + p_k m_k = p_{k+1} m_{k+1} + \ldots + p_M m_M \tag{51}$$

для переменных и

2) уравнения для каждой константы $c_j, j = 1, 2, \ldots, L$,

$$p_1 n_{j1} + p_2 n_{j2} + \ldots + p_k n_{jk} = p_{k+1} n_{jk+1} + \ldots + p_M n_{jM} + d_j. \tag{52}$$

Базисные решения уравнений (51) и (52) находятся при помощи двух функций:

Function DIOHOM(s,t)

шаг 1: Составить однородное уравнение для переменных

$$p_1 m_1 + \cdots + p_k m_k = p_{k+1} m_{k+1} + \cdots + p_M m_M; \qquad \text{(i)}$$

шаг 2: Решить (i).

выход: Множество $\{e_1, \cdots, e_n\}$ – минимальных линейно независимых M-мерных векторов, которые составляют базис пространства решений (i), или 0, если (i) не имеет решений.

End of DIOHOM

Function DIOINHOM(s,t,c_j).

шаг 1: Составить неоднородное уравнение для константы c_j

$$p_1 n_{j1} + \cdots + p_k n_{jk} = p_{k+1} n_{jk+1} + \cdots + p_M n_{jM} + q_j; \qquad \text{(ii)}$$

шаг 2: Решить (ii).

выход: Множество $S_{c_j} = \{e_1, \cdots, e_k\}$ – минимальных линейно независимых M-мерных векторов, которые составляют базис пространства решений (ii), или $\emptyset$, если (ii) не имеет решений.

End of DIOINHOM

Остальную работу и всю ее организацию выполняет функция $AK1$-UNIFY [21]:

Function $AK1$-**UNIFY**(s,t)

шаг 0: $G1 := (s', t')$, де s', t' – термы без общих переменных и констант;
шаг 1: $\{m_1, \ldots, m_n\} :=$ **DIOHOM**$(G1)$;
шаг 2: if $L = 0$ then $\mu U_E(s,t) :=$ **CONSTRUCT**$(m_1, \cdots, m_n)$ else
шаг 3: ($PRODUCT :=$ **DIOINHOM**$(G1, S_{c_1}) \times \cdots \times$ **DIOINHOM**$(G1, S_{c_L})$;
$\mu U_{AK1}(s,t) := \emptyset$;
forall $(n_1, n_2, \cdots, n_L) \in$ PRODUCT do
$\mu U_{AK1}(s,t) := \mu U_{AK1}(s,t) \cup \{$**CONSTRUCT**$(m_1, \cdots, m_n, n_1, n_2, \cdots, n_L)\}$;
od);
выход: Множество наиболее общих унификаторов $\mu U_{AK1}(s,t)$.

End of $AK1$-**UNIFY**

Алгоритм $AK1$-UNIFY использует две функции: $\times$ – функцию вычисления декартового произведения множеств и $CONSTRUCT$ – функцию, которая берет $M + L$ векторов как входные данные, преобразует их в матрицу

$$\begin{bmatrix} m_1 \\ \cdot \\ \cdot \\ m_M \\ n_1 \\ \cdot \\ \cdot \\ n_L \end{bmatrix}$$

и вычисляет по ней унификатор.

Понятно, что $\mu U_{AK1}(s,t)$ – конечное множество и, следовательно, $AK1$-теория финитарная для элементарной $AK1$-унификации.

Следующее утверждение характеризует алгоритм $AK1$-UNIFY.

Теорема 44. *Алгоритм $AK1$-$UNIFY$ заканчивает свою работу, а множество $\mu U_{AK1}(s,t)$ – корректное, полное и является минимальным полным множеством $AK1$-унификаторов термов s и t.*

Унификация в коммутативных полугруппах. В этом случае $T(\Omega, V)$ не имеет единицы – абелева полугруппа. Имеет место

Лемма 12. *Унификатор $\sigma \in \mu U_{AK}(s,t)$ тогда и только тогда, когда $\sigma \in U_{AK1}(s,t)$ и $e \notin CCOD(\sigma)$, где $CCOD(\sigma)$ – множество констант, входящих в термы из σ.*

Следовательно, для построения множества $\mu U_{AK}(s,t)$ из множества $\mu U_{AK1}(s,t)$, к множеству $\mu U_{AK1}(s,t)$ нужно добавить те $AK1$-унификаторы, которые являются AK-унификаторами. Для этого, как вытекает из леммы 12, необходимо рассмотреть все $AK1$-примеры наиболее общих $AK1$-унификаторов и удалить те из них, которые содержат единицу. Иными словами, $\mu U_{AK}(s,t) = = \{\tau | \tau = (\sigma_i * \lambda)_N,\ e \notin COD(\tau) \ \wedge \ \sigma_i \in \mu U_{AK1}(s,t)\}$, где $\lambda = \{x_{i1} \to e, \ldots, x_{in} \to e\}, i = 1,2,\ldots,n, \{x_{i1}, \ldots, x_{in}\}$ – подмножество переменных из $VCOD(\sigma_i)$ и $(\sigma_i * \lambda)_N$ – подстановка, получена из $\sigma_i * \lambda$ путем удаления всех единиц и ограничена на область $W = Var(s,t)$.

Характеристику полученного множества AK-унификаторов дает

Теорема 45. *Множество $\mu U_{AK}(s,t)$, полученное выше: является корректным, полным и минимальным множеством AK-унификаторов термов s и t.*

Пример 37. Для термов s и t, рассмотренных в предыдущем примере, было найдено $\mu U_{AK1}(s,t) = \{\sigma_1 = \{x \to v_1 b, y \to u_1, z \to u_1 v_1^2 a\}, \sigma_2 = \{x \to v_2, y \to u_2 b^2,\ z \to u_2 v_2^2 a\}\}$.

