ISSN 0132-3474

Номер 4 Июль - Август 2023

ПРОГРАММИРОВАНИЕ

www.sciencejournals.ru

СОДЕРЖАНИЕ

Номер 4, 2023

CONTENTS

No. 4, 2023

COMPUTER GRAPHICS AND VISUALIZATION

INFORMATION SECURITY

DATA ANALYSIS

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 3–20

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

ДОПУСТИМЫЙ ПОРЯДОК НА МОНОМАХ ВПОЛНЕ ЗАДАН. КОНСТРУКТИВНОЕ ДОКАЗАТЕЛЬСТВО

© 2023 г. С. Д. Мешвелиани[a,*]

[a]Институт программных систем им. А.К. Айламазяна РАН
152021 Ярославская обл., Переславский р-н, с. Веськово, ул. Петра Первого, 4 “а”, Россия
E-mail: mechvel@scico.botik.ru

Поступила в редакцию 23.09.2022 г.
После доработки 12.11.2022 г.
Принята к публикации 14.11.2022 г.

Обсуждаются конструктивное доказательство завершаемости алгоритма NF построения нормальной формы для многочленов нескольких переменных и связанное с ним понятие допустимого упорядочения $<_e$ на показателях мономов. В классической математике свойство обрыва убывающей цепи (well-quasiorder) отношения $<_e$ выводится из леммы Диксона, и этого достаточно для обоснования завершаемости алгоритма NF. В доказательном программировании на основе конструктивной теории типов (`Coq`, `Agda`) требуется более сильное (в конструктивной математике) свойство: свойство вполне-заданности отношения порядка (соответствует понятию well-founded – в конструктивном определении, как подобие свойства фундированности). Предлагается конструктивное доказательство этой теоремы (Т) для $<_e$, основанное на некотором известном методе, который здесь назовем “метод образцов”. Эта теорема о вполне-заданности произвольного допустимого упорядочения важна и сама по себе, независимо от алгоритма NF. Нам не известны другие работы, в которых (конструктивно) доказана эта теорема. Оказывается, она не очень сложно следует из результатов, достигнутых другими исследователями еще в 2003-м году. Доказательство запрограммировано автором на языке `Agda` в виде библиотеки `AdmissiblePPO-wellFounded` доказательных программ вычислительной алгебры, разработанной автором. Разработка включает в себя применение этой теоремы к доказательному программированию алгоритма NF. Поэтому библиотека также содержит часть доказательных программ алгебры многочленов, которая по объему значительно больше того, что нужно для доказательства теоремы Т.

Ключевые слова: компьютерная алгебра, многочлены, допустимое упорядочение мономов, конструктивное доказательство, лемма Диксона

DOI: 10.31857/S0132347423040106, **EDN:** RDEZQB

1. ВВЕДЕНИЕ

Словоупотребление.

Мы используем следующее словоупотребление.

Воплощение – программная реализация.

Свидетельство – доказательство.

Словарный порядок – лексикографический порядок.

Слово “библиотека” обозначает нашу библиотеку `AdmissiblePPO-wellFounded` [4] доказательных программ вычислительной алгебры.

Термин “хороший предпорядок” – это перевод с английского термина well-quasiorder (WQO), но – для его конструктивной разновидности (будет объяснен ниже). Например, этот перевод применен в [1].

Термин “вполне заданное упорядочение” – это наш перевод с английского термина well-founded ordering (WellFounded – в программах) – версии свойства фундированности для конструктивной логики – для применения конструктивного вывода по трансфинитной индукции (термин будет объяснен ниже). Мы не нашли в источниках перевода на русский язык термина well-founded в конструктивном значении.

Иногда применяем сокращение WF – вполне заданное (well-founded).

Логические связи между понятиями WQO и WellFounded (хороший предпорядок и вполне-заданность) – не такие, как в классической логике для понятия фундированности ([2] Глава 2, параграф 2.2). И классические определения, содержащие выражения наподобие “в любом непустом подмножестве в X существует минимальный элемент”, представлены здесь в более слабом виде, пригодном для алгоритмов, и они имеют несколько другие логические связи.

Понятия well-quasiorder и well-founded relation, – оба в конструктивном смысле, – уже давно употребляются в работах, изданных на английском языке, а также в библиотеках доказательных программ. Например, оба они определены в [18] (параграфы 2, 3). Понятие WellFounded таким же образом определено в стандартной библиотеке языка Agda.

Понятие многочлена определяется для области коэффициентов – коммутативного кольца. Здесь это кольцо считается алгоритмическим (DecCommutativeRing): операции задаются алгоритмами (функциями в программе), и дан алгоритм разрешения равенства.

Применяем разреженное представление многочлена в виде списка *ненулевых мономов*, упорядоченного по определенному закону. Показатель каждого монома представляется в плотном виде – вектор натуральных чисел (то есть – не пропуская нули в записи).

На сложность вычисления во многих методах, опирающихся на арифметику многочленов, существенно влияет выбор упорядочения для списка мономов в многочлене. Даже само решение некоторых вычислительных задач получается просто подбором подходящего упорядочения на множестве показателей мономов.

В этой статье мы приводим описание машинно-проверяемого (формального) конструктивного доказательства теоремы о том, что всякое *допустимое упорядочение* на показателях мономов является вполне заданным (well-founded – в конструктивном определении).

Нам не известны другие работы, в которых (конструктивно) доказана эта теорема.

Оказывается, что она не очень сложно следует из результатов, достигнутых другими исследователями [5] еще в 2003-м году – мы называем их метод “метод образцов”. Надо лишь добавить к нему некоторые построения – они описаны в Разделе 6.

При этом мы здесь вынуждены сначала переформулировать построения из [5], по-своему уточнить подробности, ибо по-другому не получилось построить формальное доказательство на языке Agda. Поэтому будем говорить о нашей разновидности метода образцов.

Полные формальные конструктивные доказательства соответствующих теорем содержится в библиотеке программ [4], написанной автором на языке Agda [6, 7].

Также описывается, как эта теорема применяется к доказательному программированию алгоритма нормальной формы для многочленов, применяемого в методе базиса Грёбнера [8].

В статье говорится не только об алгебре и логике, но и о машинно-проверяемых программах формальных доказательств на подходящем языке программирования. Поэтому мы также вынуждены привести какие-то необходимые объяснения относительно системы программирования.

Порядок изложения таков.

Во **Введении** объясняется, что такое допустимое упорядочение $<e$ показателей в алгебре многочленов, как оно используется в алгоритме нормальной формы; что такое вполне-заданность (WF) отношения порядка и почему доказательство WF нужно для доказательного программирования алгоритма нормальной формы для многочленов нескольких переменных. Также в Разделе 2.2 дается небольшое введение в построение формальных доказательств на языке с зависимыми типами.

В Разделе 3 описываются некоторые подходы к решению задачи.

В Разделе 4 даны определения понятиям из метода образцов: белая область, образцы, уровни, мультимножества образцов (сокращение – PM). Также определяется сведéние по показателю образцов, уровней и PM. Определяется упорядочение $<pm$ на PM и объясняется построение доказательства WF отношения $<pm$.

В Разделе 5 описываются свойства сведéния образцов и PM. Вводится понятие лестницы показателей, отображение “ esToPM : Лестница $\to$ PM”, вводится WF упорядочение $<pps$ на лестницах. Разбирается пример построения PM по лестнице.

В Разделе 6 с помощью использования упорядочения $<pps$ доказывается конструктивно теорема о вполне-заданности допустимого упорядочения. Описывается структура библиотеки программ, воплощающих это доказательство и функцию нормальной формы многочлена.

1.1. Допустимое упорядочение показателей

Всюду ниже PP обозначает множество (и коммутативный моноид по сложению) показателей.

Аксиомы допустимого упорядочения $<_e$ на показателях таковы:

- STO-e | это строгое линейное (total) упорядочение (IsStrictTotalOrder $<_e$),
- $+_e$mono | $\forall e \in$ PP, $(e +_e)$ сохраняет $<_e$,
- >0 | $\quad e \neq 0_e \Rightarrow e >_e 0_e$.

Например, словарное упорядочение на векторах над $\mathbb{N}$ установленной длины n удовлетворяет этим законам. Умножению мономов соответствует сложение $+_e$ показателей. Например, при $n = 2$ произведение

$$(x^2y^0) * (x^1y^5)$$

мономов имеет показатель [3, 5].

x $\leq_e$ y определим как x $<_e$ y или x = y.

Также для нашей цели важно учитывать *поточечное упорядочение на показателях*: $\forall$i, $e_i \leq e'_i$.

Обозначим его $\leq_p$.

Соответственно, e $<_p$ e' определим как e $\leq_p$ e' и e $\neq$ e'.

Упорядочения $<_p$ и $\leq_p$ являются частичными и не являются линейными.

Как легко следует из аксиом допустимого упорядочения, порядок $<_e$ является расширением для $<_p$:

$$\forall x, y, \quad x <_p y \Rightarrow x <_e y$$

Далее, для мономов m, m' с обратимыми коэффициентами

m делит m' тогда и только тогда, когда

exp(m) $\leq_p$ exp(m').

Будем говорить в таком случае "показатель exp(m) делит exp(m')".

Также в этом случае exp(m) $\leq_e$ exp(m') – в силу аксиом $+_e$mono, >0.

С вычислительной точки зрения одно из выгодных представлений многочлена – это список мономов, упорядоченный по убыванию относительно $<_e$. Понятия

старший показатель lExp многочлена,

старший моном lm, старший коэффициент lc

заданы в смысле упорядочения $<_e$.

1.2. Алгоритм нормальной формы многочлена

Для исследования свойств систем алгебраических уравнений от нескольких переменных важно уметь решать задачу распознавания отношения

$$f \in \text{Ideal(gs)}.$$

Это задача поиска разложения

$$f = q_1 g_1 + \ldots + q_m g_m, \qquad \text{(I)}$$

где gs – список многочленов, $g_i \in$ gs, а q_i – искомые многочлены (задача распознавания принадлежности многочлена идеалу). Она решается с помощью алгоритма вычисления базиса Грёбнера [8]. Частью этого метода является алгоритм нормальной формы многочлена.

Алгоритм NF приведения многочлена f к нормальной форме относительно списка gs многочленов находит многочлены $h, q_1, \ldots, q_m$ такие что

$$f = q_1 g_1 + \ldots + q_m g_m \; + h, \qquad \text{(II)}$$

где h является нормальной формой относительно gs, в том смысле, что $h = 0$ или ни один из старших мономов многочленов из gs не делит старший моном h.

Формула (II) представляет собой некоторое обобщение деления многочленов с остатком: h – остаток, q_i – "частные".

Заметим, что не для всех списков gs алгоритм NF распознает принадлежность идеалу (I); она распознается, например, когда gs является базисом Грёбнера.

Здесь мы считаем, что область K коэффициентов является *полем*, то есть всякий ненулевой коэффициент обратим (например, это так для рациональных чисел). Существуют разновидности этого алгоритма для более общих случаев, но и этот случай уже весьма общий.

Алгоритм NF действует следующим образом.

Нулевой f считается нормальной формой.

Для ненулевого f в списке gs находится первый многочлен, старший показатель e которого делит старший показатель e' многочлена f.

Если такого нет, то нормальной формой является f. Если такой многочлен g найден, то старший коэффициент многочлена g не равен нулю и потому делит старший коэффициент f. Поэтому старший моном m многочлена g делит старший моном m' многочлена f.

Пусть моном m_q является частным этого деления. Тогда алгоритм NF применяется рекурсивно к аргументам f', gs, где

$$f' = f - m_q g \qquad \text{(III)}$$

В нашей библиотеке функция нормальной формы также накапливает частные q_i. Здесь же рассматриваем только вычисление остатка и вопрос завершаемости программы.

(III) – это очередной шаг сведе́ния к нормальной форме.

Список мономов в многочлене упорядочен согласно любому выбранному для всей области многочленов допустимому упорядочению $<_e$ показателей.

Старший моном f при сведе́нии к f' на очередном шаге сведе́ния сокращается, и имеет место

Лемма 1. Все мономы в f' меньше старшего монома f в смысле $<_e$.

Доказательство просто следует из такой же простой леммы:

Лемма 2. Если старшие мономы многочленов f и g равны моному m, то все мономы разности

f – g меньше m в смысле $<_e$ (при этом для нулевой разности список этих мономов пуст).

Доказательство очевидно следует из алгоритма сложения списков мономов и из упорядоченности этих списков.

Теперь для доказательства Леммы 1 надо заметить, что умножение на моном m_q сохраняет порядок на мономах g – в силу свойства $+_e$mono допустимого упорядочения. Также старшие мономы многочленов f и (m_q * g) оказываются равными, и применение Леммы 2 доказывает Лемму 1.

Алгоритм NF работает до тех пор, пока в списке gs находится сводящий многочлен g. Получается убывающая последовательность $e_1 >_e e_2 >_e \ldots$ старших показателей сводимого многочлена.

Определение. Частичный порядок $\preceq$ на множестве X называется хорошим предпорядком (well-quasi-order, WQO), если для любой бесконечной последовательности a_n в X существуют $i < j$ такие, что $a_i \preceq a_j$.

Здесь мы всюду предполагаем, что строгий частичный порядок $\prec$ определен через нестрогий частичный порядок $\preceq$ как $x \prec y = x \preceq y$ и $x \neq y$.

Определение. Назовем частичный порядок $\prec$ *обрывающимся-вниз*, если он не допускает никакой бесконечно убывающей последовательности.

Заметим, что из свойства WQO следует свойство обрывание-вниз. Ибо для любой бесконечной последовательности a_n существуют $i < j$ такие, что $a_i \preceq a_j$, а из этого неравенства в наших условиях следует отрицание отношения $a_j \prec a_i$, и это доказывает, что всякая бесконечная последовательность не является убывающей в смысле $\prec$.

Далее, для упорядочения $\leq_p$ свойство WQO доказывается как известная лемма Диксона, [9] (Section 2.2), [5, 10, 11]. Ее можно выразить так: всякая последовательность e_i показателей, в которой каждый член не делится ни на один предыдущий, обрывается.

Удивительно, что доказательство этого подсознательно очевидного высказывания не так уж и просто изобрести, тем более – когда требуется конструктивное доказательство.

Из леммы Диксона также следует, что любое допустимое упорядочение $<_e$ на показателях является обрывающимся вниз. Ибо для любой бесконечной последовательности e_i показателей по лемме Диксона находятся i < j такие, что e_i делит e_j. Так как $\leq_e$ включает в себя $\leq_p$, то $e_i \leq_e e_j$.

Следовательно, согласно *классической математике*, алгоритм NF считается завершающимся.

Но этого не достаточно для систем доказательного программирования, основанных на конструктивной теории типов: Coq, Agda, ...

2. ПРОБЛЕМА ДОКАЗАТЕЛЬСТВА ЗАВЕРШАЕМОСТИ АЛГОРИТМА NF

Мы имеем дело с доказательным программированием символьных вычислений в алгебре, основанным (по умолчанию) на конструктивных машинно-проверяемых доказательствах. Для выражения алгоритмов и доказательств применяем язык Agda [6, 7].

В работах [9] (Section 2.2), [10] рассматривается среди других предмет конструктивного доказательство леммы Диксона.

[11] доказывает на языке Agda лемму Хигмана для алфавита из двух букв. Вполне возможно, что из этого метода получится доказательство леммы Диксона. Для этого надо показатель монома записать в унарной системе, используя две буквы, где вторая используется как разделитель между записями чисел.

Наконец, в [5] мы увидели, что конструктивное доказательство леммы Диксона на языке Agda для современной версии библиотеки языка Agda можно получить, применяя метод образцов из [5] (Section 2.2). Это и было нами сделано. Этот метод выбран нами так как он, как видно из дальнейшего изложения, служит и другим важным целям данного исследования.

Но в языках конструктивных доказательств, основанных на теории зависимых типов, для доказательства завершаемости алгоритма не достаточно опираться на свойство обрывание-вниз некоторого отношения порядка.

Из-за некоторых свойств интуиционистской теории типов оказывается необходимым немного более сильное свойство упорядочения: вполне-заданность (well-founded relation). В частности это требуется в системах Agda и Coq.

Достижимость и вполне-заданность.

В стандартной библиотеке языка Agda написано на языке Agda следующее определение (здесь обозначения заменяем на более привычные).

```
WfRec : (_<_ : Rel A) → RecStruct A
WfRec _<_ P x = ∀y (y < x → P y)
"The accessibility predicate: x is accessible
if everything which is smaller than x is also
accessible (inductively)."
data Acc {A : Set} (_<_ : Rel A) (x : A) : Set
  where
  acc : (WfRec _<_ (Acc _<_) x) → Acc _<_ x
WellFounded : Rel A → Set
WellFounded _<_ = ∀x (Acc _<_ x)
```

Этот отрывок программы определяет свойство Acc *достижимости* элемента (x) относительно упорядочения _<_. В строчке acc ... в конструктор WfRec вторым аргументом в переменную P ставится значение – свойство (Acc _<_). Итого получается

$$\mathrm{acc} : (\forall y\ (y < x \to \mathrm{Acc}\ \_<\_\ y)) \to \mathrm{Acc}\ \_<\_\ x$$

При этом определение WfRec и произвольное свойство P на A исчезают из определения достижимости. Поэтому это определение равносильно следующему.

Элемент $x \in A$ является достижимым по отношению $<$ на A, если и только если все y, меньшие x по отношению $<$, являются достижимыми.

(заметим, что отсюда следует достижимость всякого минимального элемента).

Отношение $<$ порядка на множестве A называется вполне заданным (WF), если все элементы A являются достижимыми для $<$.

Для простоты мы здесь рассматриваем эти определения – относительно любого частичного порядка $<$. Хотя в системах `Agda` и `Coq` (в конструктивной теории типов) это так определяется для любого двуместного отношения.

Это не что иное, как свойство *трансфинитной индукции*. То есть для любого вполне заданного упорядочения $\prec$ на множестве A и любого свойства P на множестве A доказательство этого свойства способом трансфинитной индукции по отношению $\prec$ считается законным. Некоторые упорядочения обладают этим свойством. Наша цель – это доказать вполне-заданность отношения $<_e$, или хотя-бы как-то доказать конструктивно (на языке `Agda`), что функция NF завершается.

Ссылки на доказательство леммы Диксона для этого не достаточно.

В работе [10] пишется о доказательной программе метода базиса Грёбнера (алгоритм нормальной формы является частью этого метода). Но в части доказательства завершаемости указывается на неконструктивное доказательство леммы Диксона. Это не соответствует нашей цели.

В работе [9] описывается построение доказательной программы (в системе `Coq`) метода базиса Грёбнера. Но при этом изначально предполагают вполне-заданность упорядочения на мономах (Раздел 2. “We assume a compatible well-founded order $\prec$ on the monomials ...”). Слабым местом такого подхода является то, что пользователю программы придется для каждого применяемого допустимого упорядочения отдельно доказывать вполне-заданность (well-founded). А в зависимости от задачи применяются самые разные упорядочения мономов.

Мы же конструктивно выводим это свойство вполне-заданности в общем случае, из аксиом допустимого упорядочения.

2.1. Примеры вполне заданных упорядочений

Обычный порядок $<$ на $\mathbb{N}$. Здесь WF обращается в обычный принцип индукции по построению натурального числа.

Порядок $<$ на $\mathbb{Z}$ (целые числа) не является ни обрывающимся вниз, ни тем более – WF.

Словарный порядок по $<$ на множестве $\mathbb{N} \times \mathbb{N}$ пар – вполне задан. Предлагаем это доказать как упражнение. Заметим, что здесь существует бесконечно много пар меньших, чем, например, (1, 0).

Известно, что прямое поизведение вполне заданных порядков $\prec$ и $\prec'$ является вполне заданным порядком на множестве пар со словарным упорядочением по порядкам $\prec$ и $\prec'$. В частности словарное упорядочение на множестве n-к (векторов) над $\mathbb{N}$ при постоянном n является WF (это также доказано в нашей библиотеке).

Другой пример: словарное упорядочение на списках натуральных чисел не является WF. Например, последовательность списков 1, 01, 001, 0001, ... бесконечно убывает в словарном упорядочении на списках, чего не может быть для WF упорядочения.

Для некоторых частных случаев допустимого упорядочения $<_e$ нетрудно доказать WF. Например: для словарного порядка <lex, для порядка degLex – “сначала по полной степени, потом – словарно”. Но ясно, что здесь более всего ценно одно доказательство для общего случая.