Для σ_1 имеем такое множество переменных: а) $\emptyset$, б) $\{v_1\}$, в) $\{u_1, v_1\}$, г) $\{u_1\}$.

Соответствующие подстановки имеют такой вид:

а) $\lambda_1 = \varepsilon$ – тождественная подстановка, т. е. $\sigma_1 * \lambda = \sigma_1$ – AK-унификатор;

б) $\lambda_2 = \{v_1 \to e\}$;

в) $\lambda_3 = \{u_1 \to e, v_1 \to e\}$;

г) $\lambda_4 = \{u_1 \to e\}$.

Следовательно, имеем такие подстановки:

б) $\sigma_{12} = \sigma_1 * \lambda_2 = \{x \to b, y \to u_1, z \to u_1 a\}$ – AK-унификатор,

в) $\sigma_{13} = \sigma_1 * \lambda_3 = \{x \to b, y \to e, z \to a\}$ – удаляется,

г) $\sigma_{14} = \sigma_1 * \lambda_4 = \{x \to v_1, y \to e, z \to v_1^2 a\}$ – удаляется.

Аналогично, только для σ_2 и ее множества переменных получаем AK-унификаторы $\sigma_2, \sigma_{24} = = \{x \to v_2, y \to b^2, z \to v_2^2 a\}$.

В результате находим $\mu U_{AK}(s,t) = \{\sigma_1, \sigma_2, \sigma_{12}, \sigma_{24}\}$. ♠

Невзирая на то, что проблема AK-унификации имеет финитарный тип, множество $\mu U_{AK}(s,t)$ может быть потенциально бесконечным (комбинаторный взрыв). Рассмотрим

пример, иллюстрирующий такую возможность.

Пример 38. Пусть дано уравнение вида

$$mx_1 + mx_2 + \cdots + mx_p = ny_1 + ny_2 + \cdots + ny_q, \tag{53}$$

где $+ \in \Omega_{AK}$ и НОД$(m,n) = 1$.

Если $(x,y) = (x_1, \ldots, x_p, y_1, \ldots, y_q)$ является решением (53), то

$$m \text{ должно делиться на } y_1 + \cdots + y_q, \text{ и } n \text{ должно делиться на } x_1 + \cdots + x_p.$$

Более того, если (x,y) – минимальное решение (53), то

$$x_1 + \cdots + x_p = n, \tag{54}$$

$$y_1 + \cdots + y_q = m. \tag{55}$$

Из комбинаторики известно, что уравнение (54) имеет C^n_{n+p-1} решений над $\mathcal{N}^p$. Следовательно, справедлива

Теорема 46. *a) Уравнения* (54)-(55) *имеют* $C^n_{n+p-1} \cdot C^m_{m+p-1}$ *минимальных решений над* $\mathcal{N}^{p+q}$.

b) Число минимальных унификаторов термов $s = mx_1 + \cdots + mx_p$ *и* $t = ny_1 + \cdots + ny_q$ *равняется*

$$(-1)^{p+q} \sum_{i=0}^{p} \sum_{j=0}^{q} (-1)^{i+j} C^i_p \cdot C^j_q \cdot 2^{C^n_{n+j-1} \cdot C^m_{m+i-1}}.$$

Например, согласно приведенным теоремам, термы $3x$ и $y_1 + y_2 + y_3 + y_4$ имеют 1044599 минимальных AK-унификаторов, а термы $4x$ и $y_1 + y_2 + y_3 + y_4$ имеют 34359607481 минимальных AK-унификаторов. ♠

13.4 Анализ множества дизъюнктов

Приведем одну из модификаций TSS-алгоритма для решения задачи проверки противоречивости множества дизъюнктов в исчислении высказываний. Эта модификация хороша тем, что ее применение дает возможность не только проверять противоречивость, но и определять минимальные противоречивые подмножества дизъюнктов.

Наиболее эффективным в данном случае методом определения противоречивости множества дизъюнктов является метод резолюций. Рассмотрим способ сведения задачи определения противоречивости множества дизъюнктов по методу резолюций к решению СЛОДУ.

Пусть дано множество дизъюнктов $S = \{D_1, D_2, \ldots, D_n\}$ над упорядоченным множеством атомарных формул $X = \{p_1, p_2, \ldots, p_m\}$. Рассмотрим дизъюнкт

$$D_k = q_1 \vee q_2 \vee \ldots \vee q_m,$$

где $D_k \in S$, а q_i – литералы над множеством X. Построим по этому дизъюнкту линейное выражение

$$a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{km}x_m,$$

где

$$a_{ki} = \begin{cases} 1, & \text{если } q_i = p \in X; \\ -1, & \text{если } q_i = \neg p, p \in X; \\ 0, & \text{если } q_i \text{ не входит в } D_k. \end{cases}$$

. Тогда множеству S будет соответствовать матрица A_S. Например, для множества дизъюнктов $S = \{\neg p \vee r, \neg r \vee b \vee d, \neg d \vee s, \neg s \vee r \vee p\}$, где $X = \{p, r, b, d, s\}$ получаем матрицу

$$A_S = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

Построим систему линейных диофантовых уравнений (СЛОДУ) $A_S^T x = 0$, где A_S^T – матрица, транспонированная к матрице A_S. Тогда определение противоречивости множества дизъюнктов S резолюционным методом сводится к проверке совместности СЛОДУ $A_S^T x = 0$ согласно с таким алгоритмом.