Также известна классификация всех допустимых упорядочений [12]. Но она основана на очень сложных построениях. И если пользоваться этой классификацией, то скорее всего требуемое доказательство окажется сложнее, чем приводимое ниже, это будет проблемой.

2.2. О доказательном программировании на языке `Agda`

Наша программа опирается на базовую библиотеку вычислительной алгебры `Admissble-PPO-wellFounded` [3, 4] доказательных программ, написанных автором на языке `Agda` [6, 7]. Этот язык основан на чистой функциональности, “ленивом” способе вычисления по умолчанию, поддержке обобщенного программирования (generic programming), аппарате зависимых типов. Приблизительно можно считать, что это язык `Haskell`, расширенный аппаратом зависимых типов. Зависимые типы позволяют адекватно описать алгебраическую область, зависящую от обычного значения ([3], Введение), например, от целого числа. Кроме того, этот аппарат позволяет вставлять в программу доказательства утверждений, и эти доказательства будут проверяться компилятором. Становится возможным описывать метод вычисления в программе так, как это дела-

ется в учебниках и научных статьях, вместе с доказательством правильности.

Техника проверки доказательств компилятором (точнее – проверяльщиком типов, type checker) основана на изоморфизме Карри–Ховарда [13, 14]. Всякое утверждение S представляется подходящим типом T (зависящим от значений). Доказательство для S есть любая функция (завершающийся алгоритм), которая выдает любое значение v типа T. Проверяльщик типов проверяет отношение v : T посредством символьных вычислений с выражениями типов. Таким образом доказательство утверждения проверяется до начала компиляции в исполняемый код.

Недоказуемое высказывание соответствует пустому типу $\bot$ с пустым множеством значений. Отрицание высказывания соответствует функции, отображающей соответствующий высказыванию тип в пустой тип. Импликация выражена типом $S \to T$ всех функций из S в T, где S и T представляют соответствующие утверждения. Конъюнкция выражена произведением $S \times T$ типов, дизъюнкция выражена суммой $S \uplus T$ типов. Доказательство индукцией по построению данного выражается в виде рекурсивно заданной функции. Применение леммы выражается в виде вызова функции, представляющей доказательство леммы.

Пока ограничиваемся только конструктивными доказательствами [15, 16] (без использования принципа Маркова для доказательства завершаемости алгоритмов). В частности, всякий объект существует лишь как итог некоторого данного алгоритма, и должно быть дано доказательство завершаемости этого алгоритма. Мы часто пользуемся законом исключенного третьего – это возможно в тех случаях, когда приведен алгоритм разрешения соответствующего отношения.

По умолчанию, доказательства в языке Agda – конструктивные. Конструктивное доказательство имеет то преимущество перед неконструктивным, что, например, вместе с существованием объекта предъявляется алгоритм его построения. Если же доказательство необходимо, а конструктивное доказательство трудно предъявить, то Agda допускает введение аксиомы исключенного третьего (чем в нашей библиотеке мы пока ни разу не воспользовались).

В каком смысле функция на языке Agda может быть доказательством?

Если функция задает доказательство (как, например, в нашем случае доказательства $<_e$wellFounded : WellFounded _$<_e$_ теоремы о WF отношения $<_e$), то заголовок этой функции (выражение ее типа, signature) является формулировкой теоремы, а тело (implementation) этой функции является доказательством этой теоремы. Это доказательство проверяется проверяльщиком типов в общем, символьном виде, так же, как читатель проверяет доказательство, изложенное в книге. Если и когда эта функция (f) вычисляется во время исполнения, то она выдает доказательство для какого-то частного случая. И если головная функция не требует разбора строения этого доказательства, то всякая функция – клиент для f не разбирает это строение при исполнении (здесь также существенно “ленивое” вычисление в языке Agda). За счет этого затраты на проверку необходимых доказательств для головной функции не входят в затраты на вычисление результата головной функции во время работы исполняемого кода.

Другой пример: функция polNF нормальной формы многочлена из нашей библиотеки запрограммирована с использованием доказательства $<_e$wellFounded вышеназванной теоремы (Раздел 6). Это доказательство опирается на некоторые символьные вычисления с типами. Исполняемый код для polNF вычисляет нормальную форму многочлена, но на этом этапе нет затрат на вычисления для доказательства $<_e$wellFounded.

О доказательстве завершаемости

По умолчанию Agda должна убедиться в завершаемости каждой заданной функции (алгоритма). Часто Agda сама убеждается в завершаемости путем обнаружения синтаксического уменьшения рекурсивных вызовов. Но также часто выдается сообщение “Termination checking failed”. В этом случае надо помочь проверяльщику типов распознать завершаемость. Обычно проще всего ввести в функцию дополнительный аргумент в виде счетчика – натурального числа. Но этот подход не является достаточно общим. Например, он не годится для рассматриваемых здесь функций NF, polNF.

В наболее общем случае завершаемость доказывается введением некоторого вполне заданного упорядочения $\prec$ на некотором типе T, добавлением в определение функции f аргумента типа T и доказательством некоторых свойств этого аргумента, в частности того, что этот аргумент уменьшается в отношении $\prec$ при каждом рекурсивном вызове f. Пример этому приводится в дальнейшем изложении. Но два простейших показательных примера содержатся в модуле WellFounded-help.agda библиотеки [4].

Здесь же мы доказываем среди других утверждений, что для доказательств завершаемости функции NF нормальной формы достаточно использовать теорему WF для того допустимого упорядочения $<_e$ на показателях мономов, которое выбрано параметром для арифметики многочленов.

В этой статье мы не можем дать более подробные объяснения по данной системе программирования. Некоторые объяснения и примеры содержатся в [3, 7].

Хорошим введением для начинающих в программирование алгебры на языке Agda является статья [17].

2.3. О библиотеке `AdmissiblePPO-wellFounded`

В этой статье мы говорим об алгоритмах, которые запрограммированы с опорой на арифметику коммутативного кольца многочленов (тип `Pol`) над произвольным полем K, воплощенную в виде доказательных программ. Поэтому эта библиотека включает в себя задание некоторых категорий алгебраических областей дополняющих иерархию алгебры из стандартной библиотеки: IntegralDomain (целостное кольцо с разрешимым равенством), GCDDomain, EuclideanDomain, Field (поле) и некоторые другие. Из конструкторов алгебраических областей реализованы конструктор SVector векторов над моноидом в разреженном представлении, конструктор Pol кольца многочленов нескольких переменных над коммутативным кольцом.

Это на самом деле части библиотеки `DoCon-A` доказательных программ алгебры, разработанной тем же автором; взяты те части, которые нужны для программного воплощения доказательств, обсуждаемых в этой статье.

2.4. О синтаксисе языка Agda

Имена операторов и отношений отделяются пробелами. Например, в строке

```
a+b≈0 : a + b ≈ 0#
```

программы a+b≈0 есть имя значения, двоеточие отделяет имя значения от выражения типа, символы ‘+’ и ‘≈’ в выражении типа суть соответственно имя оператора сложения и имя двуместного отношения, 0# есть имя нулевой постоянной. Вся эта строка означает объявление: значение a+b≈0 имеет тип a + b ≈ 0# (и этот тип зависит от значений a, b, 0#).

Знак подчерка в имени отношения или операции означает, что в синтаксисе программы во входном выражении на этой позиции ставится выражение аргумента этой операции.

Некоторые замечания о языке и стандартной библиотеке:

x : T означает “x принадлежит типу T” (конструкция языка Agda).

Знак ‘=’ это присваивание, конструкция языка.

Равенство _≈_ это знак абстрактного отношения равенства на некоторой области, для каждой области это отношение может задаваться программой пользователя или стандартной библиотекой.

_::_ это функция из стандартной библиотеки, x :: xs означает присоединение x к списку xs.

Прописные и строчные буквы в программе различаются проверяльщиком типов. Обычно имена типов начинаются с прописной буквы, а имена функций начинаются со строчной буквы.

@ это знак присваивания в образце (конструкция языка). Например, в программе

```
foo f@(mkPol mons cs≉0 ord-exps) g =
          f + (foo' mons) * g where …
```

функция foo принимает аргумент f, разбирая его по образцу, и в правой части используется как имя f, так и часть mons образца для f.

2.5. Использование вполне-заданности $<_e$ в программе NF

Представим себе, что мы получили доказательство $<_e$wellFounded того, что всякий допустимый порядок $<_e$ на показателях является вполне заданным.

Тогда функция NF в простейшем виде программируется примерно так (условный код, для показа замысла):

```
NF gs f = nf gs f (<e wellFounded e[f])
  where
  e[f] = lExp f
  gs0 = gs
  nf : (gs : List Pol) (h : Pol) →
                          let e[h] = lExp h hm
                          in
                          Acc _<e_ e[h] → Pol
  nf gs h (acc rc) = ...
```

То есть вызывается местная вспомогательная функция nf, которая будет ниже программироваться рекурсивно.

($<_e$wellFounded e[f]) это *свидетельство* (доказательство) того, что показатель e[f] достижим по $<_e$. В предложениях для nf имеем следующие обозначения.

h это текущий остаток, который сводится по списку gs и в конце оказывается нормальной формой.

(acc rc) это свидетельство того, что показатель e[h] старшего монома из h достижим (для случая, когда h не нулевой).

А всякое свидетельство достижимости имеет вид (acc rc), где acc это стандартный конструктор, rc функция из определения достижимости во вполне заданном порядке.

В случае h ≠ 0 многочлен h имеет старший моном. Обозначим его m, а его показатель – e[h]. Когда в gs встретился многочлен g, старший моном которого делит m, работает такое предложение из функции nf :

nf gs h (acc rc) = nf gs_0 h’ (rc h’ e[h’]<e[h])

В правой части: gs_0 это запомненное первоначальное значение для gs,

h' = h – q*g это итог шага сведéния, когда старший моном сокращается.

e[h'] = lExp h' (если h' не нулевой).

e[h']<e[h] это свидетельство того, что e[h'] $<_e$ e[h]. Это доказательство получается в теле функции nf применением функции, воплощающей Лемму 1.

Данное (rc h' e[h']<e[h]) это свидетельство того, что показатель e[h'] достижим.

Таким образом от h и свидетельства достижимости e[h] функция nf рекурсивно переходит к h' и достижимости e[h']. Проверяльщик типов (type checker) системы Agda распознает завершаемость этой рекурсии, так как видит доказательство $<_e$wellFounded, доказательство достижимости e[h'] и доказательство e[h']<e[h].

Это самый простой способ запрограммировать метод NF в системе доказательного программирования на основе зависимых типов.

Уточнение программы NF

Когда очередной g не сводит h, и список gs не пуст, метод вызывает nf на аргументах gs' = хвост gs, h, acc' – это шаг "пробовать следующий элемент из gs". При этом многочлен h остается неизменным. Это нарушит доказательство завершаемости, так как третий аргумент должен содержать доказательство уменьшения соответствующего аргумента относительно выбранного WF упорядочения.

Поэтому в аргументы nf добавлены данные cnt : ℕ, eq : cnt ≡ length gs, где cnt – счетчик длины текущего списка gs. Так что при каждой рекурсии уменьшается либо старший показатель h (в смысле $<_e$), либо значение cnt. Чтобы обеспечить требуемое уменьшение, программа сначала определяет словарный порядок на множестве PP × ℕ пар. Пары (e , i) и (e' , j) сравниваются сначала по первой составляющей через $<_e$, и в случае равенства, сравниваются вторые составляющие по упорядочению _<_ на ℕ. Это выражено кодом

$_<_e<_$: Rel (PP × ℕ) _

$_<_e<_$ = ×-Lex _≡_ $_<_e$_ _<_,

где ×-Lex функция из стандартной библиотеки, задающая словарное произведение двух упорядочений. Потом программа доказывает что порядок $<_e<$ вполне задан:

$<_e$<-wellFounded : WellFounded $_<_e<_$

$<_e$<-wellFounded =

×-wellFounded $<_e$wellFounded <-wellFounded

Это делается вызовом общей функции ×-wellFounded, которая принимает свидетельства WF для каждого из двух порядков и возвращает свидетельство WF для прямого произведения этих порядков (простое доказательство из стандартной библиотеки). Наконец, вместо аргумента ($<_e$wellFounded e[f]) в начальном вызове nf ставится

($<_e$<-wellFounded (e[f] , length gs_0))

И при рекурсии в соответствующий аргумент ставится доказательство убывания по упорядочению $<_e<$ пары при переходе

(e[h] , длина gs) -> (e[h'] , длина gs')

В подробностях эту функцию (polNF) можно увидеть в модуле

source/GroebnerBasis/Field/PolNF-sample.agda

нашей библиотеки [4]. Это частный случай метода NF, служащий для ясного показа метода.

Настоящая функция polNF нормальной формы относительно списка многочленов содержится в модуле PolNF.agda того же раздела. Она еще накапливает список частных и в итоге выдает запись (record), содержащую нормальную форму h, список частных, свидетельство равенства (II) (Раздел 1.2), свидетельство нормальности h по gs.

Таким образом главный успех предприятия зависит от того, сумеем ли мы конструктивно доказать теорему $<_e$wellFounded вполне-заданности отношения $<_e$.

Эта теорема важна и сама по себе. Так как понятие допустимого упорядочения на показателях мономов часто используется в вычислительной алгебре.

Изложим решение этой задачи.

3. ПОДХОД К ДОКАЗАТЕЛЬСТВУ ВПОЛНЕ-ЗАДАННОСТИ УПОРЯДОЧЕНИЯ $<_e$

Известно, что в классической логике свойства WQO (или свойство обрывание-убывающей) и вполне-заданность равносильны. Но в конструктивной логике вполне-заданность не выводится из свойства обрывание-убывающей.

В статье [18] в Разделе 3.1 описано, каким образом в довольно общем случае для отношения со свойством WQO можно конструктивно доказать вполне-заданность – при некотором дополнительном условии наличия некоторого вполне заданного дерева – well-founded tree (WFT).

Но мы не видим, как строить такое дерево для отношения $<_e$.

Есть еще такой выход из положения. Для доказательства завершаемости функции NF не обязательно использовать вполне-заданность $<_e$. Достаточно связать (правильным образом) с ходом вычисления NF некое данное, которое с каждой рекурсией будет уменьшаться в смысле некоторого подобранного упорядочения $\prec$ на множестве

таких данных. При этом должна быть доказана вполне-заданность $\prec$.

И оказалось, что метод мультимножеств образцов (multiset of patterns) из Раздела 2.2 статьи [5] как раз применим в таком виде к конструктивному доказательству завершаемости метода NF.

Авторы [5] доказывали только лемму Диксона, причем использовали язык ACL2*, с которым мы не знакомы. Нам достаточно было обозреть общий замысел метода образцов в двух страницах раздела 2.2, чтобы домыслить подробности и запрограммировать на языке Agda этот метод. Наш алгоритм reducePM сведéния множества образцов (reduction of pattern set) отличается от того алгоритма, который имеется в виду в [5] (и подробности которого нам не ясны). Но это отличие не существенно, так как для нашего алгоритма мы доказали свойства, которых достаточно для конструктивного доказательства вполне-заданности допустимого упорядочения на показателях а также леммы Диксона.

При применении этого метода reducePM получается данное pm в виде мультимножеств образцов, порядок <pm на этих данных и доказательство свойства WF для <pm. Если это данное pm и некоторые ему сопутствующие данные добавить к аргументам функции polNF, то получится доказательно завершающаяся функция.

Но эта программа выглядела бы гораздо сложнее, чем целевая программа NF приведенная выше: много дополнительных аргументов в функции и доказательств связи между ними.

К счастью оказалось, что через свойство WF отношения <pm и свойства сведéния reducePM данных pm нетрудно доказать вполне-заданность отношения $<_e$ – что и приводит нас к простой программе NF, показанной выше.

Ниже описываем этот подход.

4. ОБЛАСТИ, ОБРАЗЦЫ, УРОВНИ

Пусть es список показателей. Затененной областью Sh(es) назовем (следуя [5]) множество показателей e таких, что e делится на какой-либо показатель из es. Это объединение нескольких ортантов.

Назовем белой областью W(es) дополнение PP\Sh(es).

Назовем *лестницей* такой список es показателей, в котором каждый показатель не делится ни на один последующий в списке.

Для лестницы es белая область оказывается той частью пространства PP, которая находится “под лестницей”, заданной показателями es как ступеньками.

Например, для $n = 2$ лестница списка $[e_0, e_1] = [(3,2), (1,5)]$ изображена на рис. 1.

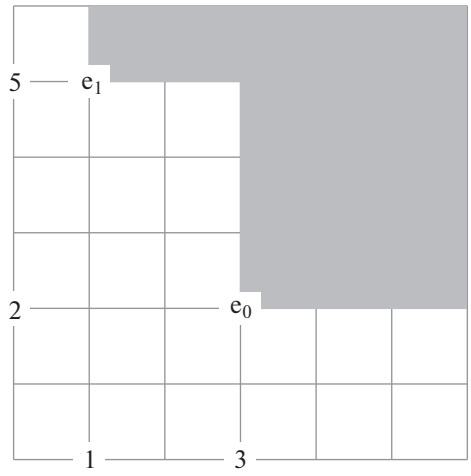


Рис. 1.

(Оба примера представления белых областей в этой статье взяты из [5]).

На рис. 1 изображена часть квадранта $0 \le x, y \le 6$. Здесь белая область состоит из двух горизонтальных и одного вертикального лучей и еще нескольких точек.

Добавление к лестнице спереди любого показателя e_2, принадлежащего $W[e_0, e_1]$, дает белую область меньшую, чем $W[e_0, e_1]$, в смысле включения $\subset$ множеств. И так далее.

Пусть <st частичное упорядочение на множестве Stc лестниц, при котором es <st es’ означает $W(es) \subset W(es')$.

Очередной целью является подобрать подходящее конечное символьное представление для белой области и доказать, что отношение <st вполне задано. Оказывается, что из этого можно вывести все остальные цели статьи.

Обратимся к методу образцов из работы [5].

Обозначим

$$\mathbb{N}_* = \mathbb{N} \cup \{*\}$$

множество натуральных чисел, расширенное элементом *, называемым *свободой*.

Образцом называется вектор в $\mathbb{N}_*^n$.

Количество свобод в образце называется *размерностью* образца.

Размерность каждого образца не меньше нуля и не больше n – числа переменных в алгебре многочленов.

Образец без свободы называем основным (в программе – Ground), он имеет размерность 0.

Образец p представляет множество показателей e таких, что вектор p совпадает с вектором e в каждой позиции, кроме тех, что содержат *.

Как в статье [5], обозначим это множество S(p). Для списка или множества ps образцов обозначим S(ps) объединение $\cup \{S(p) \mid p \in ps\}$ (сокращение: Subset in PP defined by Patterns ps).

Таким образом всякая белая область получает конечное символьное представление в виде конечного множества образцов (таких представлений может быть много).

Имеет смысл собирать в один *уровень* образцы одной размерности и представлять белую область в виде списка уровней, убывающего по размерности, пропуская пустые уровни.

Такой список уровней называем *мультимножеством образцов* (PMultiset, коротко – PM) (в [5] близкое этому понятие называют multiset).

Заметим, что длина списка уровней в PM не превосходит 1 + n, так как эти уровни упорядочены по убыванию размерности, а размерность уровня не превосходит n.

Уровень и упорядоченный список уровней введены автором, этих построений нет в [5]. В таком виде мы понимаем их отношение к построению мультимножества по образцам, к сведéнию мультимножества и наконец, к доказательству цели, и сумели запрограммировать доказательство на языке Agda. Хотя здесь мы все это описываем на неформальном математическом языке.

Область S(pm), задаваемая образцами в нашей разновидности метода, не обязательно совпадает с W(es), но она включает в себя W(es). Вместе с вполне-заданностью упорядочения <pm (определяемого ниже в этом разделе) это дает основу для доказательства цели.

Из [5] мы взяли только главное – замысел представления области в виде образцов и построения из этого мультимножества, которое можно представить в виде вектора натуральных чисел.

Уровень – это пара из значения размерности и соответствующего списка образцов.

Например, для n = 2, образец [* , *] (generalPattern) представляет все PP – белую область для пустого списка.

Другой пример: белая область $W[e_0, e_1]$ на рис. 1 может быть представлена, например, такими уровнями размерности 1 и 0:

```
{(1, [0*, *0, *1]),
   (0, [10, 11, 12, 13, 14, 20, 21,
22, 33, 24])}
```

В этом и последующих примерах распечатки образцов мы часто пишем ij как краткое обозначение для пары (i, j), когда i, j < 10, и i* вместо (i, *).

Кратностью (lMulty) уровня назовем длину списка этого уровня.

Заметим, что длина списка не обязательно равна порядку множества элементов списка. Но для наших построений эта разница не существенна.

О применяемых упорядочениях:

чтобы облегчить чтение статьи, перечислим здесь все используемые нами упорядочения.

$<, \leq$ – упорядочения на натуральных числах.

$\leq_p$ – покоординатное упорядочение по $\leq$ на векторах над $\mathbb{N}$ (частичное).

$<_e$ – произвольное допустимое упорядочение на PP (векторах над $\mathbb{N}$).

$\leq^*$ – отношение между элементами $\mathbb{N}$ и элементами $\mathbb{N}_*$ (определенное ниже).

≤ep – отношение между показателями (типа PP) и образцами (типа Pattern), заданное (ниже) покоординатно через $\leq^*$ (частичное).

<nn – упорядочение на парах натуральных чисел, заданное словарно через <.

<lev – упорядочение на уровнях, заданное (ниже) через упорядочение <nn пар (размерность уровня, длина списка образцов уровня).

<ls – упорядочение списков уровней, заданное (ниже) словарно относительно упорядочения <lev.

<pm – упорядочение на мультимножествах образцов (PM), заданное (ниже) через <ls (почти то же, что <ls).

<pps – упорядочение на списках показателей, определенное (ниже) через <pm и отображение esToPM! : List PP → PMultiset.

Некоторые из этих упорядочений, – в общем случае скажем $\prec$ на X, – определяются через другое упорядочение, скажем $\prec'$ на Y, путем перенесения по подобранному отображению f : X → Y. При этом x $\prec$ x' определено как f x $\prec'$ f x'.

В стандартной библиотеке эта конструкция осуществляется оператором _on_ :

$$\_\prec\_ = \_\prec'\_ \text{ on } f$$

Если $\prec'$ вполне задано на Y, то $\prec$ вполне задано на X. Эта простая лемма доказана в стандартной библиотеке.

Назовем *размером* уровня lev пару натуральных чисел lSize lev = (d , |pats|)

– размерность уровня lev и его кратность.

Пусть <nn это словарный порядок на множестве $\mathbb{N} \times \mathbb{N}$. Через него определим порядок на уровнях:

<lev : Rel DimLevel _

<lev = _<nn_ on lSize

– то есть соответствующие значения lSize сравниваются по <nn.

Порядок <ls на списках уровней задаем как словарный порядок над порядком <lev :

_<ls_ : Rel DimLevels _

_<ls_ = Lex-< _ _<lev_

Порядок <pm на мультимножествах образцов задаем так:

_<pm_ : Rel PMultiset _

_<pm_ = _<ls_ on dimLevels

– сравнить списки уровней в словарном порядке над порядком <lev.

Видно, что <pm – это порядок на PMultiset изоморфный порядку lex(<nn) на мультимножествах над $\mathbb{N}$, представленных в виде упорядоченных по убыванию размерности списков пар (размерность, кратность), где 0 ≤ размерность ≤n.

То есть <pm по сути дела задает порядок на списках вида

$$(d_1, m_1), \ldots, (d_k, m_k), \qquad \text{(IV)}$$

где d_i, $m_i \in \mathbb{N}$, $n \geq d_1 > \ldots > d_k \geq 0$, и все кратности m_i ненулевые.

Этот порядок вкладывается в словарный порядок на множестве векторов над $\mathbb{N}$ длины 1 + n. При этом вложении d_i суть номера мест в векторе, а составляющие в остальных местах заполняются нулями.

Известно, что такой порядок является вполне заданным.

В стандартной библиотеке `lib-1.7` языка Agda нет доказательства этому. Это доказательство запрограммировано в нашей библиотеке для разреженного представления вектора в виде (IV).

Так что программа теперь ссылается на функцию <pm-wellFounded конструктивного доказательства вполне-заданности упорядочения <pm на мультимножествах образцов.

Далее, задуманное нами доказательство вполне-заданности допустимого порядка $<_e$ имеет следующий вид. Пусть e : PP – любой показатель. Надо доказать его достижимость Acc _$<_e$_ e. То есть доказать, что всякий показатель e' $<_e$ e является достижимым.

e не делит e', в силу законов отношения допустимого упорядочения для $<_e$ (Раздел 0.1).

Поэтому получаем лестницу из e' :: e :: [] из двух показателей (смотри начало Раздела и рисунок 1).

Далее, надо доказать Acc _$<_e$_ e" для каждого e" $<_e$ e'. Отношение $<_e$ транзитивно. Поэтому выполнено e" $<_e$ e, и потому список e" :: e' :: e :: [] является лестницей. Рассмотрим все последовательности лестниц вида

$$[], [e], (e' :: e :: []), (e'' :: e' :: e :: []), \ldots$$

– очередная лестница получается из предыдущей добавлением головного показателя, который меньше по $<_e$ всех остальных показателей лестницы.

Каждой лестнице st из такой последовательности сопоставляем ее белую область W(st). Очевидно, что эти множества убывают по включению:

$$W([]) \supset \ldots \supset W(st_i) \supset W(st_{1+i}) \supset \ldots$$

Ниже мы по закону сведе́ния каждой такой области (лестнице) сопоставляем мультимножество pm(st) образцов так, что pm(st(1+i)) получается из pm(st(i)) некоторым сведе́нием образцов (pattern reduction) по очередному показателю e_i и перемещением полученных образцов между уровнями.

Ниже определяем это сведе́ние и доказываем, что каждое сведе́ние pm(st(i)) → pm(st(1+i)) уменьшает мультимножество в смысле порядка <pm.

В нашей разновидности метода образцов область S(pm), задаваемая образцами, не обязательно совпадает с W(es). Но она включает в себя W(es). В методе, описываемом ниже, последовательность pm_0 >pm pm_1 >pm ... убывает в подобранном нами вполне заданном порядке <pm, и это дает основу для доказательства цели.

4.1. Сведе́ние образца, уровня, мультимножества образцов

Отношение ≤* между элементами $\mathbb{N}$ и $\mathbb{N}_*$ определим так:

i ≤* * для любого натурального i,

и i ≤* j значит, что i ≤ j для натурального j.

Отношение ≤ep между показателями и образцами (типа Pattern) определим как ≤* примененное поточечно. Это частичный порядок.

p : Pattern называется *сводимым* по e : PP (e Reduces p), если e ≤ep p.

Это определение равносильно определению отношения “p is reducible by e” из ([5], 2.2).

Для образца p, сводимого по показателю e, функция сведе́ния

$$\text{reducePattern} : PP \to Pattern \to Patterns$$

выдает список образцов, называемых нами редексами. В целевом доказательстве она применяется только когда e сводит p.

Если p содержит * в некотором месте вектора, и e содержит j в этом месте, то список редексов для этого места – это все образцы p'(0), ... p'(j-1), где p'(k) получен из p заменой * в этом месте на число k.

Исключение делается для j = 0, в каком случае * на этом месте заменяется на 0.

Здесь имеется отличие от сведе́ния в [5], где пишется о множестве всех чисел меньших j, тогда как при j = 0 это множество оказывается пустым. Сначала мы пытались следовать этому построению, но при этом получаемое множество образ-

цов иногда оказывалось недостаточным. Поэтому мы применили описанное выше исключение для j = 0. По крайней мере, доказательство цели – проходит.

Это действие для * производится по очереди для всех мест свободы в p. Полученные списки редексов соединяются вставкой применением функции insertPatterns так, что возвращаемый список редексов не содержит повторов.

Наш алгоритм сведе́ния устроен так, что для основного образца, сводимого по e, выдается пустой список.

Доказываем в программе, что для образца размерности d сводимого по e все остатки в (reducePattern e p) имеют размерность d – 1.

Функция

reducePatternToLevel : (e : PP) (p : Pattern) →
e Reduces p → DimLevel

требует, чтобы p был сводим по e. Она применяет (reducePattern e p) и выдает уровень, состоящий из этих остатков, и все они имеют размерность (dim p) – 1.

Сводимый основной образец дает при таком сведе́нии пустой уровень.

Уровень lev называем сводимым по показателю e (e ReducesLevel lev), если e сводит некоторый образец в lev.

Функция

```
reduceLevel : (e : PP) (lev : DimLevel) →
                                  DimLevels
```

сводит уровень, она выдает список не более, чем из двух уровней следующим образом.

Если lev не сводим по e, то выдается [lev].

Иначе, если lev содержит один образец p, то выдается [reducePatternToLevel e p rd].

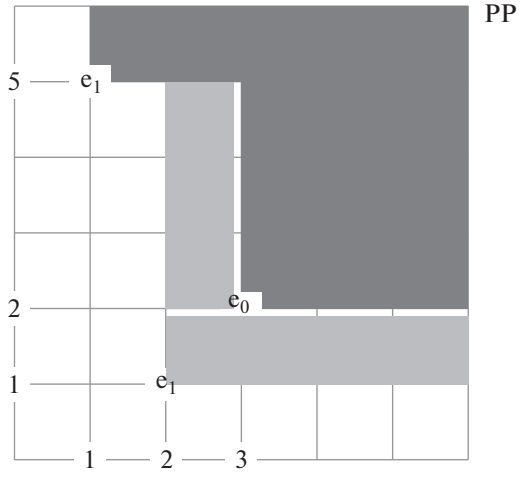


Рис. 2.

Иначе в lev находится первый образец p сводимый по e, вычисляется уровень

rLev = reducePatternToLevel e p e-reduces-p,

и выдается список (delPattern p lev) :: rLev ::[]

из двух уровней. Первый из них получен удалением из lev первого вхождения образца p. В итоге первый уровень имеет размерность dim(lev) и кратность lMulty(lev) – 1,

второй – имеет размерность dim(lev) ∸ 1 (здесь стоит знак усеченного вычитания натуральных чисел: если уменьшаемое меньше вычитаемого, то итогом считается 0).

Вообще, сведе́ние образцов, уровней и списков уровней устроены так, что было легче доказать согласованность сведе́ния с упорядочением <pm и с отношением включения белых областей.

Функция

```
preReduceLevels : PP → DimLevels → DimLevels
preReduceLevels _ [] = []
preReduceLevels e (l :: levs)
                    with e reducesLevel? l
… | no _ = l :: (preReduceLevels e levs)
… | yes rd-l =
        insertLevels (reduceLevel e l) levs
```

принимает показатель e и список levs уровней и возвращает список уровней. Она ищет в levs первый сводимый по e уровень. Если такого нет, то возвращает levs.

Иначе она сводит найденный уровень в список levs' уровней и вставляет levs' в оставшийся хвост списка levs.

Здесь функция insertLevels вставляет уровни в список по одному, применяя функцию insertLevel.

insertLevel вставляет уровень l в список согласно убыванию размерности. Если встретится уровень l' той же размерности, то l и l' сливаются в один уровень (joinLevels^{r} l l' d≡d'), при этом спис-

ки образцов приставляются друг к другу, соответственно, кратности складываются.

Функция reduceLevels применяет preReduceLevels и удаляет из результата пустые уровни.

Напомним, что мультимножество образцов (PMultiset, PM) состоит из списка непустых уровней, упорядоченного по убыванию размерности уровня.

Отношение

```
_ReducesPM_ : REL PP PMultiset _
_ReducesPM_ e pm =
             Any (e ReducesLevel_) (dimLevels pm)
```

означает, что показатель e сводит PM, то есть сводит некоторый уровень в PM.

Сведе́ние PM выражается функцией

```
reducePM : (e : PP) → PMultiset → PMultiset
reducePM e pm@(mkPM levs allHasHead-levs ord-levs)
                              with e reducesPM? pm
  … | no _             = pm
  … | yes any-rd-levs =
          mkPM levs' allHasHead-levs' ord-levs'
  where
  levs' = reduceLevels e levs
  ord-levs' = reduceLevels-preserves-<d-DecrOrdered
                                          e ord-levs
  allHasHead-levs' = allHasHead∘reduceLevels e levs
```

Она оставляет pm : PMultiset (с уровнями levs) неизменным, если e не сводит pm.

Иначе она выдает pm' : PMultiset, в котором уровни получены вызовом (reduceLevels e levs).

Здесь allHasHead∘reduceLevels это функция, которая строит доказательство того, что каждый уровень, выдаваемый функцией reduceLevels, не пуст.

5. СВОЙСТВА СВЕДЕ́НИЯ МУЛЬТИМНОЖЕСТВА ОБРАЗЦОВ

В нашей программе доказана на языке Agda теорема

```
reducePM<pm : {e : PP} {pm : PMultiset} →
                       e ReducesPM pm →
                       reducePM e pm <pm pm
reducePM<pm {e} {pm} e-rd-pm with e reducesPM? pm
… | no ¬rd = contradiction e-rd-pm ¬rd
… | yes rd = reduceLevels<ls (decrOrdered pm) rd
```

– если e сводит мультимножество pm образцов, то (reducePM e pm) выдает pm' : PMultiset, которое меньше pm в смысле <pm.

Доказательство основано на следующих соображениях.

Если сводится уровень нулевой размерности, то это выражается в удалении одного образца из этого уровня. При этом уменьшается кратность уровня.

Если сводится уровень ненулевой размерности, то либо он заменяется на уровень меньшей размерности, либо он заменяется на уровень той же размерности с меньшей кратностью и еще на уровень меньшей размерности.

Надо иметь в виду, что в нашей задаче сведе́ние списка уровней применяется только к тем спискам, которые уже упорядочены по убыванию размерности уровня. Подробности такого рода существенны при построении доказательств.

5.1. Соответствие лестница → мультимножество образцов

Есть важное свойство согласованности между PM, лестницами (их белыми областями) и действием сведе́ния. В ходе действий над PM (сведе́ние образцов, вставка уровней, и тому подобное)

появляются разные PM. Для нашей цели необходимо убедиться в сохранении важных свойств связи появляющихся PM и белых областей.

Говорим, что показатель e принадлежит образцу p (e ∈ pat p),

если e получается подстановкой некоторых чисел вместо вхождений * в p.

Заметим, что если e ∈ pat p, то e Reduces p.

Соответственно считаем, что e принадлежит списку ps образцов (e ∈ pats ps),

если e ∈ pat p для некоторого p из ps (Any (e ∈ pat_) ps).

Считаем, что e принадлежит уровню lev, если e принадлежит списку образцов lev. Считаем, что e принадлежит pm : PMultiset (e ∈ pm pm), если e принадлежит некоторому уровню из pm.

Определение

```
LevelsCoverRegion : REL DimLevels (List PP) _
LevelsCoverRegion levs es =
          ∀(e : PP) → es ¬EsDivides e →
                    e ∈els levs
```

значит, что список levs уровней покрывает белую область (“Region”), заданную списком показателей. То есть всякий показатель e, принадлежащий области W(es), также принадлежит некоторому уровню из levs.

Отношение “ pm : PMultiset покрывает белую область W(es)” (PMCoversRegion pm e) означает, что список уровней из pm покрывает W(es).

Далее, функция

esToPM : (es : List PP) →

∃(\(pm : PMultiset) → PMCoversRegion pm es)

сопоставляет списку es показателей некоторое PM, которое покрывает W(es). Это задает отображение \es → pm и еще доказательство соответствующего утверждения об этом отображении. Эта функция устроена следующим образом. Начальное PM (generalPM) состоит из свобод, покрывает все пространство PP и соответствует пустому списку показателей. По рекурсии, если текущий список es отображен в pm, которое покрывает W(es), то тогда (reducePM e pm) покрывает область W(e :: es). Этот шаг индукции доказывается в теле функции esToPM.

Функция

esToPM! : List PP → PMultiset

esToPM! es = proj_1 (esToPM es)

отличается от esToPM тем, что возвращает только PM.

Пример. Зададим три показателя:

$e_0 = [3, 2]$, $e_1 = [1, 5]$, $e_2 = [2, 1]$.

Применяя функцию esToPM! (опирающуюся на сведéние PM), построим три PM, соответствующие лестницам $[e_0]$, $[e_1, e_0]$, $[e_2, e_1, e_0]$:

$pm[e_0]$ = esToPM! [e_0]

$pm\text{-}e_1e_0$ = esToPM! (e_1 :: e_0 :: [])

$pm\text{-}e_2e_1e_0$ = esToPM! (e_2 :: e_1 :: e_0 :: [])

Белая область для лестницы $[e_2, e_1, e_0]$ показана на рис. 2.

Демонстрационная функция (модуль Pol. Dickson.Test нашей программы) этого примера показывает такие значения для этих PM:

```
pm[e₀] = PM [ Lev [*1, *0, 2*, 1*, 0*] ]
pm-e₁e₀ = PM [ Lev [*1, *0, 1*, 0*],
           Lev [24, 23, 22, 21, 20] ]
pm-e₂e₁e₀ = PM [ Lev [*0, 1*, 0*],
      Lev [11, 01, 24, 23, 22, 21, 20]]
```

Согласно алгоритму esToPM! эти три PM получены последовательным применением функции reducePM :

[*, *] –reducePM e_0–> $pm[e_0]$ –reducePM e_1–>

$pm\text{-}e_1e_0$ –reducePM e_2–> $pm\text{-}e_2e_1e_0$

При этом выполнены неравенства

$pm[e_0]$ >pm $pm\text{-}e_1e_0$ >pm $pm\text{-}e_2e_1e_0$.

Белая область $W[e_0]$ представима в виде двух горизонтальных и трех вертикальных лучей. Это в точности выражает значение $pm[e_0]$.

$W[e_1e_0]$ представима двумя горизонтальными и одним вертикальным лучами, и еще несколькими точками. $pm\text{-}e_1e_0$ содержит образец 1*, из-за чего $pm\text{-}e_1e_0$ задает область, строго включающую $W[e_1e_0]$. Это допустимо – условие правильности алгоритма названо ниже.

$W[e_2e_1e_0]$ представима одним горизонтальным и одним вертикальным лучами, и еще четырьмя точками. $pm\text{-}e_2e_1e_0$ задает область, строго включающую $W[e_2e_1e_0]$. Кроме того, в этом PM имеются наложения. Например, образец 1* включает образец 11. На правильность применения метода это не влияет.

Достаточные условия правильности этого метода образцов таковы.

• Очередное PM покрывает соответствующую белую область.

• Очередное PM меньше предыдущего PM в смысле <pm.

Первое из этих свойств следует из леммы о принадлежности показателя сведéнию (доказанной в нашей библиотеке):

```
e∤ e' ∧ e' ∈ pat ∧ e-reduces-pat ⇒ e'∈reduce-e-pat :
                {e e' : PP} {p : Pattern} →
                e ⋠ S' e' → e' ∈pat p → e ≤ ep p →
                Any (e' ∈pat_) (reducePattern e p)
```

– “Если e не делит e’, e’ принадлежит образцу p и e сводит p, то e’ принадлежит некоторому из образцов reducePattern e p”.

6. ТЕОРЕМА О ВПОЛНЕ-ЗАДАННОСТИ ДОПУСТИМОГО УПОРЯДОЧЕНИЯ

Зададим порядок на списках показателей:

```
_<pps_ : Rel (List PP) _
_<pps_ = _<pm_ on esToPM!
```

То есть для списков es и es’ сравниваются два PM, полученные применением esToPM к спискам es и es’ соответственно.

Так как порядок <pm вполне задан, то и <pps вполне задан. Стандартная библиотека весьма просто доказывает это в общем случае, как свойство отображения _on_ на упорядочениях.

Функция

```
e:es<pps-es : ∀e es → All (e <e_) es →
                                (e :: es) <pps es
```

доказывает Лемму:

если показатель e меньше всех показателей из es по допустимому упорядочению $<_e$, то (e :: es) <pps es.

В частности, при добавлении такого показателя спереди к лестнице получается меньшая лестница в смысле <pps.

Доказательство легко следует из свойств функции reducePM, доказанных раньше.

Теорема WF-admissible.

Всякое допустимое упорядочение $<_e$ показателей вполне задано.