АПС-TSS (S)
Вход: Множество дизъюнктов S.
Выход: ВЫПОЛНИМО или ПРОТИВОРЕЧИВО.
Метод:
начало
1. Построить A_S^T для S;
2. Если в A_S^T есть строки, включающие только 0 и 1 или только 0 и -1, то
   (Удалить в A_S^T все столбики, включающие
   ненулевые элементы этих строк);
3. $i := 1$;
3. Вычислить TSS для $L_i(x) = 0$;
4. Для всех $j = i + 1$ до p выполнить
   Для всех $e \in TSS$ найти значения $L_j(e)$
   кц;
5. Удалить те $e \in TSS$, которые порождают тавтологии;
   (т. е. те векторы, для которых значения $L_j(e) = 0$, причем ненулевые
   координаты e не попадают на нулевые коэффициенты $L_j(x)$)
6. Если $TSS = \emptyset$, то (ВЫПОЛНИМА; на 8);
7. $i := i + 1$; Если $i < p$ то на 4 иначе
   Если $TSS = \emptyset$ то ВЫПОЛНИМА иначе ПРОТИВОРЕЧИВА;
8. СТОП;

конец

Обосновывает этот алгоритм

Теорема 47. *Множество дизъюнктов S противоречиво тогда и только тогда,когда СЛОДУ $A_S^T x = 0$ имеет непустое TSS, которое построено алгоритмом АПС-TSS.*

Доказательство. Если СЛОДУ $A_S^T x = 0$ имеет хотя бы одно ненулевое решение, то по построению дизъюнкты, которые соответствуют ненулевым координатам решения, составляют минимальное противоречивое подмножество дизъюнктов. А это значит, что и все множество S тоже будет противоречивым.

Пусть S противоречиво и $S' = \{D_1, D_2, \ldots, D_k\} \subseteq S$ – его минимальное противоречивое подмножество, где $k \leq m$, $m = |S|$. Среди D_i, $i = 1, 2, \ldots, k$ должна существовать хотя бы одна пара дизъюнктов, резольвента которых нетавтологична. В противном случае получаем противоречие с минимальностью множества S'. Обозначим эту резольвенту через D_{js}. Тогда, в силу тех же причин, в множестве $S' \setminus \{D_j, D_s\} \cup \{D_{js}\}$ должна существовать пара дизъюнктов, резольвента которых нетавтологична и т. д. Действуя таким образом дальше, в конце придем ко пустому дизъюнкту 0 и получим вектор c, который будет решением СЛОДУ $A_S^T x = 0$. ■

Пример 39. Пусть A_S^T соответствует такому множеству дизъюнктов $S = \{p \vee \neg q \vee \vee r, \neg p \vee q \vee r, p \vee q \vee \neg r \vee \neg u, \neg q \vee \neg r \vee u, p \vee r \vee u, \neg p \vee q \vee \neg u\}$

$$A_S^T = \begin{Vmatrix} 1 & -1 & 1 & 0 & 1 & -1 \\ -1 & 1 & 1 & -1 & 0 & 1 \\ 1 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \end{Vmatrix}.$$

Построение TSS для первого уравнения дает такое множество решений (в 2, 3 и 4 столбиках показаны значения соответственно 2, 3 и 4 уравнений, если подставить решения первого уравнения в эти уравнения):

	L_2	L_3	L_4
000100	-1	-1	1
110000	0	-	-
011000	2	0	-
010010	1	2	1
100001	0	-	-
001001	2	-1	-2
000011	1	1	0

.

После чистки получаем такие векторы:

	L_2	L_3	L_4
000100	-1	-1	1
010010	1	2	1
001001	2	-1	-2

.

Комбинирование по значениям второго столбика порождает такие векторы:

	L_3	L_4
010110	1	2
001201	-3	0

.

После чистки получаем единственный вектор:

	L_3	L_4
010110	1	2

.

Дальнейшие комбинации невозможны, поскольку комбинировать нету с чем. Следовательно, исходная СЛОДУ не имеет решений (несовместна) и множество дизъюнктов S непротиворечиво.

б) Пусть A_S^T соответствует такому множеству дизъюнктов $S = \{p, \neg p \vee q \vee r, \neg q \vee c, \neg r \vee c, \neg c\}$

$$A_S^T = \begin{cases} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{cases}.$$

Построение TSS для первого уравнения дает такое множество решений (во 2-м, 3-м и 4-м столбиках показаны значения соответственно 2-го, 3-го и 4-го уравнений, если подставить решения первого уравнения в эти уравнения):

	L_2	L_3	L_4
11000	1	1	0
00100	-1	0	1
00010	0	-1	1
00001	0	0	-1

.

Чистка ничего не удаляет, поскольку все ненулевые значения координат векторов-решений попадают на нулевые коэффициенты уравнений L_2, L_3, и L_4.

Комбинируя по первому столбику, получаем:

	L_3	L_4
11100	1	1
00010	-1	1
00001	0	-1

.

Как и раньше, чистка ничего не удаляет, поскольку все ненулевые значения координат векторов-решений попадают на нулевые коэффициенты уравнений L_3 и L_4.

Комбинируя по первому столбику, получаем:

	L_4
11110	2
00001	-1

.

Окончательно получаем решение $c = (1, 1, 1, 1, 2)$, которое говорит о противоречивости данной системы дизъюнктов.

Поскольку все координаты полученного решения ненулевые, то минимальное противоречивое подмножество дизъюнктов совпадает со всем множеством S.

Пример 40. Рассмотрим пример, взятый из учебника по операционным системам и который можно найти в [48].

Являются ли формулы $P \wedge \neg B$ и $I \wedge \neg N$ следствиями нижеприведенных формул?

$A \to P$	$\neg A \vee P$
$(P \to \neg B) \to D$	$\neg P \vee B \vee D$
$D \to (S \wedge M \wedge H)$	$\neg D \vee S$
	$\neg D \wedge M$
	$\neg D \wedge H$
$H \wedge N \to R$	$\neg H \vee \neg N \vee R$
$H \wedge \neg R \to I$	$\neg H \vee R \vee I$
$A \wedge \neg B \wedge \neg R$	A
	$\neg B$
	$\neg R$

Добавим к этим дизъюнктам (выписанных справа в таблице) отрицания формул $P \wedge \neg B$ и $I \wedge \neg N$, которые после приведения принимают вид, который показан в последних двух строках в таблице.

$A \to P$	$\neg A \vee P$
$(P \to \neg B) \to D$	$\neg P \vee B \vee D$
$D \to (S \wedge M \wedge H)$	$\neg D \vee S$
	$\neg D \wedge M$
	$\neg D \wedge H$
$H \wedge N \to R$	$\neg H \vee \neg N \vee R$
$H \wedge \neg R \to I$	$\neg H \vee R \vee I$
$A \wedge \neg B \wedge \neg R$ P	A
	$\neg B$
	$\neg R$
	$\neg P \vee B$
	$\neg I \vee N$

Применим алгоритм АПС-TSS к полученному множеству дизъюнктов, находим два решения $x_1 = (1,0,0,0,0,0,0,1,1,0,1,0)$, которым соответствует такое минимальное противоречивое подмножество дизъюнктов $\neg A \vee P$, A, $\neg B$, $\neg P \vee B$, и второе решение $x_2 = (1,1,0,0,1,1,1,1,1,1,0,1)$, которому соответствует такое минимальное противоречивое множество дизъюнктов $\neg A \vee P$, A, $\neg B$, $\neg P \vee B \vee D$, $\neg D \vee H$, $\neg H \vee \neg N \vee R$, $\neg H \vee R \vee I$, $\neg R, \neg I \vee N$. В правильности этого решения можно убедиться непосредственной проверкой.

Поскольку отрицание и первой, и второй из формул, которые проверяются, входят в соответствующие противоречивые подмножества, то сами формулы являются логическими следствиями данного множества дизъюнктов.

♠

13.5. Построение дискретных изображений на плоскости

В теории распознавания образов возникают задачи, которые имеют дело с прямоугольным полем, разбитым на квадратные клетки, которые имеют разную раскраску и в совокупности представляют некоторый образ. Обозначим такое поле через F с числом клеток $r = mn$, где m – число горизонтальных последовательностей клеток, а n – число столбиков. Сначала предполагается, что все клетки поля F нераскрашены. Пусть $Q = \{0, 1, \ldots, k-1\}$ означает множество цветов. Построение произвольных изображений выполняется с помощью накладывания на поле F элементарных блоков или шаблонов $S = \{s_1, s_2, \ldots, s_p\}$, имеющих небольшие размеры и определенную раскраску. При этом операция накладывания на одну клетку разных цветов задается как бинарная операция на множестве Q.

Задача распознавания изображений состоит в нахождении такого набора из существующего запаса шаблонов, который при определенном накладывании на поле F составил бы нужное изображение.

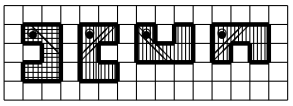
Если в поле F $m = 1$, $n > 1$, то соответсвующая задача называется *задачей о построении линейной мозаики*, а если $m, n > 1$, то такая задача называется *задачей о построении двухмерной мозаики* [3].

Математическая постановка задачи о построении двухмерной мозаики имеет вид.

Сначала определяется операция накладывания цветов. Для наглядности рассмотрим простейший случай, когда эта операция совпадает с операцией сложения двух чисел по модулю $k = 2$. Тогда все шаблоны будут одноцветными, поскольку число 0 означат отсутствие

цвета. Так как блоки можно накладывать любой стороной (внешней и внутренней), а также поворачивать в плоскости на угол кратный 90^o, то проще считать, что существует набор разных шаблонов, который всегда размещаются на плоскости фиксированным способом. Виделим в каждом шаблоне определенную клетку, которую будем называть меткой шаблона. Условимся дальше, что метка – это самая левая среди самых верхних клеток шаблона.

Пример 41. Рассмотрим шаблон, приведенный ниже на рисунке и изображенный в четырех позициях – s_1, s_2, s_3, s_4.



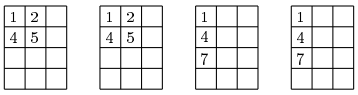
Темный кружочек указывает на метку шаблона. ♠

Занумеруем клетки поля F числами от 1 до $r = mn$ сначала по первой строке, а потом по второй и т. д. Зная координату метки шаблона, можно однозначно разместить его на поле. Если координату шаблона обозначить α, то любой шаблон можно задать его уравнением, которое в общем виде будет выражаться через эту координату:

$s_1(\alpha) = (\alpha, \alpha+1, \alpha+n+1, \alpha+2n, \alpha+2n+1)$;
$s_2(\alpha) = (\alpha, \alpha+1, \alpha+n, \alpha+2n, \alpha+2n+1)$;
$s_3(\alpha) = (\alpha, \alpha+2, \alpha+n, \alpha+n+1, \alpha+n+2)$;
$s_4(\alpha) = (\alpha, \alpha+1, \alpha+2, \alpha+n, \alpha+n+2)$.

В общем случае в зависимости от вида шаблона и величины поля F координата α может принимать только конечное число значений. Множество допустимых значений α определяет множество допустимых положений шаблона (или точнее его метки) на заданном поле F.

Пример 42. Ниже на рисунке приведены допустимые значения координаты α для каждого из шаблонов, показанных на рисунке из примера 40, на поле F размерности $3 \times 4 = 12$. В сумме число положений равняется 14.