Следующая программа на языке Agda формулирует и доказывает эту теорему.

```
<e wellFounded : WellFounded _<e_
<e wellFounded e =
  aux e [] [] (<pps-wellFounded [])
  where
  aux : (e : PP) (es : List PP) →
        All (e <e_) es → Acc _<pps_ es →
        Acc _<e_ e
  aux e es e<es (acc rec-es) = acc rc
    where
    rc : ∀e' → e' <e e → Acc _<e_ e'
    rc e' e'<e =
      aux e' (e :: es) e'<ees (rec-es _ ees<pps-es)
        where
        ees<pps-es = e:es<pps-es e es e<es
        e'<ees =
          e'<e ::
            (AllM.map (\e<x → <e trans e'<e e<x) e<es)
```

Это короткое доказательство выражает формально следующее неформальное рассуждение.

Введем сокращения и обозначения:

ees = e :: es – для показателя e и списка es,

AccE = Acc _$<_e$_ – достижимость показателя по $<_e$,

AccPPs = Acc _<pps_ – достижимость списка показателей по <pps,

e <all es = All (e $<_e$_) es – отношение “ e меньше всех показателей из списка es”.

(определение достижимости (WellFounded, Acc) дано в Разделе 2).

Пусть e – любой показатель. Надо доказать его достижимость AccE e. То есть доказать, что всякий показатель e’ $<_e$ e является достижимым (по определению, это единственный способ доказать достижимость).

Лемма Aux. Для любых показателя e и списка es показателей, если e <all es и если AccPPs es, то AccE e.

Докажем лемму Aux трансфинитной индукцией по отношению <pps для списка es. Индуктивное предположение здесь означает:

лемма Aux выполнена для всех списков es’, для которых es’ <pps es.

Формально это выражается как

(Ind)

```
∀es' → es' <pps es →
        (∀u → u <all es' → AccPPs es' → AccE u)
```

Надо из этих условий вывести AccE e. По определению достижимости, для этого достаточно вывести то, что для любого e’ $<_e$ e выполнено AccE e’.

По лемме e:es<pps-es, из условия e<es : e <all es выводим ees<pps-es : ees <pps es

(слева от двоеточия пишем имя утверждения, справа – выражение его смысла).

Из условий e’ $<_e$ e, e <all es по транзитивности следует e’<ees : e’ <all ees.

В утверждение (Ind) подставляем es’ = ees, во второй аргумент подставляем доказательство ees<pps-es, подставляем u = e’, доказательство e’<ees подставляем в условие u <all es’. Раньше доказано, что отношение <pps является вполне заданным. Поэтому, раз AccPPs es и ees <pps es, то выполнено AccPPs ees.

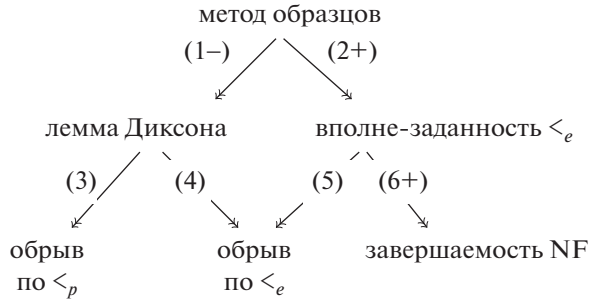


Рис. 3.

По предположению индукции, из этого следует AccE e'.

Лемма доказана.

Доказательство теоремы WF-admissible теперь получается применением леммы Aux к аргументам

e, [], [], (< pps-wellFounded []).

То есть подставляется es = []. Следующее выражение [] предсталяет доказательство того, что e меньше всех элементов пустого списка, а (<pps-wellFounded []) это доказательство достижимости пустого списка относительно упорядочения <pps.

Теперь предлагаем читателю сравнить это рассуждение с текстом функции $<_e$wellFounded.

Главными частями этого доказательства теоремы являются лемма <pps-wellFounded о вполне-заданности отношения <pps на списках показателей (которое довольно сложно устроено) и лемма e:es<pps-es об уменьшении списка es показателей в отношении <pps при добавлении показателя из *белой области* для es (определение дано в Разделе 4). Видимая простота доказательства этой теоремы достигнута тем, что от рассмотрения достижимости показателя e делается переход к рассмотрению подходящих *лестниц* es, и применяются леммы <pps-wellFounded и e:es<pps-e относительно упорядочения на этих лестницах.

Логическая зависимость по конструктивному выводу между рассматриваемыми здесь теоремами показана на рис. 3.

Здесь "обрыв по $\prec$" означает свойство обрывание-убывающей по $\prec$ цепи.

Все эти выводы могут быть выражены конструктивно. Из них знаком '+' отмечены те, которые конструктивно доказаны на языке Agda в нашей библиотеке.

Вклад автора состоит в следующем.

• Воплощение на языке Agda метода образцов из [5] (его видоизменения), при этом замысел метода принадлежит [5].

• Выводы (1+), (2+), (6+). (1+) получается из метода образцов путем введения упорядочения <pps на лестницах и функции доказательства с лестницей в качестве дополнительного аргумента. (2+) приведен выше в этом разделе. (6+) является простым следствием (2+) (Раздел 2.5).

• Воплощение на языке Agda различных определений, алгоритмов и лемм, которые используются в этой работе, но также входят в нашу библиотеку и в силу своей общности применимы во многих других случаях.

Отличие от подхода [9] к завершаемости алгоритма NF нормальной формы состоит в том, что [9] изначально предполагают вполне-заданность упорядочения на мономах, что ведет к неудобствам (Раздел 2, перед 2.1). Здесь же мы привели конструктивное машинно-проверяемое доказательство вполне-заданности всякого допустимого упорядочения мономов – в общем случае.

6.1. О программе доказательства в библиотеке

Точное выражение названных здесь определений, действий, функций доказательств входит в программу `AdmissiblePPO-wellFounded` [4], написанную на языке `Agda`.

Алгебра многочленов запрограммирована в модулях раздела source/Pol/ этой библиотеки, понятие показателя и допустимого упорядочения показателей выражены в модуле source/Pol/PowerProduct.agda.

Модули поддержки доказательства вполне-заданности допустимого упорядочения содержатся в разделе source/Pol/Dickson/.

Перечислим модули этого раздела библиотеки.

I.agda содержит понятия образца, уровня, вводит различные отношения на этих данных.

II.agda содержит функции сведéния образца, уровня, списка уровней, доказательства свойств сведéния для этих данных.

OfPMultiset.agda вводит понятие PMultiset (PM – мультимножество образцов), упорядочение <pm на PM, функцию сведения PM по показателю, доказательства различных свойств связи между этими сущностями, например, вполне-заданности упорядочения <pm.

ForAdmissiblePPO.agda вводит упорядочение <pps на лестницах, доказывает вполне-заданность <pps, вводит функцию esToPM построения PM из лестницы, доказывает, что каждый шаг продолжения лестницы дает меньшую лестницу в смысле <pps, доказывает вполне-заданность $<_e$wellFounded всякого допустимого упорядочения $<_e$ на показателях.

Dickson.agda содержит вывод леммы Диксона путем применения леммы о вполне-заданности упорядочения <pm.

Test.agda содержит показ вычисления последовательности PM по лестнице из трех показателей.

В модуле source/Vector/Sparse/LexOrd.agda воплощено доказательство теоремы о вполне-заданности словарного упорядочения на векторах из $\mathbb{N}^n$ для установленной размерности n для разреженного представления вектора. Главная часть всех вышеприведенных построений все-таки сводится к этой теореме.

В модуле source/Pol/NormalForm/PolNF.agda воплощена доказательная функция polNF нормальной формы относительно списка многочленов.

Библиотека проверена в системе `Agda 2.6.2.2, ghc-9.2.1, MAlonzo, Ubuntu Linux 18.04`.

Она должна работать в любой системе, где работает `Agda 2.6.2.2, MAlonzo`.

Файл install.txt содержит инструкцию по сборке программы в системе `Agda`. Команды

```
> cd source
> agda $agdaLibOpt $agdaFlags
      TypeCheckAll.agda
```

производят проверку типов всей библиотеки, в частности проверяются все доказательства.

Затем команда

```
> agda -c $agdaLibOpt $agdaFlags
Pol/Dickson/Test.agda
```

собирает исполняемый модуль для программы показа построения РМ по лестнице.

Затраты на сборку: на машине частоты 3 GHz с 8 Gb оперативной памяти каждый из этих двух вызовов ‘agda’ занимает меньше 5 минут.

Затем команда `> ./Test`

быстро вычислит три РМ по лестнице, заданной в `Test.agda`, и распечатает их (Пример из Раздела 5.1).

7. ЗАКЛЮЧЕНИЕ

Представлено формальное конструктивное машинно-проверяемое доказательство на языке Agda свойства вполне-заданности (well-founded в конструктивном определении) произвольного допустимого упорядочения $<_e$ показателей мономов многочленов нескольких переменных [8]. Оно основано на методе образцов из [5].

Нам не известны другие работы, в которых (конструктивно) доказана эта теорема.

Доказательство представлено библиотекой `AdmissiblePPO-wellFounded` [4] доказательных, разработанной автором.

На основе этой теоремы на языке Agda составлена доказательная программа алгоритма нормальной формы для многочленов [8], теорема обеспечивает простое доказательство завершаемости.

8. БЛАГОДАРНОСТИ

• Автор благодарен Антонине Н. Непейводе и Андрею П. Немытых за их замечания по содержанию статьи.

• Автор благодарен участникам семинара “Компьютерная алгебра” (факультет ВМК МГУ, руководитель С.А. Абрамов) за их обсуждение доклада по предмету этой статьи.

• Исследование частично поддержано Министерством науки и высшего образования РФ, госзадание 122012700089-0.

СПИСОК ЛИТЕРАТУРЫ

1. *Гаранина Н.О.* Общие знания в хорошо структурированных системах с абсолютной памятью // Модел. и анализ информ. Систем. 2013. Т. 20. № 6. С. 10–21.
2. *Ершов Ю.Л., Палютин Е.А.* Математическая логика. 6-е изд., испр. М.: ФИЗМАТЛИТ, 2011. 356 с. ISBN 978-5-9221-1301-4.
3. *Мешвелиани С.Д.* О зависимых типах и интуиционизме в программировании математики // В электронном журнале Программные системы: теория и приложения. 2014. Т. 5. Вып. 3. С. 27–50. http://psta.psiras.ru/read/psta2014_3_27-50.pdf
4. *Мешвелиани С.Д.* AdmissiblePPO-wellFounded – программа на языке Agda формального конструктивного доказательства леммы Диксона и теоремы о вполне-заданности допустимого упорядочения на мономах многочленов. Институт программных систем РАН, Переславль-Залесский, 2022, www.botik.ru/pub/local/Mechveliani/inAgda/admissiblePPO-wellFounded.zip
5. *Martin-Mateos F.J., Alonso J.A., Hidalgo M.J., Ruiz-Reina J.L.* A Formal Proof of Dickson’s Lemma in ACL2. M.Y. Vardi and A. Voronkov (Eds.): LPAR 2003, LNAI 2850. 2003. P. 49–58.
6. Agda. A proof assistant. A dependently typed functional programming language and its system. http://wiki.portal.chalmers.se/agda/pmwiki.php.
7. *Norell U.* Dependently Typed Programming in Agda. AFP 2008: Advanced Functional Programming, Lecture Notes in Computer Science, vol. 5832, Springer, Berlin–Heidelberg, 2008. P. 230–266.
8. *Бухбергер Б.* Базисы Грёбнера. Алгоритмический метод в теории полиномиальных идеалов. В сборнике “Компьютерная алгебра”, редакторы Б. Бухбергер, Дж. Коллинз, Р. Лоос. Перевод с английского. Москва, МИР, 1986. С. 331–372.
9. *Coquand Th. and Persson H.* Gröbner Bases in Type Theory. 1998. 13 p. https://www.researchgate.net/publication/221186683_Grobner_Bases_in_Type_Theory
10. *Théry L.* A Machine-Checked Implementation of Buchberger’s Algorithm // Journal of Automated Reasoning. 2001. V. 26. P. 107–137.
11. *Romanenko S.A.* Proof of Higman’s Lemma (for two letters) Formalized in Agda. In Russian. Июль 2017. https://pat.keldysh.ru/roman/doc/talks/2017_Romanenko__Higman’s_lemma_for_2_letters_in_Agda_ru__slides.pdf Agda program for the proof:

https://github.com/sergei-romanenko/agda-Higman-lemma, in the folder Berghofer.

12. *Robbiano L.* Term orderings on the polynomial ring. Proc. EUROCAL '85 European Conference on Computer Algebra. Linz 1985, Springer Leer. Notes Comp. Sei. 1986. V. 204. P. 513–517.
13. *Curry H.B., Feys R.* Combinatory Logic, vol I. Amsterdam, North-Holland, 1958. 417 p.
14. *Howard W.A.* The formulae-as-types notion of construction. To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. Boston, Academic Press, 1980. P. 479–490.
15. *Марков А.А.* О конструктивной математике. Проблемы конструктивного направления в математике. 2. Конструктивный математический анализ, Сборник работ, Тр. МИАН СССР, 67, Изд-во АН СССР, М.–Л., 1962. С. 8–14.
16. *Per Martin-Loef.* Intuitionistic type theory. Bibliopolis, ISBN 88-7088-105-9. 1984. 91 p.
17. *Stricland Neil P.* Euclid's theorem. An annotated proof in Agda that there are infinitely many primes. https://nextjournal.com/agda/euclid-theorem
18. *Vytiniotis D., Coquand Th., Wahlstedt D.* Stop When You Are Almost Full. Adventures in Constructive Termination // ITP 2012: Interactive Theorem Proving. 2012. P. 250–265.

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 21–26

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.451.3,519.683.4,519.686.2,004.94

ПРИМЕНЕНИЕ В GINV ДИНАМИЧЕСКОГО ПЕРЕРАСПРЕДЕЛЕНИЯ ПАМЯТИ

© 2023 г. Ю. А. Блинков[a,b,*], Е. Ю. Щетинин[c,]**

[a]*Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского*
410012 Саратов, ул. Астраханская, д. 83, Россия
[b]*Российский университет дружбы народов (РУДН)*
117198 Москва, ул. Миклухо-Маклая, д. 6, Россия
[c]*Финансовый университет при Правительстве Российской Федерации*
125167 Москва, Ленинградский пр-т, д. 49, Россия
**E-mail: blinkovua@info.sgu.ru*
***E-mail: riviera-molto@mail.ru*

Поступила в редакцию 01.07.2022 г.
После доработки 29.07.2022 г.
Принята к публикации 21.10.2022 г.

Представлена новая версия GInv (Gröbner Involutive) по вычислению инволютивных базисов Грёбнера в виде библиотеки на языке *C*++11. В GInv для динамических структур данных, таких как списки, красно-черные и бинарные деревья, библиотеки GMP для вычислений с целыми числами с произвольной точностью использовано объектно-ориентированное перераспределение памяти. Интерфейс пакета оформлен в виде модуля языка Python3.

DOI: 10.31857/S0132347423020061, **EDN:** GZFUDG

1. ИНВОЛЮТИВНЫЕ БАЗИСЫ ГРЁБНЕРА

Понятие базиса Грёбнера было введено Бруно Бухбергером в 1965 году [1]. Основой созданного им алгоритма является понятие *S-полинома*. Альтернативный подход к созданию конструктивных алгоритмов с помощью разбиения переменных на *мульти* и *немультикативные* переменные развит в работах [2, 3]. инволютивный подход пришел в компьютерную алгебру из теории линейных уравнений в частных производных, где метод приведения системы в инволюцию использовался уже в начале и середине XX века (работы Жане, Рикье, Томаса, Поммаре).

Во всех версиях алгоритмов построения базисов Грёбнера приходится иметь дело со структурами данных, которые меняются в ходе вычислений. Например, для представления полиномов могут использоваться списки, красно-черные деревья или строка в разряженной матрице [4].

Для вычислений с целыми числами с произвольной точностью также используют списки или массивы переменой длины состоящие из машинных слов, которые играют роль *цифр*, если рассматривать аналогию с обычными десятичными цифрами.

В настоящее время все современные системы компьютерной алгебры имеют встроенные пакеты построения базисов Грёбнера. Поскольку алгоритмы его построения имеют экспоненциальную сложность как по времени, так и требуемой памяти ее фрагментация является серьезной проблемой.

При работе пакета *INVBASE* (`http://www.reduce-algebra.com/manual/manualse151.html`), который инволюционными методами строит базис Грёбнера [2] для систем полиномиальных уравнений, в системе компьютерной алгебры *Reduce* было замечено, что в некоторых задачах процент работы сборки мусора составлял 8–10% от общего времени программы.

В предыдущей версии пакета *GInv* для построения инволютивных базисов Грёбнера [5] написанном на *C*++ при задачах, считающихся длительное время, обнаруживались значительные временные затраты на выделение/освобождение памяти.

Одним из возможных способов борьбы с этой проблемой является перезапуск программы. Другими словами программа в какой-то момент времени, выбор которого очень не тривиальная задача, сохраняет данные на диск в простом виде или используя сериализацию. Затем при перезапуске происходит чтение сохраненных данных, что во первых устраняет фрагментацию, а во вторых *соседние* данные располагаются, как правило, на одной странице памяти, что оптимизируют их для

использования кэша процессора. В результате на некоторых задачах компьютерной алгебры ускорение может возрасти в разы и даже больше, несмотря на затраты для *записи/чтения*.

В связи с вышесказанным возникает необходимость в перераспределении памяти.

2. ПРОБЛЕМА ПЕРЕРАСПРЕДЕЛЕНИЯ ПАМЯТИ

В современных языках программирования все переменные можно разделить на три категории по времени их *жизни* в программе:

• все время *жизни*, так называемая *статическая память*, а переменные, использующие эту память, часто называют *статическими*. Сюда также попадают и *глобальные*;

• время выполнения функции, память выделяется в стеке, а переменные называют *автоматическими*, но в языках программирования используют слово *локальные*;

• время *жизни* определяет программист, если нет *сборки мусора* (garbage collection), а память выделяется в *куче*. Или определяет алгоритм *сборки мусора*, если она есть в выбранном языке программирования. Иногда поддерживаются оба варианта работы.

Сборка мусора была впервые применена Джоном Маккарти в 1959 году в среде программирования на разработанном им функциональном языке программирования *Лисп* (Lisp), а сам термин возник в сноске статьи [6].

Если память в *куче* выделяется разного размера, то несмотря на возможные алгоритмы слияния соседних свободных фрагментов, возникают области памяти, которые не используются. Джоном Маккарти в языке программирования *Лисп* было предложено очень красивое решение. Все лисповские ячейки одного размера, а списками (точнее *списками списков*) можно представить любые структуры данных.

В настоящее время существует несколько базовых алгоритмов *сборки мусора* и их огромное число модификаций. Например, для языка *Java* разработчики обычно поставляют сразу несколько вариантов алгоритмов *сборки мусора* для разных вариантов использования программ написанных на *Java*.

Можно выделить основные критические моменты при использовании *сборки мусора*:

• *сборка мусора* потребляет вычислительные ресурсы для принятия решения о том, какую память освободить, даже если программист, возможно, уже знал эту информацию;

• момент, когда мусор действительно будет собран, может быть непредсказуемым, что приводит к остановкам (паузы для сдвига/освобождения памяти), разбросанных по всему сеансу. Инкрементные, параллельные *сборщики мусора* и *сборщики мусора* в реальном времени решают эти проблемы с различными компромиссами;

• самое главное: *сборка мусора* не учитывает *защищенный режим*, *страничную организацию памяти* и основу современных процессоров – *кэш* (сверхоперативную память).

Рассмотрим для сравнения реализации *malloc/free* для языко *C/C++*:

• В 2020 году компания Google представила новый вариант системы распределения памяти *TCMalloc*, которая используется во многих внутренних проектах Google. Для работы требуется наличие компилятора с поддержкой *C++17* для языка *C++*, и *C11* для языка *C* (gcc 9.2+ или clang 9.0+). Из операционных систем поддерживается только Linux (x86, PPC).

Также, с 2005 года существует ещё один вариант *tcmalloc*, который поставлялся в составе пакета *gperftools* (Google Performance Tools).

• В 2021 году выпущен *mimalloc* бесплатный компактный менеджер памяти общего назначения с открытым исходным кодом, разработанный Microsoft с акцентом на характеристики производительности. Библиотека составляет около 11000 строк кода и работает как замена *malloc* стандартной библиотеки *C* и не требует дополнительных изменений кода.