$s_1(\alpha)$ $s_2(\alpha)$ $s_3(\alpha)$ $s_4(\alpha)$

Каждому допустимому положению шаблона поставим в соответствие переменную x_j, $j = 1, 2, \ldots, 14$, которая принимает значение 1 или 0 в зависимости от того, принимает или нет данный шаблон участие в построении изображения. Если вместо координаты α подставить для каждого шаблона значения из приведенного рисунка, то каждой переменной будет отвечать кортеж, определяющий номера клеток, на которые накладывается соответствующий шаблон:

$c_1 = (1, 2, 5, 7, 8)$, $c_2 = (2, 3, 6, 8, 9)$, $c_3 = (4, 5, 8, 10, 11)$,
$c_4 = (5, 6, 9, 11, 12)$, $c_5 = (1, 2, 4, 7, 8)$, $c_6 = (2, 3, 5, 8, 9)$,
$c_7 = (4, 5, 7, 10, 11)$, $c_8 = (5, 6, 8, 11, 12)$, $c_9 = (1, 3, 4, 5, 6)$,
$c_{10} = (4, 6, 7, 8, 9)$, $c_{11} = (7, 9, 10, 11, 12)$, $c_{12} = (1, 2, 3, 4, 6)$,
$c_{13} = (4, 5, 6, 7, 9)$, $c_{14} = (7, 8, 9, 10, 12)$. ♠

Обозначим изображение, которое необходимо построить с помощью шаблонов, вектором $b = (b_1, b_2, \ldots, b_r)$, где $b_i \in \{0, 1\}$, $i = 1, 2, \ldots, r = mn$. Как отмечалось выше, каждому допустимому положению шаблона поставим в соответствие переменную x_j, $j = 1, 2, \ldots, r$, которая принимает значения 0 или 1.

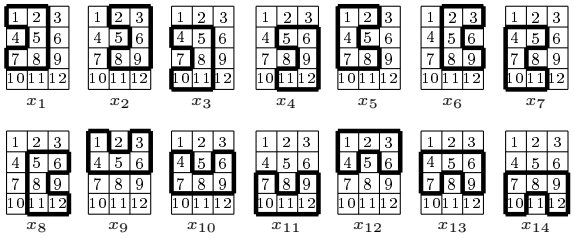
Добавляя к каждой клетке поля F переменные, в кортежи которых входит номер этой

клетки, получаем СЛНДУ вида

$$\sum_{c_i \cap j \neq \emptyset} x_i = b_j \tag{56}$$

в поле $F_2,\ j = 1, 2, \ldots, r.$

Пример 43. Ниже на рисунке показаны допустимые положения шаблонов из примера 41 и указаны переменные, которые их обозначают. В середину шаблона попадают номера клеток, которые определяют каждый кортеж $c_i,\ i = 1, \ldots, 14.$



Теперь, смотря на этот рисунок, легко построить СЛНДУ вида (56) в поле F_2 для наших шаблонов. В i-й строке добавляются переменные, кортежи которых включают клетку с номером i, а в правой части должны стоять $b_j,\ j = 1, 2 \ldots, r.$

$$\begin{cases}
x_1 + x_5 + x_9 + x_{12} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_1, \\
x_1 + x_2 + x_5 + x_6 + x_{12} \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_2, \\
x_2 + x_6 + x_9 + x_{12} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_3, \\
x_3 + x_5 + x_7 + x_9 + x_{10} + x_{12} + x_{13} \ldots \ldots = b_4, \\
x_1 + x_3 + x_4 + x_6 + x_7 + x_8 + x_9 + x_{13} = b_5, \\
x_2 + x_4 + x_8 + x_9 + x_{10} + x_{12} + x_{13} \ldots \ldots = b_6, \\
x_1 + x_5 + x_7 + x_{10} + x_{11} + x_{13} + x_{14} \ldots \ldots = b_7, \\
x_1 + x_2 + x_3 + x_5 + x_6 + x_8 + x_{10} + x_{14} = b_8, \\
x_2 + x_4 + x_6 + x_{10} + x_{11} + x_{13} + x_{14} \ldots \ldots = b_9, \\
x_3 + x_7 + x_{11} + x_{14} \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_{10}, \\
x_3 + x_4 + x_7 + x_8 + x_{11} \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_{11}, \\
x_4 + x_8 + x_{11} + x_{14} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots = b_{12}.
\end{cases}$$

Придавая разные значения координатам b_i вектора b $(i = 1, 2, \ldots, r)$ и решая полученную СЛНДУ, находим комбинации шаблонов, с помощью которых строится нужное изображение. Если СЛНДУ несовместна, то это значит, что с помощью данного набора шаблонов построить нужное изображение невозможно. ♠

13.6. Задачи теории кодирования

Метод решения СЛОДУ в поле F_2 можно использовать в теории кодирования, в частности, при использовании линейных кодов.

Линейные коды. Пусть $X = \{0, 1\}$ – двоичный алфавит, а $V_n = \{(a_1, a_2, \ldots, a_n) : a_i \in X,\ i = 1, 2, \ldots, n\}$ – множество n-мерных векторов в алфавите X.

Весом вектора $p \in V_n$ называется число ненулевых компонент в этом векторе. **Расстоянием Хемминга** между двумя векторами $p = (x_1, x_2, \ldots, x_n)$ и $q = (y_1, y_2, \ldots, y_n)$ называется число пар компонент (x_i, y_i) таких, что $x_i \neq y_i,\ i = 1, 2, \ldots, n,\ p, q \in V_n$. Расстояние Хемминга между векторами p и q обозначается $dist(p, q)$.

Определение 21. *Блочным кодом или просто кодом длины n называется подмножество $C \subseteq V_n$, состоящее из K векторов длины n. Элементы из C называются кодовыми векторами или кодовыми словами. Минимальным расстоянием кода C называется число* $d = \min\{dist(p, q) : p, q \in C, p \neq q\}$.

Определение 22. *Кодом, находящим ошибки, называется пара (C, f), где C – блочный*

код, а f – функция такая, что

$$f(p) = \begin{cases} p, & \text{если} \quad p \in C; \\ t, & \text{если} \quad p \in V_n \setminus C, \end{cases}$$

где t – символ, отличный от 0 и 1.

Кодом, исправляющим ошибки, называется пара (C, f), где C – блочный код, а $f : V_n \to C$ – функция такая, что $f(p) = p$, когда $p \in C$.

Пусть $p = (x_1, \ldots, x_n)$ – кодовое слово. При передаче p по шумовому каналу принятое слово $q = (y_1, \ldots, y_n)$ может отличаться от слова p. Код, находящий ошибки, по слову q определяет, была или нет ошибка при передаче. В случае позитивного ответа передача, как правило, повторяется.

Код, исправляющий ошибки, по слову q должен решить, какое кодовое слово было передано. Чаще всего здесь используется декодирование **по максимальному правдоподобию:** переданным считается слово p, которое отличается от q длины n в наименьшем числе компонент, т. е. считается, что передано слово p, если $d(p, q) = \min\{dist(p, q) : p \in C\}$.

Теорема 48. *а) Для того, чтобы можно было исправлять ошибки в k и меньшем числе компонент (ошибки веса k) необходимо и достаточно, чтобы для любых $p_1, q_1 \in C$ множества $\{p : d(p_1, p) \le k,\ p \in V_n\}$ и $\{q : d(q_1, q) \le k,\ q \in V_n\}$ не имели общих элементов.*

b) Пусть $D(p_1) = \{p : d(p_1, p) \le k,\ p \in V_n\}$ и $D(q_1) = \{p : d(q_1, q) \le k,\ q \in V_n\}$. Тогда $D(p_1) \cap D(q_1) = \emptyset$ тогда и только тогда, когда $d(p_1, q_1) \le 2k + 1$.

Линейные коды являются одними из важнейших в теории кодирования. Это связано с удобствами в процессе нахождения и исправления ошибок, а также с возможностью компактного задания кода и с простыми алгоритмами декодирования.

Из определения множества V_n вытекает, что это множество является векторным пространством размерности n над множеством $\{0, 1\}$, т. е. $V_n = \{0, 1\}^n$.

Определение 23. *Линейным кодом длины n с k информационными символами ((n,k)-кодом) называется подпространство C размерности k пространства V_n.*

При задании линейного кода удобно пользоваться матрицей G, которая называется **порождающей матрицей кода**. Построение матрицы G выполняется путем выбора в $(n-k)$-коде C некоторого базиса $e_1, e_2, \ldots, e_k$, т. е. $G = (e_1, e_2, \ldots, e_k)^T$ размерности $k \times n$.

Если $p = (a_1, \ldots, a_k)$ – сообщение длины $k, a_i \in \{0, 1\}$, а G – порождающая матрица, то кодовое слово a, соответствующее сообщению p, можно выбрать таким образом:

$$a = p \cdot G = (a_1, \ldots, a_k) \cdot \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ a_{21} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots \\ a_{k1} & \ldots & a_{kn} \end{pmatrix}.$$

Пример 44 Пусть имеем порождающую матрицу

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

(7, 4) линейного кода. При помощи этой матрицы можно закодировать все двоичные сообщения длины 4 (их будет 16). Например, если $p = (0101)$, то соответствующее кодовое слово равняется

$$a = p \cdot G = (0101) \cdot \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = (0100101).\spadesuit$$

Рассмотрим способ проверки линейного кода на наличие ошибок. Для линейного кода C через C_D обозначим множество векторов из V_n, ортогональных к векторам из C, т. е.

$$C_D = \{u = (u_1, \ldots, u_n) \in V_n : \forall v = (v_1, \ldots, v_n) \in C \; u * v = 0\},$$

где $*$ означает скалярное произведение векторов. Пространство C_D является подпространством размерности $n - k$ пространства V_n и называется **двойственным** или **дуальным** коду C.

Порождающая матрица H дуального кода называется **матрицей проверки** кода C. Из определения матрицы H вытекает способ ее построения. Действительно, по определению $H \cdot v^T = 0$ тогда и только тогда, когда $v \in C$. В частности, если векторы v взять в качестве базисных векторов кода C, то получаем $H \cdot G^T = 0$. Таким образом, построение матрицы H сводится к решению СЛОДУ в поле F_2 – поле вычетов по модулю 2.

Пример 45. Найти матрицу проверки линейного кода, если его порождающая матрица имеет вид

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Из сказанного, вытекает, что $u \in C_D$, если $u \cdot G^T = 0$. Получаем СЛОДУ вида

$$(u_1, u_2, \ldots, u_7) \cdot \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = 0.$$

Решая ее TSS-методом, находим такие базисные векторы-решения этой системы:

$$p = (0110011), \; q = (1010101), \; r = (1101001),$$

т. е.

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Например, для кодового слова $a = (0100101)$ (см. пример 44), получаем:

$$H \cdot a^T = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = (000),$$

а это значит, что кодирование и передача слова (0101) были выполнены без ошибок. ♠

Список литературы

[1] *Гильберт Д.* Проблемы Гильберта. – М: Наука. – 1969. – 53 с.