• *jemalloc* реализация *malloc* общего назначения, для предотвращения фрагментации и поддержки масштабируемого параллелизма. *jemalloc* впервые начал использоваться в качестве распределителя libc FreeBSD в 2005 году.

Об остроте рассматриваемой здесь проблемы, также говорит множество докладов [7–11] по работе с динамическим перераспределением памяти на конференциях ISMM (`http://www.sigplan.org/Conferences/ISMM/`).

3. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПЕРЕРАСПРЕДЕЛЕНИЕ ПАМЯТИ

Перечислим основные концепции предлагаемого подхода

• Предлагаемый подход принципиально отличается от вышеизложенных и основан на реализации ООП (Объектно-ориентированное программирование) в *C++*.

• Ограничить использование указателей на внутреннюю структуру класса *C++* объявив их в закрытой области видимости. При правильной технологии ООП, а это свойство называется *инкапсуляцией*, это и так необходимо делать, пользователь не должен видеть как реализован класс внутри, он должен использовать только его интерфейс.

• В *C++* имеется выделенный *конструктор копирования*, который позволяет построить копию объекта. Осталось для выбранного класса определить функцию *swap*, для подмены объекта на его копию, и можно организовывать динамическое перераспределение памяти на *C++*.

В качестве примера использования рассмотрим простейший класс на версии языка *C++11* с динамическим выделением памяти – линейный однонаправленный список целых чисел.

```
1 #include “allocator.h”
2
3 class List {
4   struct Node {
5     int  mData;
6     Node* mNext;
7
8     Node(int data, Node* next=nullptr):
9        mData(data),
10       mNext(next) {
11    }
12    ~Node() {};
13  };
14
15  Allocator* mAllocator;
16  Node* mHead;
17
```

Основное отличие от обычного кода строчка 15, где содержится объявление переменной указателя на тип *Allocator*. В строках 23 и 27 показано его использование для выделения и освобождения памяти. Освобождение использует шаблонную функцию *destroy*, поскольку перегрузку стандартого *delete* дополнительными аргументами позволяют не все компиляторы *C++*. Для базовых типов и классов, для которых в принципе не нужен вызов деструктора, определена шаблонная функция *dealloc*. *destroy* и *dealloc* могут также дополнительно вызываться с дополнительным целочисленным аргументом для удаления массивов.

```
18 public:
19   List()=delete;
20   List(Allocator *allocator):
21   …
22   List(const List& a)=delete;
23   List(const List& a, Allocator *allocator):
24   …
25     *mLink = new(mAllocator) Node(i,
26                                   *mLink);
27   …
28     auto* tmp=*mLink;
29     *mLink = tmp->mNext;
30     mAllocator->destroy(tmp);
31   …
32   void swap(List &a) {
33     auto *tmp1=mAllocator;
34     mAllocator = a.mAllocator;
35     a.mAllocator = tmp1;
36
37     auto* tmp2=mHead;
38     mHead = a.mHead;
39     a.mHead = tmp2;
40   }
41   …
42 };
```

В строке 19 фактически объявлен запрет на использование конструктора по умолчанию, его роль теперь играет конструктор объявленный в строке 20.

Как и для *конструктора по умолчанию* в строке 22 объявлен запрет на использование стандартного *конструктора копирования*. В строчках 23 и 32 содержатся заголовки нового *конструктора копирования* и функции *swap*, которые необходимы для замены текущего объекта на его копию. Если Вы хотите для своего класса использовать оператор =, то его тоже необходимо реализовать. Он, как известно, не наследуется, но его поддержка в классе шаблона *GC* имеется. Ниже дан пример его использования.

```
43 typedef GC<List> GCList;
44 …
45 GCList a;
46 …
47 a.reallocate();
48 …
```

На строке 43 объявляем новый тип *GCList*. Затем вызываем на строке 45 конструктор по умолчанию. Предположим на строке 46 производим массовые операции по выделению/освобождению памяти. Память выделяется блоками, кратными размеру страницы памяти, затем раздается внутри конкретного объекта класса *List*. Если процент не используемой памяти высок, то вызов *reallocate* на строке 45 приведет к построению копии объекта, с его последующей заменой на первоначальную переменную a с использованием swap внутри тела функции *reallocate*.

Естественно, вызов *reallocate* может происходит неоднократно, согласуясь с логикой программы. Также, за счет показанных в следующем разделе очень простых решений попутно выполняются учет работы времени по перераспределению памяти, выделанная в данный момент времени память, максимальная память и в *debug*-версии программы проверки на утечки памяти. Выделение памяти большими блоками очень выгодно, особенно для большего количества мелких объек-

Таблица 1.

	T	GC	GC/T	M
cyclic7				
Allocator	93.48	3.83	4.1%	172
Allocator (кроме GMP)	80.89	0.28	0.35%	180
malloc/free	119.47	–	–	124
Задача Попова 21 [13]				
Allocator	505.37	2.52	0.50%	1231
Allocator (кроме GMP)	487.59	0.64	0.13%	559
malloc/free	495.81	–	–	609
eco11				
Allocator	871.75	2.31	0.27%	797
Allocator (кроме GMP)	611.21	0.86	0.14%	243
malloc/free	924.41	–	–	186

тов. Стандартные *new*, *delete*, *malloc*, *free* должны хранить на каждое выделение информацию о начале и конце выделенной области памяти. В данном подходе этого не нужно, поскольку конкретный объект может с помощью конструктора-копирования построить свою копию и обратно вернуть используемую память в операционную систему. Замечу, что предлагаемый подход, позволяет собирать объект из *кирпичиков* с общим *Allocator*-ом. Лишь бы в каждом инкапсулированом объекте память выделялась/освобождалась с помощью указателя на общий *Allocator* и были определены соответствующие *конструкторы-копирования* и функция *swap*. Можно также, как показано ниже, определять *Allocator* вначале блока и затем его использовать для работы с автоматическими переменными, использующими *Allocator*.

```
1 {
2   Allocator a [1];
3   …
4   List lst(a);
5   …
6 }
```

После закрытия блока вся занятая ими память будет возвращена в систему. Это хорошо имитирует нестандартную функцию *alloca* из многих Unix подобных систем, которая выделяет память из стека с последующим автоматическим ее возвращением в стек при завершении блока с локальными переменными. При этом в отличии от *alloca* и фактически при тех же затратах по требуемой памяти и времени выполнения можно выделять для сложных объектов, а не только для встроенных типов данных и массивов.

Внутри класса *Allocator* используется набор статических элементов класса для ведения статистики: затраты машинного времени по перераспределению памяти, выделенная в данный момент времени память, максимальная память. Переменные класса хранят информацию по выделанной в данный момент времени и используемой памяти для конкретного объекта класса *Allocator*. При этом размер разовой выделяемой памяти выравнивается на границу кратную кэшу данного компьютера и выставляется вручную при компиляции библиотеки.

При возврате памяти в классе *Allocator* память физически не возвращается в систему, единственно уменьшается переменная используемой памяти. При вызове деструктора класса *Allocator* в *debug* версии программы происходит проверка, что счетчик используемой памяти равен 0.

Сама память представляет собой односвязный список из блоков памяти кратный странице памяти компьютера. Блоки для переносимости библиотеки выделяются стандартной функцией *malloc*. Возврат памяти в систему происходит при вызове деструктора с помощью последовательных вызовов функции *free*. Структура классов и их основные атрибуты и методы показаны на рисунке.

За счет директивы макросов языка *C++* при сборке можно собрать класс *Allocator*, который фактически только вызывает стандартные *malloc/free*. Это бывает удобно для сравнения и отладки.

Размер обоих реализаций составляет в сумме примерно 320 строчек кода со всеми комментариями и пробельными строками и расположен на GitHub по адресу `https://github.com/blinkovua/GInv/tree/master/util`.

4. ИСПОЛЬЗОВАНИЕ В GInv

Объектно-ориентированное перераспределение памяти в новой версии *GInv* (`https://github.com/blinkovua/GInv`) используется для объектов, которым приходится многократно перестраиваться в памяти. Рассмотрим стандартный пример представления полинома в виде упорядоченного списка мономов и их коэффициентов. Для сохранения порядка мономы с их коэффициентами часто меняются местами, при равенстве нулю коэффициента происходит их удаление, а деление полинома на его *содержание* приводит к изменению всех коэффициентов.

В настоящий момент в *GInv* все структуры данных, которые меняются в процессе построения инволютивного базиса Грёбнера имеют динамическое перераспределение памяти. В качестве наиболее характерного примера можно привести бинарные деревья специального вида – деревья Жане [12].

Библиотека *GMP* (`https://gmplib.org/`) для вычислений с целыми числами с произвольной точностью не позволяет на прямую воспользоваться классом *Allocator* пакета *GInv*. Для нее была сделана *обертка*, которая заранее вычисляет

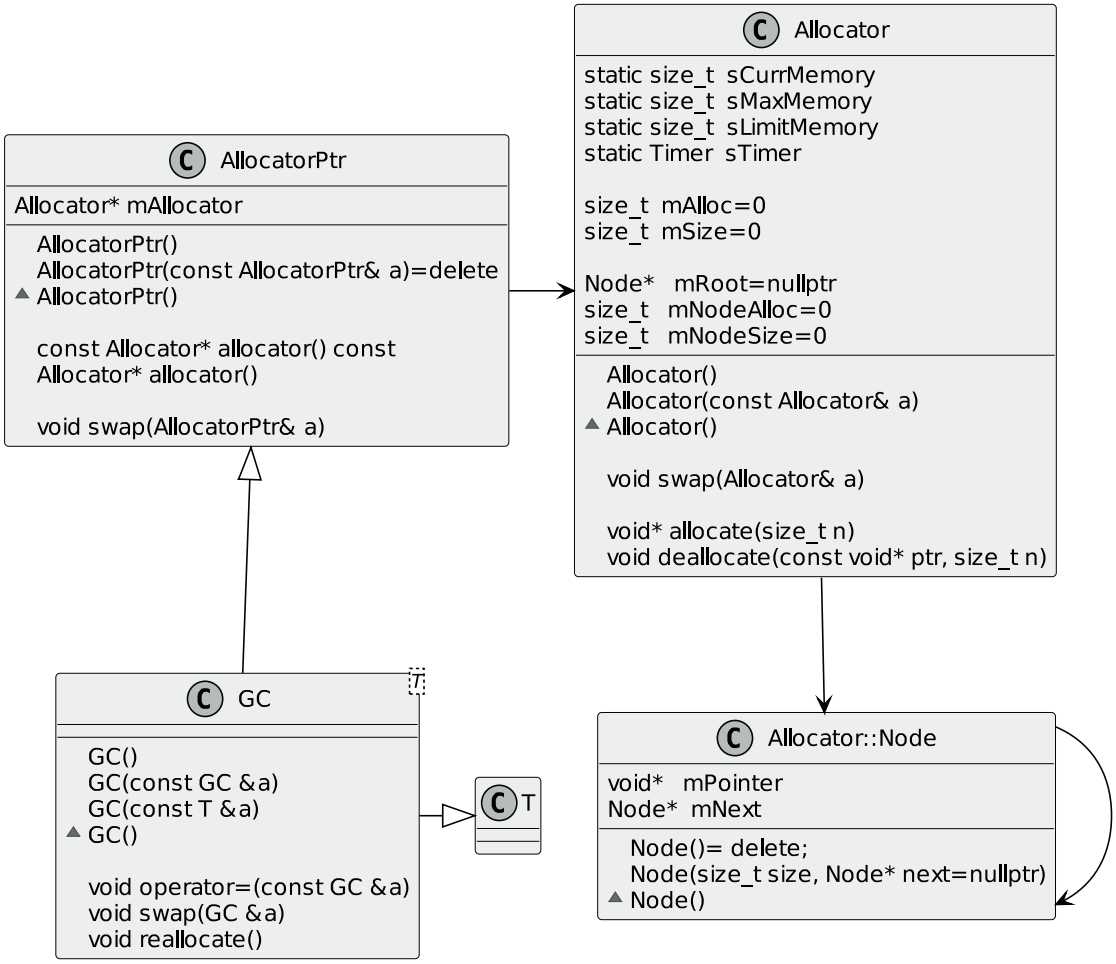


Рис. 1.

требуемый объем памяти и выделяет ее используя класс *Allocator* для проведения операций *сложения/вычитания*, *умножения/деления*, возведения в степень и вычисления наибольшего делителя.

Результаты представлены в таблице 1 при вычислении на процессоре *Intel(R) Xeon(R) CPU L5630 @ 2.13GHz, cache size: 12288 KB* в операционной системе Debian GNU/Linux 11. Столбцы таблицы: T – время в секундах, GC – затраченное время на динамическое перераспределение памяти и составляет часть T, GC/T – процент от общего времени. По середине таблицы условное название примера, M – максимальная память в мегабайтах (MiB), измеренная системной утилитой *time*.

5. ЗАКЛЮЧЕНИЕ

Как видно из результатов вычислений в некоторых случаях заметно значительное ускорение программ, кроме *Задача Попова 21* [13], что связанно с тем, что внутри этой задачи размеры чисел очень велики, а остальные структуры памяти практически не используются, а имеются лишь затраты на лишнее копирование чисел при их динамическом перераспределении памяти.

Предложенный подход практически лишен недостатков стандартных *malloc/free*, так и подхода с использованием *сборки мусора*. Хорошо использует страничную организацию памяти и кэш процессора. Очень прост в реализации по сравнению со стандартными *malloc/free* и позволяет эффективно находить ошибки, связанные с *утечками* памяти.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Программы стратегического академического лидерства РУДН.

СПИСОК ЛИТЕРАТУРЫ

1. *Buchberger B.* Gröbner bases: an Buchberger algorithmic method in polynomial ideal theory // Recent Trends in Multidimensional System Theory / Ed. by *N.K. Bose*. V. 6. Reidel, Dordrecht, 1985. P. 184–232.

2. *Жарков А.Ю., Блинков Ю.А.* Инволютивные системы алгебраических уравнений // Программирование. 1994. № 1. С. 53–56.
3. *Gerdt V.P., Blinkov Yu.A.* Minimal involutive bases // Mathematics and Computers in Simulation. 1998. V. 45. P. 543–560.
4. *Faugère J.-C.* A new efficient algorithm for computing Gröbner bases (F4) // Journal of Pure and Applied Algebra. V. 139 (1–3). 1999. P. 61–88.
5. *Блинков Ю.А., Гердт В.П.* Специализированная система компьютерной алгебры GINV // Программирование. 2008. № 2. С. 67–80.
6. *McCarthy J.* Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I // Commun. ACM, 1960. № 4. P. 184–195.
7. *Bansal A., Goel S., Shah P., Sanyal A., Kumar P.* Garbage Collection Using a Finite Liveness Domain // Proceedings of the 2020 ACM SIGPLAN ISMM, 2020. P. 1–15.
8. *Yang A.M., Österlund E., Wilhelmsson J., Nyblom H., Wrigstad T.* ThinGC: Complete Isolation with Marginal Overhead //Proceedings of the 2020 ACM SIGPLAN ISMM, 2020. P. 74–86.
9. *Onozawa H., Ugawa T., Iwasaki H.* Fusuma: Double-Ended Threaded Compaction //Proceedings of the 2021 ACM SIGPLAN ISMM, 2021. P. 94–106.
10. *Tripp C., Hyde D., Grossman-Ponemon B.* FRC: A High-Performance Concurrent Parallel Deferred Reference Counter for C++ // Proceedings of the 2018 ACM SIGPLAN ISMM, 2018. P. 14–28.
11. *Seyri A., Pan A., Vamanan B.* MemSweeper: Virtualizing Cluster Memory Management for High Memory Utilization and Isolation //Proceedings of the 2022 ACM SIGPLAN ISMM, 2022. P. 15–28.
12. *Гердт В.П., Янович Д.А., Блинков Ю.А.* Быстрый поиск делителя Жане // Программирование. 2001. № 1. С. 32–36.
13. *Попов А.С.* Кубатурные формулы на сфере, инвариантные относительно группы вращений икосаэдра // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. Новосибирск, 2008. Т. 11. № 4. С. 433–440.

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 27–32

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ПОСТОЯННОГО МОМЕНТА НА ПОЛОЖЕНИЯ РАВНОВЕСИЯ СПУТНИКА НА КРУГОВОЙ ОРБИТЕ С ПРИМЕНЕНИЕМ МЕТОДОВ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

© 2023 г. С. А. Гутник[a,b,*], В. А. Сарычев[c,]**

[a] *МГИМО МИД России*
119454, Москва, Проспект Вернадского, 76, Россия
[b] *Московский физико-технический институт, МФТИ (НИУ)*
141701, Долгопрудный, Институтский переулок, 9, Россия
[c] *Институт прикладной математики им. М.В. Келдыша РАН*
125047, Москва, Миусская пл., дом 4, Россия
**E-mail: s.gutnik@inno.mgimo.ru*
***E-mail: vas31@rambler.ru*

Поступила в редакцию 20.07.2022 г.
После доработки 14.08.2022 г.
Принята к публикации 30.10.2022 г.

С использованием методов компьютерной алгебры проведено исследование положений равновесия спутника, движущегося по круговой орбите под действием гравитационного и постоянного моментов. Основное внимание уделено исследованию положений равновесия для случаев, когда вектор постоянного момента параллелен плоскостям, образуемым главными центральными осями инерции спутника. С использованием методов построения базисов Гребнера проведена редукция системы шести алгебраических уравнений, определяющих равновесные ориентации спутника, к одному алгебраическому уравнению шестого порядка от одной неизвестной. Проведена классификация областей с равным числом положений равновесия с применением алгебраических методов построения дискриминантных гиперповерхностей. Построены бифуркационные кривые в пространстве параметров задачи, которые задают границы областей с равным числом положений равновесия спутника. Выполнен сравнительный анализ влияния выбора порядка переменных при построении базисов Гребнера для решения рассматриваемой задачи. С использованием предложенного подхода показано, что спутник с неравными главными центральными моментами инерции при действии постоянного момента имеет на круговой орбите не более 24 положений равновесия.

DOI: 10.31857/S0132347423020103, **EDN:** MFVDTU

1. ВВЕДЕНИЕ

В работе представлены результаты применения методов компьютерной алгебры для исследования положений равновесия спутника, на который кроме гравитационного момента действует постоянный в связанной со спутником системе координат момент (далее постоянный момент), обусловленный, например, истечением газа или рабочего тела из его корпуса. Влияние на спутник постоянного момента изменяет его ориентацию и может вывести систему из заданного положения равновесия.

Исследование движения спутника в центральном ньютоновом силовом поле по круговой орбите под действием гравитационного и постоянного моментов представляет практический интерес для создания систем управления ориентацией искусственных спутников Земли. Важным свойством гравитационных систем ориентации является возможность функционировать на орбите продолжительное время без расходования энергии и рабочего тела.

Задаче определения положений равновесия спутника под действием гравитационного и постоянного моментов посвящено много работ. В [1] была показана возможность существования положений равновесия спутника под действием гравитационного и постоянного моментов для некоторых частных случаев.

Общий случай задачи был впервые изучен в [2], где представлены результаты символьно-численного исследования положений равновесия спутника, подверженного действию гравитационного и постоянного моментов. Показано, что равновесия спутника определяются действительными корнями алгебраического уравнения ше-

стой степени и возможно существование не более 24 положений равновесия в орбитальной системе координат. Алгебраическое уравнение, определяющее равновесия спутника, было получено с использованием понятия результанта.

В [3] с использованием подхода работы [2] был проведен детальный анализ эволюции областей существования различного числа положений равновесия спутника в пространстве параметров задачи. В [4, 5] была показана возможность применения методов компьютерной алгебры для исследования свойств системы нелинейных алгебраических уравнений, определяющих положения равновесия спутника, подверженного действию гравитационного и постоянного моментов. С использованием алгоритмов построения базисов Гребнера [6, 7] и понятия результанта было получено более простое по сравнению с [2] алгебраическое уравнение, определяющее положения равновесия спутника.

В настоящей статье проводится подробное исследование положений равновесия спутника, когда вектор постоянного момента параллелен плоскостям, образуемым главными центральными осями инерции спутника с применением алгебраических методов, которые ранее успешно использовались для изучения положений равновесия спутника-гиростата в [8, 9], равновесий спутника при влиянии аэродинамического момента [10] и динамики движения связки двух тел [11, 12]. Рассмотрение более простых случаев по сравнению с общим случаем [2] позволяет получить более простые решения задачи, позволяющие качественно оценить действие постоянного момента на равновесные ориентации спутника и более наглядно продемонстрировать возможности применения методов компьютерной алгебры для решения задач механики космического полета.

Для исследования применялась система компьютерной алгебры Wolfram Mathematica 12.1 [13, 14]. Системы компьютерной алгебры широко используются при решении задач небесной механики, в частности, при исследовании задачи динамики многих тел. Так, с применением символьных вычислений были получены новые результаты при рассмотрении классической задачи трех тел с переменными массами в общем случае и поиске положений равновесия в ограниченной задаче четырех тел в работах [15–18].

2. УРАВНЕНИЯ ДВИЖЕНИЯ

Рассмотрим движение спутника – твердого тела относительно центра масс на круговой орбите под действием гравитационного момента. Пусть на спутник, кроме гравитационного момента, действует постоянный момент, обусловленный, например, истечением газа из корпуса спутника. Для записи уравнений движения введем две правые прямоугольные системы координат с началом в центре масс O спутника. Орбитальную систему координат $OXYZ$, ось OZ которой направлена вдоль радиуса–вектора, соединяющего центры масс Земли и спутника, ось OX направлена вдоль вектора линейной скорости центра масс O спутника. Тогда ось OY будет направлена вдоль нормали к плоскости орбиты. Связанную со спутником систему координат $Oxyz$; здесь Ox, Oy, Oz – главные центральные оси инерции спутника.

Определим ориентацию системы координат $Oxyz$ относительно орбитальной системы координат с использованием самолетных углов тангажа α, рыскания β и крена γ. Направляющие косинусы осей Ox, Oy, Oz в орбитальной системе координат выражаются через самолетные углы с помощью соотношений [19]:

$$
\begin{gathered}
a_{11} = \cos(x,X) = \cos\alpha\cos\beta,\\
a_{12} = \cos(y,X) = \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma,\\
a_{13} = \cos(z,X) = \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma,\\
a_{21} = \cos(x,Y) = \sin\beta,\\
a_{22} = \cos(y,Y) = \cos\beta\cos\gamma,\\
a_{23} = \cos(z,Y) = -\cos\beta\sin\gamma,\\
a_{31} = \cos(x,Z) = -\sin\alpha\cos\beta,\\
a_{32} = \cos(y,Z) = \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma,\\
a_{33} = \cos(z,Z) = \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma.
\end{gathered}
\tag{1}
$$

При малых колебаниях спутника углу тангажа соответствует поворот вокруг оси OY, углу рыскания – поворот вокруг оси OZ, углу крена – поворот вокруг оси OX.

Тогда уравнения движения спутника относительно центра масс запишутся в виде [19]:

$$
\begin{gathered}
A\dot p + (C-B)qr - 3\omega_0^2(C-B)a_{32}a_{33} - \bar a = 0,\\
B\dot q + (A-C)rp - 3\omega_0^2(A-C)a_{33}a_{31} - \bar b = 0,\\
C\dot r + (B-A)pq - 3\omega_0^2(B-A)a_{31}a_{32} - \bar c = 0,
\end{gathered}
\tag{2}
$$

$$
\begin{gathered}
p = \bar p + \omega_0 a_{21}, \quad \bar p = \dot\alpha a_{21} + \dot\gamma,\\
q = \bar q + \omega_0 a_{22}, \quad \bar q = \dot\alpha a_{22} + \dot\beta\sin\gamma,\\
r = \bar r + \omega_0 a_{23}, \quad \bar r = \dot\alpha a_{23} + \dot\beta\cos\gamma.
\end{gathered}
\tag{3}
$$

В уравнениях (2), (3) A, B, C – главные центральные моменты инерции спутника; p, q, r и $\bar a$, $\bar b$, $\bar c$ – проекции абсолютной угловой скорости и проекции вектора постоянного момента спутника на оси Ox, Oy, Oz; ω_0 – угловая скорость движения центра масс спутника по круговой орбите. Точкой обозначено дифференцирование по времени t.

3. ПОЛОЖЕНИЯ РАВНОВЕСИЯ СПУТНИКА

Введем обозначения $a = \bar{a}/\omega_0^2(C-B)$, $b = \bar{b}/\omega_0^2(A-C)$, $c = \bar{c}/\omega_0^2(B-A)$ и положим в уравнениях (2) и (3) $\alpha = \alpha_0 = \text{const}$, $\beta = \beta_0 = \text{const}$, $\gamma = \gamma_0 = \text{const}$, тогда получим при $A \neq B \neq C$ уравнения

$$\begin{aligned} a_{22}a_{23} - 3a_{32}a_{33} - a &= 0, \\ a_{23}a_{21} - 3a_{33}a_{31} - b &= 0, \\ a_{21}a_{22} - 3a_{31}a_{32} - c &= 0, \end{aligned} \tag{4}$$

позволяющие определить положения равновесия спутника в орбитальной системе координат. С учетом (1) систему (4) можно рассматривать как систему трех уравнений с неизвестными α_0, β_0 и γ_0. Исследовать решения такой тригонометрической системы не представляется возможным.

Другой более удобный для исследования способ замыкания уравнений (4) заключается в добавлении условий ортогональности направляющих косинусов

$$\begin{aligned} a_{21}^2 + a_{22}^2 + a_{23}^2 - 1 &= 0, \\ a_{31}^2 + a_{32}^2 + a_{33}^2 - 1 &= 0, \\ a_{21}a_{31} + a_{22}a_{32} + a_{23}a_{33} &= 0. \end{aligned} \tag{5}$$

Уравнения (4) и (5) образуют замкнутую алгебраическую систему уравнений относительно шести направляющих косинусов, определяющих положения равновесия спутника. Для этой системы уравнений ставится следующая задача: при заданных a, b, c определить все девять направляющих косинусов, т.е. все положения равновесия спутника в орбитальной системе координат. После нахождения шести направляющих косинусов a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33} оставшиеся направляющие косинусы a_{11}, a_{12}, a_{13}, с учетом того, что каждый элемент направляющих косинусов равен своему алгебраическому дополнению, можно определить из условий ортогональности.

Решения системы уравнений (4), (5) исследовались в научной литературе. Так в работах [2, 3], было показано, что равновесия спутника определяются действительными корнями алгебраического уравнения шестой степени. Алгебраическое уравнение, определяющее равновесия спутника было получено с использованием понятия результанта. В явном и довольно сложном виде эти уравнения, определяющие положения равновесия, были получены в работах [4, 5].

Основное внимание в данной работе уделено символьным методам исследования положений равновесия в случаях, когда вектор постоянного момента находится в одной из плоскостей, образуемых главными центральными осями инерции спутника: 1) $a = 0$, $b \neq 0$, $c \neq 0$, 2) $a \neq 0$, $b = 0$, $c \neq 0$, 3) $a \neq 0$, $b \neq 0$, $c = 0$.

4. ИССЛЕДОВАНИЕ ПОЛОЖЕНИЙ РАВНОВЕСИЯ СПУТНИКА С ПРИМЕНЕНИЕМ АЛГОРИТМОВ ПОСТРОЕНИЯ БАЗИСОВ ГРЕБНЕРА

4.1. Положения равновесия для случая a = 0

Начнем с рассмотрения первого случая $a = 0$, $b \neq 0$, $c \neq 0$, когда вектор постоянного момента находится в плоскости Oyz. Для нахождения решения алгебраической системы (4), (5) применялись алгоритмы построения базисов Гребнера [6, 7]. Метод построения базиса Гребнера представляет собой алгоритмическую процедуру для полного приведения задачи в случае системы полиномов от многих переменных к рассмотрению полинома от одной переменной. Алгоритмы построения базиса Гребнера реализованы в большинстве современных систем компьютерной алгебры. Исследование проводилось с использованием реализованного в системе компьютерной алгебры Mathematica 12.1 [13, 14] пакета построения базисов Гребнера `GroebnerBasis[]`. Построим базис Гребнера для 6 полиномов f_i, которые представляют собой левые части уравнений системы (4), (5) с 6 направляющими косинусами a_{ij} $(i = 2,3,\ j = 1,2,3)$, используя лексикографическое упорядочение по переменным направляющим косинусам:

```
GroebnerBasis[{f1,f2,f3,f4,f5,f6},{a21,a22,a23,a31,a32,a33}].
```

Следует отметить, что при изменении лексикографического порядка в списке переменных в базисе Гребнера полином от одной неизвестной по переменной a_{23} имеет более простой вид, чем по переменной a_{33}. Выпишем из построенного базиса Гребнера, состоящего из 36 полиномов, полином, который зависит только от одной переменной $x = a_{33}$

$$\begin{aligned} P(x) = p_0x^6 + p_1x^5 + p_2x^4 + p_3x^3 + \\ + p_4x^2 + p_5x + p_6 = 0, \end{aligned} \tag{6}$$

где

$$\begin{aligned} p_0 &= 9^3 4096, \\ p_1 &= -9^3 8192 \\ p_2 &= 9^2 256(25b^2 + 144), \\ p_3 &= 3456b^2(b^2c^2 - 150), \\ p_4 &= 144b^4(25b^2 + 144), \\ p_5 &= -288b^6c^2, \end{aligned}$$

$$p_6 = b^8 c^4.$$

При построении базиса Гребнера, используя другой лексикографический порядок в списке переменных

GroebnerBasis[{f1,f2,f3,f4,f5,f6},{a31,a32,a33,a21, a22,a23}] получим базис, состоящий из 28 полиномов. Выпишем из построенного базиса Гребнера полином, который зависит только от одной переменной $y = a_{23}$

$$P_1(y) = p_0 y^6 + p_1 y^5 + p_2 y^4 + p_3 y^3 + p_4 y^2 + p_5 y + p_6 = 0, \quad (7)$$

где

$$p_0 = 4096,$$
$$p_1 = -8192$$
$$p_2 = 256(17b^2 + 16),$$
$$p_3 = 128b^2(b^2c^2 - 34),$$
$$p_4 = 16b^4(17b^2 + 16),$$
$$p_5 = -32b^6c^2,$$
$$p_6 = b^8c^4.$$

Полином (7) имеет более простой вид по сравнению с (6) и именно его будем использовать для исследования равновесных решений системы (4), (5). В работе [4] было показано, что каждому действительному корню уравнения (6) или (7) соответствуют четыре равновесных решения исходной системы (4), (5). Отсюда следует, что спутник на круговой орбите под действием гравитационного и постоянного момента в случае $a = 0$, $b \neq 0$, $c \neq 0$, может иметь не более 24 положений равновесия.

При исследовании положений равновесия спутника ставится задача определения на плоскости параметров b, c областей с одинаковым числом вещественных корней уравнения (7). Для выделения в пространстве параметров областей с одинаковым числом вещественных корней построим дискриминантную гиперповерхность полинома (7). Эта гиперповерхность содержит компоненту коразмерности 1, которая является границей областей с одинаковым числом вещественных корней. Множество особых точек дискриминантной кривой на плоскости параметров b, c задается следующей системой алгебраических уравнений:

$$P_1(y) = 0, \quad P_1'(y) = 0. \quad (8)$$

Символом “штрих” обозначено дифференцирование по переменной y. Исключим переменную y из системы уравнений (8) путем вычисления определителя матрицы результанта этих уравнений с помощью символьных матричных функций системы Wolfram Mathematica и получим в результате алгебраическое уравнение дискриминантной гиперповерхности в виде

$$P_2(b,c) = p_{00}b^{10} + p_{01}b^8 + p_{02}b^6 + p_{03}b^4 + p_{04}b^2 + p_{05} = 0, \quad (9)$$

где

$$p_{00} = 729c^8,$$
$$p_{01} = c^4(27376 - 18252c^2 - 3159c^4 + 729c^6)$$
$$p_{02} = 20736 - 97344c^2 - 118976c^4 + 132619c^6 - 18252c^8,$$
$$p_{03} = -134784 + 648288c^2 - 170183c^4 - 118976c^6 + 27376c^8,$$
$$p_{04} = -36(-6084 + 29029c^2 - 18008c^4 + 2704c^6),$$
$$p_{05} = 1296(c^2 - 4)(4c^2 - 1)(4c^2 - 9).$$

Теперь мы должны проверить как изменяется число равновесий при пересечении кривой (9). Это можно сделать численно, определив число действительных решений в одной точке из каждой области, ограниченной кривой (9) на плоскости (b,c). Другим способом определить число действительных корней полинома можно путем вычисления i-х субдискриминантов, используя работы [20, 21].

Области в пространстве параметров, где существуют положения равновесия, как было показано в работе [2], ограничены следующими неравенствами:

$$a^2 + b^2 + c^2 \le 5, \quad a^2 + b^2 \le 4, \quad c^2 + b^2 \le 4, \quad a^2 + c^2 \le 4. \quad (10)$$

На рис. 1 показаны области с одинаковым числом действительных решений уравнения (7), равным 6, 4 и 2, где существуют 24, 16 и 8 равновесных ориентаций спутника соответственно. В работе [5] было показано, что каждому корню уравнения (7) соответствуют 4 равновесные ориентации. Границы этих областей задаются уравнением $P_2(b,c) = 0$. Области изменения параметров b, c ограничены в данном случае квадратом $2 \le b \le 2$; $2 \le c \le 2$. Кривые границ областей на рис. 1 построены с использованием функции *ContourPlot* в системе Wolfram Mathematica.

4.2. Положения равновесия для случая $b = 0$

Рассмотрим теперь следующий случай, когда вектор постоянного момента лежит в плоскости Oxz и $b = 0$, $a \neq 0$, $c \neq 0$. Построим базис Гребнера для полиномов, представляющих собой левые

части системы (4), (5) для случая $b = 0$. Выполнение алгоритма с лексикографическим упорядочением по переменным позволяет получить в построенном базисе полином, который зависит только от одной переменной $y = a_{23}$

$$P_3(y) = p_0y^6 + p_1y^5 + p_2y^4 + p_3y^3 + p_4y^2 + p_5y + p_6 = 0, \quad (11)$$

где

$$p_0 = 4096,$$
$$p_1 = -8192$$
$$p_2 = 256(17a^2 + 16),$$
$$p_3 = 128a^2(a^2c^2 - 34),$$
$$p_4 = 16a^4(17c^2 + 16),$$
$$p_5 = -32a^6c^2,$$
$$p_6 = a^8c^4.$$

Полином (11) с точностью до замены параметра a на b совпадает с (7). Поэтому и уравнение дискриминантной гиперповерхности для полинома (11) совпадает с точностью до обозначений с уравнением (9) и области с одинаковым числом действительных решений уравнения (11) имеют такой же вид как показано на рис. 1.

4.3. Положения равновесия для случая $c = 0$

В последнем случае, когда вектор постоянного момента лежит в плоскости Oxy и $c = 0$ ($a \neq 0$, $b \neq 0$), построим базис Гребнера для полиномов, представляющих собой левые части системы (4), (5). Вычисляя базис Гребнера с опцией лексикографического упорядочения по переменным для системы полиномов, представляющих собой левые части уравнений (4), (5) для случая $c = 0$, получим в результате полином, зависящий только от одной переменной a_{23}, который имеет вид

$$P_4(y) = p_0y^6 + p_1y^5 + p_2y^4 + p_3y^3 + p_4y^2 + p_5y + p_6 = 0, \quad (12)$$

где

$$p_0 = 4096,$$
$$p_1 = -8192$$
$$p_2 = 256(17(a^2 + b^2) - 6a^2b^2 + 16),$$
$$p_3 = -128(a^2b^2(a^2 + b^2) - 8a^2b^2 + 34(a^2 + b^2)),$$
$$p_4 = 16(16(a^4 + b^4) + 9a^4b^4 - 43a^2b^2(a^2 + b^2) + 289a^2b^2),$$

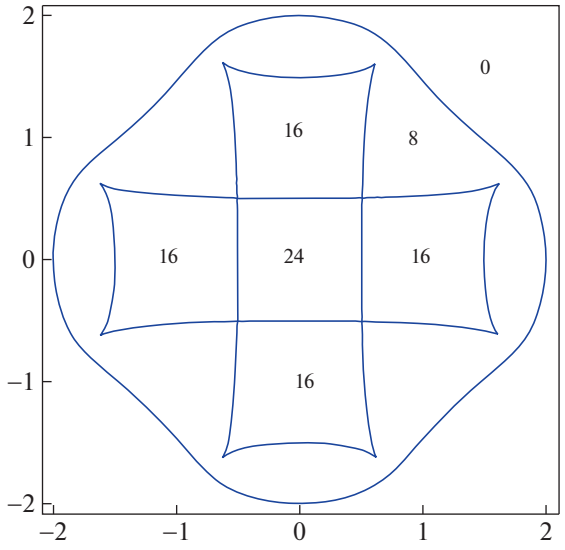


Рис. 1. Области с равным числом положений равновесия.

$$p_5 = 8a^2b^2(4(a^4 + b^4) + 3a^2b^2(a^2 + b^2) - 30a^2b^2 - 34(a^2 + b^2)),$$
$$p_6 = a^4b^4(a^2 + b^2 + 4)^2.$$

Алгебраическое уравнение дискриминантной гиперповерхности уравнения (12) имеет вид

$$P_5(b,c) = p_{50}a^{10} + p_{51}a^8 + p_{52}b^6 + p_{53}b^4 + p_{54}b^2 + p_{55} = 0, \quad (13)$$

где

$$p_{50} = 729b^8,$$
$$p_{51} = b^4(729b^6 - 3159b^4 - 18252b^2 + 27376)$$
$$p_{52} = 20736 - 97344b^2 - 118976b^4 + 132619b^6 - 18252b^8,$$
$$p_{53} = 27376b^8 - 118976b^6 - 170183b^4 + 648288b^2 - 134784,$$
$$p_{54} = -36(2704b^6 - 18008b^4 + 29029b^2 - 6084),$$
$$p_{55} = 1296(b^2(13 - 4b^2)^2 - 36).$$

График уравнения (13) на плоскости a, b, где указаны границы областей с одинаковым числом действительных решений, имеет такой же вид, как на рис. 1.

5. ЗАКЛЮЧЕНИЕ

В работе проведено исследование положений равновесия спутника, движущегося по круговой

орбите при влиянии постоянного момента с применением методов построения базиса Гребнера. Метод построения базиса Гребнера является универсальным методом, применяемым для решения систем алгебраических уравнений. Существенным ограничением для применения данного метода является экспоненциальная сложность алгоритмов вычисления базиса Гребнера. В нашем случае время построения базиса Гребнера в системе Wolfram Mathematica 12.1 на персональном компьютере с 8 Гигабайт оперативной памяти и процессором Intel Core i7 2.8 ГГц составило всего около 0.1 сек. Анализ полиномов, входящих в базис Гребнера, позволяет провести детальное исследование влияния постоянного момента на положения равновесия спутника.

На основании полученных в данной работе с использованием методов компьютерной алгебры результатов можно сделать вывод, что число положений равновесия спутника в случаях, когда вектор постоянного момента параллелен плоскости, образуемой любыми двумя главными центральными осями инерции спутника, увеличиваются от нуля до 8, далее до 16 и далее до 24 с шагом 8 при последовательном уменьшении величины вектора постоянного момента. Применение методов компьютерной алгебры позволяет в достаточно простой форме получить решение задачи механики космического полета.