[2] *Донец Г. А.* Решение задачи о сейфе на (0,1)-матрицах. *ж. Кибернетика и системный анализ. – 2002. – N 1. – С. 98–105.*

[3] *Донец Г. А., Самер И. М. Альшаламе.* Решение задачи о построении линейной мозаики. *Теория оптимальных решений. – К.: Ин-т кибернетики им. В. М. Глушкова НАН Украины. – 2005. – С. 15 – 24.*

[4] Компьютерная алгебра (Символьные и алгебраические вычисления). – Под ред. Бухбергера Б., Коллинза Дж., Лооса Р. – М.: Мир, – 1986. – 386 с.

[5] *Крывый С. Л.* Критерий совместности систем линейных диофантовых уравнений над множеством натуральных чисел. *ж. Допов. НАНУ. – 1999. – N 5. – С. 107-112.*

[6] *Крывый С.Л.* О некоторых методах решения и критериях совместности систем линейных диофантовых уравнений в области натуральных чисел. *ж. Кибернетика и системный анализ. – 1999. – N 4. – С.12 – 36.*

[7] *Крывый С. Л.* Алгоритмы решения систем линейных диофантовых уравнений в целочисленных областях. *ж. Кибернетика и системный анализ. – 2006. – N 2. – С. 3 – 17.*

[8] *Крывый С. Л.* Алгоритмы решения систем линейных диофантовых уравнений в полях вычетов. *ж. Кибернетика и системный анализ. – 2007. – N 2. – С. 15 – 23.*

[9] *Крывый С. Л.* Алгоритмы решения систем линейных диофантовых уравнений в кольце вычетов. *ж. Кибернетика и системный анализ. – 2007. – N 6. – С. 27 – 40.*

[10] *Крывый С.Л.* Об алгоритмах решения систем линейных диофантовых ограничений в области $\{0, 1\}$. – *ж. Кибернетика и системный анализ. – 2003. – N 5. – С. 58–69.*

[11] *Крывый С. Л.* Алгоритмы построения базиса множества решений систем линейных диофантовых уравнений в кольце целых чисел. *ж. Кибернетика и системный анализ. – 2009. – N 6. – С. 36 – 41.*

[12] *Крывый С.Л.* Комбинаторный метод решения систем линейных ограничений. *ж. Кибернетика и системный анализ. – 2014. – N 4. – С.14 – 26.*

[13] *Капитонова Ю.В., Кривой С. Л., Летичевский А. А., Луцкий Г.М.* Лекции по дискретной математике. – Санкт-Петербург: БХВ-Петербург. – 2004. – 634 с.

[14] *Матиясевич Ю. В.* Диофантовость перечислимых множеств. – ДАН СССР. – 1970 . – 191. – N 2. – С. 279–282.

[15] *Черемушкин А. В.* Лекции по арифметическим алгоритмам в криптографии. - *М.: МЦНМО. – 2002. – 103 с.*

[16] *Чугаенко А.В.* О реализации TSS-алгоритма. *ж. Управляющие системы и машины. – 2007. – N 3. – С. 14 – 26.*

[17] *Шевцов Г. С.* Линейная алгебра: теория и прикладные аспекты. – *М.: Финансы и статистика. – 2003. – 576 с.*

[18] *Gordan P.* Ueber die Auflösung linearen Gleidungen mit reallen Coefficienten. – *Mathematische Annalen. – 1873. – N 6, – P. 23–28.*

[19] *Hilbert D.* Ueber die Theorie der algebraischen Formen. – *Mathematische Annalen. – 1890. – N 36, – P. 473–534.*

[20] *Allen R., Kennedy K.* Automatic translation of FORTRAN program to vector form. *ACM Transactions on Programming Languages and systems. – 1987. – v. 9. – N 4. – P. 491–542.*

[21] *Baader F., Ziekmann J.* Unification theory. *Handbook of Logic in Artificial Intelligence and Logic Programming. – Oxford University Press. – 1994. – P. 1–85.*

[22] *Bodirsky M., Nesetril J.* Constraint satisfaction with countable homogeneous templates. *Lecture Notes in Computer Science. – Springer. – 2003. – v. 2803. – P. 44–57.*

[23] *Bulatov A.* H-coloring dichotomy revisited. *Theoretical Computer Science. – 2005. – v. 349. – N 1. – PP. 31 – 39.*

[24] *Bulatov A., Krokhin A., Jeavons P.G.* Classifying the complexity of constraints using finite algebras. *SIAM Journ. Computing. – 2005. – v. 34. – N 3. – PP. 720 – 742.*

[25] *Creignou N., Khanna S., Sudan M.* Complexity Classification of Boolean Constraint Satisfaction Problems. *SIAM Monographs on Discrete Mathematics and Applications: Society for Industrial and Applied Mathematics. Philadelphia. PA. – 2001. – v. 7. – 347 p.*

[26] *Drakengren T., Jonsson P.* A complete classification of tractability in Allen's algebra relative to subsets of basic relations. *Artificial Intelligence. – 1998. – v. 106. – PP. 205 – 219.*

[27] *Jeavons P.G.* Constructing constraints. *In Proceed. 4th Intern. Conf. on Constraint Programming - CP'98 (Pisa, October 1998). – 1998. – v. 1520. – Lecture Notes in Comput. Science. : Springer-Verlag. – P. 2 – 16.*

[28] *Jeavons P.G.* On the algebraic strukture of combinatorial problems. *Theoretical Computer Science. – 1998. – v. 200. – PP. 185 – 204.*

[29] *Cohen D.A., Jeavons P.G.* The Complexity of Constraint Languages. *in Handbook of Constraint Programming. – Elsevier. – 2006. – PP. 245 – 280.*

[30] *Jeavons P.G., Cohen D.A., Gyssens M.* Closure properties of constraints. *Journ. of the ACM. – 1997. – v. 44. – PP. 527 – 548.*

[31] *Jeavons P.G., Cohen D.A., Gyssens M.* How to determine the expressive power of constraints. *Constraints. – 1999. – v. 4. – PP. 113 – 131.*