СПИСОК ЛИТЕРАТУРЫ

1. *Garber T.B.* Influence of Constant Disturbing Torques on the Motion of Gravity Gradient Stabilized Satellites. AIAA J. 1963. V. 1. № 4. P. 968–969.
2. *Сарычев В.А., Гутник С.А.* Равновесия спутника под действием гравитационного и постоянного моментов. Космич. исслед. 1994. Т. 32. № 4–5. С. 43–50.
3. *Sarychev V.A., Paglione P., Guerman A.* Influence of Constant Torque on Equilibria of a Satellite in a Circular Orbit. Celestial Mechanics and Dynamical Astronomy. 2003. V. 87. P. 219–239.
4. *Герман А.Д., Гутник С.А., Сарычев В.А.* Динамика спутника под действием гравитационного и постоянного моментов и их устойчивость. Изв. РАН. ТИСУ. 2016. № 3. С. 142–155.
5. *Gutnik S.A., Guerman A., Sarychev V.A.* Application of Computer Algebra Methods to Investigation of Influence of Constant Torque on Stationary Motions of Satellite. In: Gerdt V.P., Koepf W., Seiler W.M., Vorozhtsov, E.V. (eds.) CASC 2015. Lecture Notes in Computer Science (LNCS). Springer Verlag. 2015. V. 9301. P. 198–209.
6. *Buchberger B.* Theoretical basis for the reduction of polynomials to canonical forms, SIGSAM Bull. 1976. V. 10. № 3. P. 19–29.
7. *Бухбергер Б.* Базисы Грёбнера. Алгоритмический метод в теории полиномиальных идеалов. Компьютерная алгебра. Символьные и алгебраические вычисления. М.: Мир, 1986. С. 331–372.
8. *Гутник С.А., Сарычев В.А.* Символьно-численные методы исследования положений равновесия спутника-гиростата. Программирование. 2014. № 3. С. 49–58.
9. *Гутник С.А., Сарычев В.А.* Применение методов компьютерной алгебры для исследования стационарных движений спутника-гиростата. Программирование. 2017. № 2. С. 35–44.
10. *Gutnik S.A., Sarychev V.A.* Symbolic-numeric Simulation of Satellite Dynamics with Aerodynamic Attitude Control System. Lect. Notes Comput. Sci., Springer, Cham. 2018. V. 11077. P. 214–229.
11. *Гутник С.А., Сарычев В.А.* Применение методов компьютерной алгебры для исследования динамики системы двух связанных тел на круговой орбите. Программирование. 2019. № 2. С. 32–40.
12. *Гутник С.А., Сарычев В.А.* Символьные методы вычисления положений равновесия системы двух связанных тел на круговой орбите. Программирование. 2022. № 2. С. 16–22.
13. http://www.wolfram.com/mathematica
14. *Hastings C., Mischo K., Morrison M.* Hands-on Start to Wolfram Mathematica and Programming with the Wolfram Language. 3-d Edition, Wolfram Media, Ink. Champaign. 2020.
15. *Прокопеня А.Н., Минглибаев М.Дж., Маемерова Г.М.* Символьные вычисления в исследованиях проблемы трех тел с переменными массами. Программирование. 2014. № 2. С. 51–59.
16. *Прокопеня А.Н., Минглибаев М.Дж., Маемерова Г.М., Иманова Ж.У.* Исследование ограниченной задачи трех тел с переменными массами методами компьютерной алгебры. Программирование. 2017. № 5. С. 18–23.
17. *Прокопеня А.Н., Минглибаев М.Дж., Шомшекова С.А.* Применение компьютерной алгебры в исследованиях двухпланетной задачи трех тел с переменными массами. Программирование. 2019. № 2. С. 58–65.
18. *Будько Д.А., Прокопеня А.Н.* Символьно-численные методы поиска положений равновесия в ограниченной задаче четырех тел. Программирование. 2013. № 2. С. 30–37.
19. *Сарычев В.А.* Вопросы ориентации искусственных спутников. Итоги науки и техники. Сер. “Исследование космического пространства”. Т. 11. М.: ВИНИТИ, 1978.
20. *Батхин А.Б.* Параметризация дискриминантного множества многочлена. Программирование. 2016. № 2. С. 8–21.
21. *Батхин А.Б.* Параметризация множества, определяемого обобщенным дискриминантом многочлена. Программирование. 2018. № 2. С. 5–17.

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 33–38

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.422.8

СРЕДСТВА КОМПЬЮТЕРНОЙ АЛГЕБРЫ ДЛЯ ГЕОМЕТРИЗАЦИИ УРАВНЕНИЙ МАКСВЕЛЛА

© 2023 г. А. В. Королькова[a],*, М. Н. Геворкян[a],, Д. С. Кулябов[a,b],***, Л. А. Севастьянов[a,b],******

[a] *Российский университет дружбы народов, 117198 ул. Миклухо-Маклая, д. 6, Москва, Россия*

[b] *Объединенный институт ядерных исследований, 141980 ул. Жолио-Кюри 6, Дубна, Московская область, Россия*

**E-mail: korolkova-av@rudn.ru*

***E-mail: gevorkyan-mn@rudn.ru*

****E-mail: kulyabov-ds@rudn.ru*

*****E-mail: sevastianov-la@rudn.ru*

Поступила в редакцию 05.08.2022 г.
После доработки 04.09.2022 г.
Принята к публикации 21.10.2022 г.

При расчете оптических приборов в рамках геометризованной теории Максвелла используются широко известные формализмы общей теории относительности и дифференциальной геометрии. В частности, для подобных вычислений требуется знать аналитический вид уравнений геодезических. Что приводит к необходимости вычислять большое количество однообразных математических выражений. Одним из предназначений средств компьютерной алгебры является облегчение работы исследователя путем автоматизации громоздких символьных расчетов. Таким образом, использование систем компьютерной алгебры представляется вполне очевидным действием. В работе рассмотрено несколько свободных реализаций символьных вычислений для аппарата общей теории относительности. В конце статьи приводится практический пример символьных расчетов для геометризованной теории Максвелла.

DOI: 10.31857/S0132347423020127, **EDN:** MGAIHN

1. ВВЕДЕНИЕ

Геометрический подход к уравнениям Максвелла прошел несколько этапов в своем развитии. Первоначальный интерес был вызван общей теорией относительности. Это работы Л.И. Мандельштама, И.Е. Тамма [1–3], В. Гордона [4]. В отсутствии практических приложений интерес к данной теме угас. Новая вспышка интереса возникла во время золотого века теории относительности (1960–1975 гг.). Это работы Е. Плебаньского [5], Ф. де Феличе [6]. Впрочем, следует заметить, что и в этот период не удалось определиться с приложениями данной теории.

Новая вспышка интереса к геометрическому подходу проявилась в середине 2000-х годов на фоне интереса к метаматериалам [7]. Это работы Д.Б. Пендри [8, 9] и У. Леонгардта [10, 11]. Эти работы породили целое направление – трансформационную оптику [12].

Геометризация материальных уравнений Максвелла позволяет изменить взгляд на прямую и обратную задачу оптики. Нахождение траектории лучей по параметрам среды можно назвать прямой задачей оптики, а нахождение параметров среды по заданным траекториям лучей – обратной. Естественно, что обратная задача сложнее прямой. В геометризованной максвелловской оптике эти задачи меняются местами. Прямая задача – нахождение диэлектрической и магнитной проницаемости по заданной эффективной геометрии (по траекториям лучей), обратная – нахождение эффективной геометрии по диэлектрической и магнитной проницаемости. Сложность обеих задач сопоставимая.

Геометризация заключается в построении индуцированной (электромагнитной) метрики. При этом индуцированная метрика должна быть связанна с метрикой лабораторного пространства. В прямой задаче геометризованной оптики (соответствующей обратной задаче при обычном подходе) задается траектория в лабораторной системе координат, далее по этим траекториям задаются геодезические в индуцированной (электромагнитной) системе координат. По уравнениям геодезических строится метрика в индуцированной системе координат. Такого типа задачи характе-

ризуются большим количеством несложных операций. Ручные вычисления не только выматывающие, но и подвержены накапливающимся ошибкам. Поэтому естественно выглядит желание переложить эти вычисления на плечи системы компьютерной алгебры.

1.1. Структура статьи

В разделе 3 описываются основные положения методики геометризации уравнений Максвелла. Выбор конкретной библиотеки для символьных преобразований описано в разделе 3. Процесс применения средств компьютерной алгебры для исследования в рамках геометризованной теории Максвелла представлен в разделе 4.

2. МЕТОДИКА ГЕОМЕТРИЗАЦИИ УРАВНЕНИЙ МАКСВЕЛЛА

Под геометризацией уравнений Максвелла мы понимаем эффективную геометризацию. То есть мы устанавливаем соответствие между уравнением Максвелла в среде (в плоском пространстве Минковского) и вакуумным уравнением Максвелла в эффективном римановом пространстве. Мы не считаем, что электромагнитное поле искривляет пространство, но устанавливаем соответствие между средой и метрикой [13–15].

Это позволяет рассматривать траектории лучей как геодезические в римановом пространстве. Задавая же вначале пути распространения лучей и восстанавливая по геодезическим метрику, мы можем решать обратную задачу оптики.

Входными данными для задачи перевода материальных уравнений Максвелла в вакуумные уравнения Максвелла в искривленном пространстве являются параметры среды: диэлектрическая ε_i^j и магнитная μ_i^j проницаемости, показатель преломления среды n_{ij}. В связи со спецификой геометризации на основе квадратичной метрики [1, 2] реально используется только показатель преломления.

Вид эффективного метрического тензора $g_{\alpha\beta}$ подбирается из соображений симметрии задачи [2, 3]. Сам метрический тензор зависит от параметров среды.

Имея эффективный метрический тензор, можно решить геометризованные уравнения Максвелла и получить искомые траектории распространения электромагнитных волн. В рамках статьи упростим задачу и будем использовать приближение геометрической оптики. В этом случае лучи будут распространяться по геодезическим:

$$\frac{\mathrm{d}^2 x^\alpha}{\mathrm{d}t} + \Gamma^\alpha_{\beta\gamma}\frac{\mathrm{d}x^\beta}{\mathrm{d}t}\frac{\mathrm{d}x^\gamma}{\mathrm{d}t} = 0,$$

где $x^\alpha(t)$ – координаты геодезической, а $\Gamma^\alpha_{\beta\gamma}$ – символы Кристоффеля второго рода:

$$\Gamma^\alpha_{\beta\gamma} = \frac{1}{2}g^{\alpha\delta}\left(\frac{\partial g_{\delta\beta}}{\partial x^\gamma} + \frac{\partial g_{\delta\gamma}}{\partial x^\beta} - \frac{\partial g_{\beta\gamma}}{\partial x^\delta}\right),$$

где $g^{\alpha\delta}$ – обратный метрический тензор, определяемый тождеством $g^{\alpha\delta}g_{\delta\beta} = \delta^\alpha_\beta$, где δ^α_β – дельта-символ Кронекера.

Для случая изотропной среды метрический тензор имеет следующий вид [1, 5]:

$$g_{\alpha\beta} = \begin{pmatrix} \frac{1}{\varepsilon\sqrt{\mu}} & 0 & 0 & 0 \\ 0 & -\sqrt{\mu} & 0 & 0 \\ 0 & 0 & -\sqrt{\mu} & 0 \\ 0 & 0 & 0 & -\sqrt{\mu} \end{pmatrix} \quad (1)$$

Поскольку в методе геометризации на основе квадратичной метрики диэлектрическая и магнитная проницаемость равны [5], то дополнительно запишем:

$$n = \sqrt{\varepsilon\mu} = \varepsilon = \mu. \quad (2)$$

Из уравнения (2) видно, что стандартная геометризация уравнений Максвелла подходит для исследования оптических задач в приближениях геометрической оптики и приближении эйконала.

Таким образом можно отметить, что исследование в рамках геометризованной теории Максвелла подразумевает операции с тензорами в римановом пространстве. Каждое такое действие достаточно простое, но количество таких манипуляций просто огромно. И это может привести исследователя в состояние уныния. И только возможность использования средств компьютерной алгебры сможет примирить его с таким подходом.

3. БИБЛИОТЕКИ СИСТЕМЫ КОМПЬЮТЕРНОЙ АЛГЕБРЫ SymPy ДЛЯ ИССЛЕДОВАНИЯ ИСКРИВЛЕННЫХ ПРОСТРАНСТВ

Нами было рассмотрено несколько вариантов библиотек для работы с формализмом общей теории относительности, который мы применяем для геометризации уравнений Максвелла. Ранее для таких вычислений мы применяли систему компьютерной алгебры Cadabra [16, 17]. Однако в данном исследовании нам было интересно рассмотреть библиотеки для системы компьютерной алгебры SymPy [18]. Заметим, что мы рассматриваем преимущественно свободнораспространяемые программные продукты, и предпочитаем не использовать проприетарные системы компью-

терной алгебры, как многие другие исследователи [19].

Основной сложностью при выводе системы уравнений, задающих геодезическую кривую, является вычисление символов Кристоффеля $\Gamma^{\alpha}_{\beta\gamma}$. Задача несколько облегчается тем, что используется диагональный метрический тензор, однако все равно остается сложной в вычислительном плане. Поэтому разумно вычислить символы Кристоффеля с помощью системы компьютерной алгебры.

Все вычисления будут проводиться с помощью библиотеки символьной алгебры SymPy [20] и оболочки Jupyter Notebook или Jupyter Lab [21].

Мы выделили две библиотеки SymPy: EinsteinPy и gravipy.

3.1. EinsteinPy

Модуль [22] представлялся нам крайне перспективным. Однако при попытке задать метрику с помощью стандартной символьной матрицы SymPy мы столкнулись с ошибкой, которая заключалась в невозможности задать дробь $\frac{1}{\varepsilon\sqrt{\mu}}$. Так, вместо дроби в метрике (1) появляется артефакт, имеющий вид выражения $1\varepsilon\sqrt{\mu}$. Кроме того, библиотека EinsteinPy позиционируется как модуль для численных расчетов и символьные вычисления являются не главной ее задачей.

То есть при всей привлекательности данная библиотека оказалась просто непригодной для символьных вычислений (численные расчеты на базе этой библиотеки мы не рассматривали).

3.2. Gravipy

Так как использовать встроенные средства SymPy оказалось невозможным из-за указанной выше ошибки, авторы попытались найти альтернативу, которая работала бы в рамках SymPy. Такой альтернативой оказался небольшой модуль gravipy [23], в зависимостях которого указана только библиотека SymPy и, опционально, оболочка JupyterLab. Сам модуль фактически состоит из одного файла. Авторы работы использовали версию 0.2.0. Последнее изменение датируется сентябрем 2019 года.

При отсутствии альтернативы мы решили использовать именно эту библиотеку.

Из-за минимальных зависимостей и компактности, модуль можно удобно использовать без установки. Для этого следует в директории с Jupyter-блокнотом, в котором проходят вычисления, создать каталог gravipy и в него поместить файлы `__init__.py` и `tensorial.py` из каталога `gravipy` с официального репозитория [23]. После чего можно пользоваться модулем импортируя его в Jupyter-блокноте обычным способом.

4. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ СИСТЕМЫ КОМПЬЮТЕРНОЙ АЛГЕБРЫ ДЛЯ ИССЛЕДОВАНИЯ УРАВНЕНИЙ МАКСВЕЛЛА

Проведем необходимые вычисления.

Вначале импортируем SymPy и gravipy:

```
import sympy as sp
from gravipy.tensorial import *
```

Настроим отображение формул, включив кодировку юникод, отображение формул в окружении `equation` и их рендеринг с помощью `mathjax`.

```
sp.init_printing(use_unicode=True,
↪ latex_mode='equation' ,
↪ use_latex='mathjax' )
```

Далее определим необходимые символы:

```
# Координаты пространство-времени
t, x, y, z = symbols(' t, x, y, z ' )
# Характеристики среды, как функции
epsilon = sp.Function(' epsilon ' )(x, y,
↪ z)
mu = sp.Function(' mu ' )(x, y, z)
n = sp.Function(' n ' )(x, y, z)
```

Для работы модуля gravipy также необходимо отдельно задать координатный 4-вектор

```
# Создаем координатный 4-вектор
X = Coordinates('X' , [t, x, y, z])
```

Для создания метрики необходимо вначале определить символьную матрицу, а затем преобразовать ее в метрический тензор. Матрицу зададем с помощью функции `diag`, которая импортируется из SymPy автоматически при импорте модуля gravipy

```
# Metric = diag(1 / epsilon /
↪ sp.sqrt(mu), -sp.sqrt(mu),
↪ sp.sqrt(mu), -sp.sqrt(mu))
Metric = diag(1 / n / sp.sqrt(n),
↪ -sp.sqrt(n), -sp.sqrt(n),
↪ -sp.sqrt(n))
```

После задания вспомогательной матрицы можно задать уже метрический тензор, для чего помимо матрицы `Metric` нужно передать еще координатный объект `X`.

```
g = MetricTensor('g' , X, Metric)
```

Для отображения метрики следует выполнить ячейку с кодом `g(All,All)` в результате чего получим следующую матрицу[1]:

$$\begin{pmatrix} \frac{1}{n^{\frac{3}{2}}} & 0 & 0 & 0 \\ 0 & -\sqrt{n} & 0 & 0 \\ 0 & 0 & -\sqrt{n} & 0 \\ 0 & 0 & 0 & -\sqrt{n} \end{pmatrix}$$

После создания метрического тензора, сразу же можно вычислить символы Кристоффеля. Делается это с помощью функции `Christoffel`, которой требуется передать метрический тензор:

```
Gamma = Christoffel('Gamma', g)
```

для распечатки результата также посчитаем в отдельной ячейке выражение `Gamma(All, All, All)`. Получим список из четырех матриц (срезу компонентного представления символа Кристоффеля):

$$\begin{pmatrix} 0 & -\frac{3\frac{\partial n}{\partial x}}{4n^{\frac{5}{2}}} & -\frac{3\frac{\partial n}{\partial y}}{4n^{\frac{5}{2}}} & -\frac{3\frac{\partial n}{\partial z}}{4n^{\frac{5}{2}}} \\ -\frac{3\frac{\partial n}{\partial x}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \\ -\frac{3\frac{\partial n}{\partial y}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \\ -\frac{3\frac{\partial n}{\partial z}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} \frac{3\frac{\partial n}{\partial x}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \\ 0 & -\frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} \\ 0 & -\frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} & \frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} & 0 \\ 0 & -\frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} & 0 & \frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} \end{pmatrix},$$

$$\begin{pmatrix} \frac{3\frac{\partial n}{\partial y}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \\ 0 & \frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} & 0 \\ 0 & -\frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} \\ 0 & 0 & -\frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} & \frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} \end{pmatrix},$$

$$\begin{pmatrix} \frac{3\frac{\partial n}{\partial z}}{4n^{\frac{5}{2}}} & 0 & 0 & 0 \\ 0 & \frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} & 0 & -\frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} \\ 0 & 0 & \frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial y}}{4\sqrt{n}} \\ 0 & -\frac{\frac{\partial n}{\partial x}}{4\sqrt{n}} & -\frac{\frac{\partial}{\partial y}n}{4\sqrt{n}} & -\frac{\frac{\partial n}{\partial z}}{4\sqrt{n}} \end{pmatrix}.$$

Функция `Geodesic` позволяет вычислить уравнения геодезической линии на основе метрики. Символы Кристоффеля вычисляются ею автоматически и отдельно передавать их в качестве аргументов не требуется.

```
w = Geodesic('w', g, t)
```

Результат выполнения функции выглядит крайне громоздко. Нет необходимости приводить здесь этот вывод.

При вычислениях функция `Geodesic` автоматически использует параметризацию по параметру t, однако не отождествляет этот параметр с координатой времени и поэтому не производит замены $\frac{dt}{dt} = 1$ и $\frac{d^2t}{dt^2} = 0$. Параметризацию можно отключить вызовом следующей функции:

```
Parametrization.deactivate(t)
# выводит сообщение о том включена ли
↪ автоматизация или нет
Parametrization.info()
```

Однако на работе функции `Geodesic` эта настройка не сказывается, из-за чего требуется произвести некоторые дополнительные преобразования уже вручную.