[32] *Krokhin A., Jeavons P.G., Jonsson P.* Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journ. of the ACM. – 2003. – v. 50. – PP. 591 – 640.*

[33] *Krokhin A., Jeavons P.G., Jonsson P.* Constraint satisfaction problems on intervals and lengths. *SIAM Journ. on Discrete Mathematics. – 2004. – v. 17. – PP. 453 – 477.*

[34] *Ladner R.E.* On the structure of polinovial time reducibility. *Journ. of the ACM. – N 22. – 1975. – PP. 155 – 171.*

[35] *Nebel B., Bürkert J.* Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. *Journal of the ACM. – 1995. – v. 42. – PP. 43 – 66.*

[36] *Renz J., Nebl B.* On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus. *Artificial Intelligence. – 1999. – v.108. – PP. 69 – 123.*

[37] *Cooper M. C., Cohen D.A., Jeavons P.G.* Caracterising tractable constraints. *Artificial Intelligence. – 1994. – v. 65. – PP. 347 – 361.*

[38] *Schaefer T. J.* The Complexity of satisfiability problems. *In Propc. 10-th ACM Symposium on Theory of Computing, STOC'78. – 1978. – PP. 216 – 226.*

[39] *Papadimitriou C.H.* Computational complexity. *Addison-Wesley. – 1994. – 462 p.*

[40] *Pöschel R., Kalužnin L.A.* Funktionen und Relationenalgebren. *DVW. Berlin. – 1979. – 262 p.*

[41] *Perrin D.* Finite automata. In Handbook of Theoretical Computer Science. – vol. 2. – Elsevier. – 1990. – P. 1–58.

[42] *Post E.L.* The two-valued iterative systems of mathematical logic. *Annals Mathematical Studies. – Princeton University Press. – 1941. – 26 p.*

[43] *Szendrei A.* Clones in universal algebras. *Seminares de Mathematigues Superieures. University of Montreal. – 1986. – PP. 253 – 262.*

[44] *Romeuf J. F.* A polinomial Algorithm for Solvin systems of two linear Diophantine equations. *Theoretical Computer Science. – 1990. – 74. – N 3. – P.329–340.*

[45] *Clausen M., Fortenbacher A.* Efficient solution of linear diophantine equations. – *Journ. Symbolic Computation. – 1989. – v.8. – N 1,2. – P. 201–216.*

[46] *Comon H.* Constraint solving on terms: Automata techniques (Preliminary lecture notes). – *Intern. Summer School on Constraints in Computational Logics: Gif-sur-Yvette, France, September 5–8. – 1999. – 22 p.*

[47] *Contenjean E., Devie H.* An efficient algorithm for solving systems of diophantine equations. – *Inform. Comput. – 1994. – N 113. – v. 1. – P. 143–172.*

[48] *Contejean E., Ajili F.* Avoiding slack variables in the solving of linear diophantine equations and inequations. – *Theoretical Computer Science. – 1997. – v. 173. – P. 183–208.*

[49] *Domenjoud E.* Outils pour la deduction automatique dans les theories associatives-commutatives. – *Thesis de Doctorat d'Universite: Universite de Nancy I. – 1991.*

[50] *Filgueiras M., Tomas A.P.* A Fast Method for Finding the Basis of Non-negative Solutions to a Linear Diophantine Equation. – *Journ. Symbolic Computation. – 1995. – v.19. – N 2. – P. 507–526.*

[51] *Pottier L.* Minimal solution of linear diophantine systems: bounds and algorithms. *In Proc. of the Fourth Intern. Conf. on Rewriting Techniques and Applications. – Como. – Italy. – 1991. – P.162–173.*

[52] *Bockmair A., Weispfenning V.* Solving Numerical Constraints. – *In Handbook of Automated Reasoning. – Elsevier Science Publishers B.V. – 2001. – P. 753–842.*

[53] *Murata T.* Petri Nets: Properties, Analysis and Applications. – *Proc. of the IEEE. – 1989. – v. 77. – N 4, P. 541–580.*

Крывый Сергей Лукьянович (10.08.1949, Украина), д-р. физико.-математических наук, профессор. 1972 закончил факультет кибернетики Киевского университета.

С 1975-2008 гг. работал в Институте кибернетики имени В. Глушкова НАН Украєни, а с 2008 и до настоящего времени работает на должности профессора кафедры информационных систем факультета кибернетики. В 1982 защитил кандидатскую диссертацию, а в 1997 докторскую диссертацию. В 2003 году присвоено ученое звание профессора.

Основные направления научной деятельности: теория автоматов, прикладная алгебра и логика, сети Петри, диофантовы ограничения, теория программных инвариантов, прикладные аспекты теории графов.

Научный руководитель 6 кандидатских диссертаций, защищенных в Украине и Польще. Член двух специализированных ученых советов по защите докторских и кандидатских диссертаций.

Автор более 200 научных и учебно методических работ. Основные работы: (учебники) "Основы дискретной математики. К., 2002; "Дискретная математика". Черновицы, 2014, Дискретная математики. Избранные вопросы". К., 2007; (учебные пособия): Лекции по дискретной математике. Санкт-Петерб., 2004; "Введение к методам создания программных продуктов". Киев-Черновицы:Букрек. - 2012, (монографии) "Wybrane zagadnienia informatyki teoretycznej". - Czestochowa.- Polska.- 2010.- 410 s. (на польськом языке), (монографии): "Алгебраические аспекты информационных технологий". К. - 2011, Онтологические методы и средства обработки предметных областей. Луганск. -2012, "Линейные диофантовы ограничения и их приложения". Киев-Черновицы:Букрек. - 2015.