[1] Здесь $n := n(x, y, z)$

Прежде всего подставим $\frac{dt}{dt} = 1$ и $\frac{d^2t}{dt} = 0$ во все уравнения. Первое уравнение принимает наиболее простой вид:

$$-\frac{3}{2}\left(\frac{\partial n}{\partial x}\frac{dx}{dt} + \frac{\partial n}{\partial y}\frac{dy}{dt} + \frac{\partial n}{\partial z}\frac{dz}{dt}\right) = 0, \quad (3)$$

а оставшиеся три уравнения будут иметь следующий вид:

$$-\frac{1}{2}\left(\frac{\partial n}{\partial x}\frac{dx}{dt} + \frac{\partial n}{\partial y}\frac{dy}{dt} + \frac{\partial n}{\partial z}\frac{dz}{dt}\right)\frac{dx}{dt} - n\frac{d^2x}{dt^2} + \frac{1}{4}\frac{\partial n}{\partial x}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2\right] + \frac{3}{4}\frac{\partial n}{\partial x}\frac{1}{n^2} = 0,$$

$$-\frac{1}{2}\left(\frac{\partial n}{\partial x}\frac{dx}{dt} + \frac{\partial n}{\partial y}\frac{dy}{dt} + \frac{\partial n}{\partial z}\frac{dz}{dt}\right)\frac{dy}{dt} - n\frac{d^2y}{dt^2} + \frac{1}{4}\frac{\partial n}{\partial y}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2\right] + \frac{3}{4}\frac{\partial n}{\partial y}\frac{1}{n^2} = 0,$$

$$-\frac{1}{2}\left(\frac{\partial n}{\partial x}\frac{dx}{dt} + \frac{\partial n}{\partial y}\frac{dy}{dt} + \frac{\partial n}{\partial z}\frac{dz}{dt}\right)\frac{dz}{dt} - n\frac{d^2z}{dt^2} + \frac{1}{4}\frac{\partial n}{\partial z}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2\right] + \frac{3}{4}\frac{\partial n}{\partial z}\frac{1}{n^2} = 0.$$

В каждом из этих трех уравнений в качестве левой части выступает выражение (3), что дает возможность заменить его на ноль. В результате получаем систему из трех уравнений следующего вида:

$$\begin{aligned}
\frac{1}{4}\frac{\partial n}{\partial x}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2 + \frac{3}{n^2}\right] &= n\frac{d^2y}{dt^2},\\
\frac{1}{4}\frac{\partial n}{\partial y}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2 + \frac{3}{n^2}\right] &= n\frac{d^2z}{dt^2}, \quad (4)\\
\frac{1}{4}\frac{\partial n}{\partial z}\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2 + \frac{3}{n^2}\right] &= n\frac{d^2x}{dt^2}.
\end{aligned}$$

В свою очередь данную систему уравнений второго порядка можно преобразовать в систему ОДУ первого порядка из шести уравнений с помощью следующей замены переменных:

$$p_x = \frac{dx}{dt}, \quad p_y = \frac{dy}{dt}, \quad p_z = \frac{dz}{dt}.$$

Тогда уравнения (4) приводятся к виду:

$$\begin{cases}
n\dfrac{dp_x}{dt} = \dfrac{1}{4}\dfrac{\partial n}{\partial x}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right],\\
n\dfrac{dp_y}{dt} = \dfrac{1}{4}\dfrac{\partial n}{\partial y}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right],\\
n\dfrac{dp_z}{dt} = \dfrac{1}{4}\dfrac{\partial n}{\partial z}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right].
\end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases}
\dfrac{dp_x}{dt} = \dfrac{1}{4n}\dfrac{\partial n}{\partial x}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right],\\
\dfrac{dp_y}{dt} = \dfrac{1}{4n}\dfrac{\partial n}{\partial y}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right],\\
\dfrac{dp_z}{dt} = \dfrac{1}{4n}\dfrac{\partial n}{\partial z}\left[p_x^2 + p_y^2 + p_z^2 + \dfrac{3}{n^2}\right].
\end{cases}$$

Следует отметить, что аналогичная замена получается при решении уравнения эйконала [24] методом характеристик [25–28].

5. ЗАКЛЮЧЕНИЕ

В статье приведён алгоритм расчёта характеристик оптических приборов в рамках геометризованной теории Максвелла. Одним из основных этапов расчётов является получение уравнения геодезической и преобразование его в систему обыкновенных дифференциальных уравнений, пригодную для дальнейших вычислений. Поскольку данная задача требует от исследователя крайне кропотливых вычислений, то для её решения используется система компьютерной алгебры. В статье даётся сравнительный обзор нескольких систем компьютерной алгебры, предназначенных для решения задач общей теории относительности. На примере одного такого пакета демонстрируются основные операции при практическом исследовании в рамках геометризованной теории Максвелла.

БЛАГОДАРНОСТИ

Публикация выполнена при поддержке Программы стратегического академического лидерства РУДН.

СПИСОК ЛИТЕРАТУРЫ

1. *Тамм И.Е.* Электродинамика анизотропной среды в специальной теории относительности // Журнал Русского физико-химического общества. Часть физическая. 1924. Т. 56. № 2–3. С. 248–262.
2. *Тамм И.Е.* Кристаллооптика теории относительности в связи с геометрией биквадратичной формы // Журнал Русского физикохимического общества. Часть физическая. 1925. Т. 57. № 3–4. С. 209–240.
3. Mandelstam L.I., Tamm I.Y. Elektrodynamik der anisotropen medien in der speziellen relativittstheorie // Mathematische Annalen. 1925. Bd. 95. H. 1. S. 154–160.
4. *Gordon W.* Zur Lichtfortpflanzung nach der Relativita..tstheorie // Annalen der Physik. 1923. Bd. 72. S. 421–456.
5. *Plebanski J.* Electromagnetic waves in gravitational fields // Physical Review. 1960. V. 118. № 5. P. 1396–1408.

6. *Felice F.* On the Gravitational Field Acting as an Optical Medium // General Relativity and Gravitation. 1971. V. 2. № 4. P. 347–357.
7. *Smolyaninov I.I.* Metamaterial 'Multiverse' // Journal of Optics. 2011. V. 13. № 2. P. 024004.
8. *Pendry J.B., Schurig D., Smith D.R.* Controlling Electromagnetic Fields // Science. 2006. V. 312. № 5781. P. 1780–1782.
9. *Schurig D., Pendry J.B., Smith D.R.* Calculation of Material Properties and Ray Tracing in Transformation Media // Optics express. 2006. V. 14. № 21. P. 9794–9804.
10. *Leonhardt U.* Optical Conformal Mapping // Science. 2006. V. 312. № June. P. 1777–1780.
11. *Leonhardt U., Philbin T.G.* Transformation Optics and the Geometry of Light // Progress in Optics. 2009. V. 53. P. 69–152.
12. *Foster R., Grant P., Hao Y. et al.* Spatial Transformations: from Tundamentals to Applications // Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2015. 8. V. 373. № 2049. P. 20140365.
13. *Kulyabov D.S., Korolkova A.V., Sevastianov L.A.* A naive geometrization of maxwell's equations // The 15th small triangle meeting of Theoretical Physics. Star Lesn, 2013. P. 104–111.
14. *Кулябов Д.С., Королькова А.В., Севастьянов Л.А.* Простейшая геометризация уравнений Максвелла // Вестник РУДН. Серия. Математика. Информатика. Физика. 2014. № 2. С. 115–125.
15. *Kulyabov D.S., Korolkova A.V., Sevastianov L.A. et al.* Algorithm for lens calculations in the geometrized maxwell theory // Saratov Fall Meeting 2017. V. 10717 of Proceedings of SPIE. Saratov : SPIE, 2018. 4. P. 107170Y.1–6.
16. *Королькова А.В., Кулябов Д.С., Севастьянов Л.А.* Тензорные расчеты в системах компьютерной алгебры // Программирование. 2013. № 3. С. 47–57.
17. *Кулябов Д.С., Королькова А.В., Севастьянов Л.А.* Новые возможности второй версии пакета компьютерной алгебры cadabra // Программирование. 2019. № 2. С. 41–48.
18. Sandon D. Symbolic Computation with Python and SymPy. Independently published, 2021. ISBN: 979-8489815208.
19. *Диваков Д.В., Тютюнник А.А.* Символьное исследование спектральных характеристик направляемых мод плавно-нерегулярных волноводов // Программирование. 2022. № 2. С. 23–32.
20. Sympy. 2022. URL: http://www.sympy.org/ru/index.html.
21. Project jupyter. 2022. URL: https://jupyter.org/.
22. Einsteinpy–making einstein possible in python. 2022. URL:https://einsteinpy-einsteinpy.readthedocs.io/en/latest/index.html.
23. Gravipy tensor calculus package for general relativity based on sympy. 2022. URL: https://github.com/wojciechczaja/GraviPy.
24. *Bruns H.* Das Eikonal. Leipzig: S. Hirzel, 1895. Bd. 35.
25. *Borovskikh A.V.* The two-dimensional eikonal equation // Siberian Math. J. 2006. V. 47. P. 813–834.
26. *Moskalensky E.D.* Finding exact solutions to the two-dimensional eikonal equation // Num. Anal. Appl. 2009. V. 2. P. 201–209.
27. *Kabanikhin S.I., Krivorotko O.I.* Numerical solution eikonal equation // Sib. Elektron. Mat. Izv. 2013. V. 10. P. 28–34.
28. *Kulyabov D.S., Korolkova A.V., Velieva T.R., Gevorkyan M.N.* Numerical analysis of eikonal equation // Saratov Fall Meeting 2018. Vol. 11066 of Proceedings of SPIE. Saratov: SPIE, 2019. 6. P. 56.

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 39–57

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.43

ВАРИАНТ РЕАЛИЗАЦИИ ПРОЦЕДУРЫ АНАЛИЗА ИНФОРМАЦИОННЫХ ПОТОКОВ В ПРОГРАММНЫХ БЛОКАХ PL/SQL С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ PLIF

© 2023 г. А. А. Тимаков[a],* (ORCID: 0000-0003-4306-789X)

[a]МИРЭА – Российский технологический университет
119454 Москва, Проспект Вернадского, д. 78, Россия

*****E-mail: timakov@mirea.ru*

Поступила в редакцию 15.07.2022 г.
После доработки 15.12.2022 г.
Принята к публикации 13.01.2023 г.

Формальное доказательство эффективности реализуемых мер защиты и безопасности вычислений (обработки информации) является важнейшим условием доверия к критическим информационным системам. Важно понимать, что при построении таких систем формальная проверка безопасности должна применяться на всех инфраструктурных уровнях (от физического до прикладного). В настоящее время на практике проблемой остается формальная проверка безопасности вычислений на прикладном уровне, требующая сложной разметки элементов среды вычислений. Для решения данной задачи традиционно используются методы, в основе которых лежит контроль информационных потоков (КИП). В отличие от методов на основе управления доступом, нашедших широкое практическое применение в операционных системах (ОС) и системах управления базами данных (СУБД), КИП в программном обеспечении на практике применяется весьма ограниченно и в основном сводится к анализу помеченных данных – Taint Tracking. В работе приводится вариант реализации КИП в программных блоках PL/SQL с использованием платформы PLIF. Кроме того, описана общая схема проверки безопасности вычислений в приложениях уровня предприятия, работающих с реляционными базами данных. Преимуществом подхода можно считать явное разделение функций разработчиков программного обеспечения и аналитиков безопасности.

DOI: 10.31857/S0132347423040118, **EDN:** RDRCOC

1. ВВЕДЕНИЕ

Требования к компьютерным системам заданного класса защиты определяются соответствующими оценочными стандартами. Первым подобным стандартом стали “Критерии определения безопасности компьютерных систем” Министерства обороны США [1]. В современном мире требования по безопасности формулируются в терминах “Общих критериев оценки защищенности информационных технологий” [2] и делятся на функциональные требования и требования доверия. Анализ показывает, что существует три важных категории условий, учитываемых при определении класса защиты компьютерной системы, в общем виде их можно сформулировать следующим образом:

1. Формальное доказательство эффективности реализованных механизмов защиты (модулей, реализующих функции защиты) или безопасности вычислений (обработки информации).

2. Контроль скрытых каналов.

3. Контроль “легальных” траекторий и сред распространения информации (управление доступом в случае ОС).

Далее для наглядности ограничимся грубым разбиением абстрактной компьютерной системы на три уровня: физический, системный и прикладной. Будем полагать, что физический уровень включает оконечные рабочие станции – пользовательские и серверные, а также сетевое оборудование.

Сложность выполнения обозначенных выше условий на практике возрастает от физического к прикладному уровню.

На физическом уровне “легальные” траектории распространения информации включают пользователей компьютерной системы, получающих доступ в выделенные помещения и взаимодействующих со средствами вычислительной техники посредством стандартных интерфейсов ввода-вывода. Контроль такого взаимодействия на практике успешно осуществляется путем реализации определенных организационно-техниче-

ских мер: ограничением лиц, допущенных в выделенные помещения и к работе с установленными там средствами вычислительной техники, внедрением систем контроля и управления доступом (СКУД), установкой систем видеонаблюдения и сигнализации, выполнением требований к блокировке входных и оконных проемов и т.д. Контроль скрытых акустических, электромагнитных и оптических каналов на физическом уровне (также называемых техническими каналами) реализуется специальными аппаратными средствами защиты информации.

Передача данных в компьютерных сетях осуществляется в соответствии со спецификациями сетевых протоколов. “Легальные” траектории распространения информации на этом уровне ограничиваются фильтрацией трафика по ip-адресам и номерам портов, реже – по полезному содержимому сетевых пакетов, сегментацией сетей; на канальном уровне – жесткой привязкой сетевых устройств к локальной вычислительной сети или портам коммутатора (mac filtering, port security). Возможные скрытые каналы по памяти (передача данных в заголовках сетевых пакетов, организация туннелей) контролируются системами обнаружения компьютерных атак, системами предотвращения утечек данных (DLP) и др. Для предотвращения более сложных скрытых каналов, связанных с манипулированием параметрами сигнала, также применяются специальные программно-аппаратные средства защиты информации.

Объекты файловых систем и баз данных являются стандартными контейнерами для хранения информации на системном уровне. Соответственно основным механизмом, обеспечивающим безопасную работу с такими объектами, выступает управление доступом. Современные операционные системы включают механизмы управления доступом, построенные на основе верифицированных моделей, таких, как например МРОСЛ ДП-модель [3]. Контроль скрытых каналов на системном уровне, по сути, осуществляется таким же способом. Практическая реализация монитора безопасности ОС в этом случае предполагает разделение процессов на доверенные и недоверенные. При этом доверенные процессы реализуют функции безопасности и не вступают в кооперацию с недоверенными при реализации запрещенных информационных потоков. Возможность нарушения целостности программ, инициирующих доверенные процессы исключается. Такой подход, на наш взгляд, позволяет решить проблему контроля скрытых каналов в системном программном обеспечении лишь частично. Скрытые каналы, по сути, имеют место в случаях, когда в соответствующей среде передачи (обработки) данных возникают какие-либо эффекты, различимые потенциальным нарушителем. Далеко не все эффекты, возникающие в системном программном обеспечении, имеют отношение к объектам файловой системы, баз данных или сетевым сокетам. Примерами подобных эффектов могут выступать: терминальное состояние программы, временные задержки, возникающие в процессе вычислений, интенсивность использования системных ресурсов, системный кэш и кэш базы данных – структуры, которые могут участвовать в межпроцессном взаимодействии. Второй проблемой, которая уже отмечалась в [4], является доверие к несистемным сервисам, осуществляющим деклассификацию – контролируемое раскрытие информации.

Наибольшую сложность представляет проверка безопасности вычислений на прикладном уровне. На практике проблема полноценно не решается даже в контексте контроля “легальных” траекторий распространения информации. В программном обеспечении такие траектории можно получить из графа потока управления при условии гарантии его целостности. Традиционно с целью обеспечения конфиденциальности и целостности данных в процессе их обработки в программном обеспечении используется сочетание формальных, полуформальных и неформальных методов, используемых на разных стадиях разработки системы. К таким методам относятся: динамический и статический анализ кода, символьное (косимвольное) выполнение, фаззинг-тестирование, формальная верификация. Кроме того, дополнительные защитные меры (рандомизация смещений динамической памяти, выделяемой процессу, защита от исполнения на стеке, контроль целостности потока управления и др.) реализуются на уровне платформы и компилятора. Проверки, реализуемые с использованием перечисленных методов, в основном, не учитывают специфику обрабатываемых данных и бизнес-логику приложения, но играют существенную роль в обеспечении целостности алгоритма. К формальным методам, учитывающим специфику данных, относятся, как говорилось ранее, методы на основе КИП. Полноценное внедрение КИП в языковые платформы в настоящее время не реализовано по ряду причин (см. [4]), однако при разработке защищенных систем все чаще применяется родственный механизм – анализ помеченных данных (Taint Tracking), иногда его относят к одному из видов КИП. Существенным ограничением анализа помеченных данных является его применимость лишь для выявления явных информационных потоков:

Таблица 1. Практическая реализация требований доверия к КС

	Физический уровень	Системный уровень	Уровень специального программного обеспечения
Контроль “легальных” траекторий и сред распространения информации	+	+	±
Контроль скрытых каналов	+	±	–
Формальное доказательство эффективности механизмов ЗИ	+	±	–

```
voidmain (){    voidfoo(z){
    a = new A();    x = z.g;
    b = a.g         w = source() ;
    foo(a);         x.f = w;
    sink ( b.f );   }
}
```

При этом неявные информационные потоки [5], которые, очевидно, также представляют собой “легальные” траектории распространения информации, легко могут быть использованы для обхода данного вида анализа:

```
void copy ( tainted char *src ,
            untainted char *dst , int len) {
  untainted int i, j ;
    for (i = 0; i < len; i++) {
    for (j = 0; j < sizeof(char)*256; j++) {
        if ( src[i] == (char)j )
            dst[i] == (char)j ; //illegal
    }
  }
}
```

Таким образом, с учетом изложенного выше, на текущий момент возможность выполнения условий, критичных для построения доверенных компьютерных систем, в целом может быть представлена как показано в табл. 1. Иные требования, например к резервированию данных [6], в работе воспринимаются как вспомогательные с точки зрения сложности реализации в конкретной системе.

Работа является логическим продолжением и дополнением исследований, описанных в [4], в ней детализирована процедура выявления запрещенных информационных потоков в программных блоках *PL/SQL* СУБД *Oracle* с использованием разработанной при участии автора платформы *PLIF* [7]. Основная идея заключается в формальной верификации сгенерированных на основе реализующих критичные вычисления программных блоков спецификаций *TLA+* инструментом “проигрывания моделей” – *TLC*. Принимая во внимание известное ограничение базового метода – существенное падение производительности с расширением пространства состояний [8], предполагается, что для адекватного абстрагирования проверяемых модулей критичные вычисления будут выноситься разработчиками на отдельный уровень, в нашем случае – уровень хранимых процедур *PL/SQL* (рис. 1). Выбор *PL/SQL* обусловлен несколькими факторами: близость к данным, относительно небольшой объем кода (как правило, не более 60 строк для одной процедуры), отсутствие вспомогательных вычислений, наличие встроенного механизма выявления прямых и косвенных зависимостей и др. Распространение информации от проверенных хранимых процедур и функций до соответствующих элементов графического интерфейса или выходных потоков во внешнем программном слое может жестко контролироваться статическим Taint Tracking анализом. Однако, данный вопрос требует дополнительного изучения. В работе также описана общая схема проверки безопасности вычислений в приложениях уровня предприятия, работающих с реляционными базами данных. Концептуальные сведения о платформе *PLIF* изложены в следующем разделе, после чего рассматривается пример использования платформы для проведения исследования.

2. PLIF. ОБЩИЕ СВЕДЕНИЯ

Внедрение механизма КИП, по мнению автора, должно включать следующие этапы:

– выбор языка описания политики безопасности (уровни и метки доступа, роли и привилегии и т.д.);

– определение безопасности семантики;

– разработку механизма проверки условий безопасности и доказательство его корректности;

– практическую реализацию;

– исследование влияния на написание кода и встраивание анализа в процесс разработки программного обеспечения.

Активные исследования по отдельным направлениям продолжаются уже несколько деся-

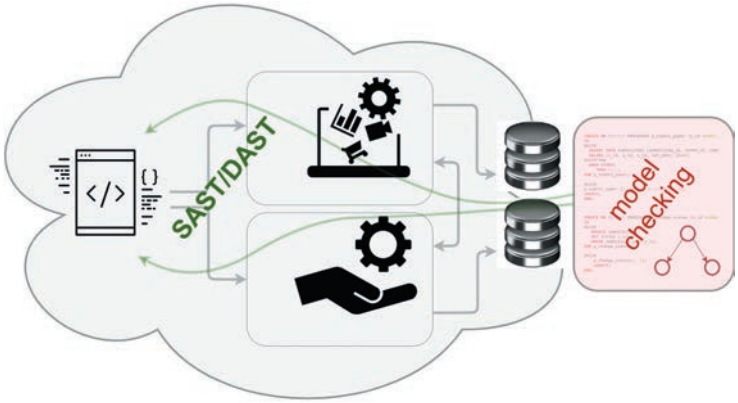


Рис. 1. Выбор приемлемого уровня абстрагирования.

тилетий. Ретроспективный взгляд на них изложен в [4]. Применительно к *PLIF* данные аспекты предстают в следующем виде.

Язык описания политики безопасности. В качестве языка описания политики безопасности выбрано подмножество языка *Paralocks* [9]. К основным его преимуществам следует отнести возможность встраивания в выражения политик условий деклассификации данных, гибкость и возможность интеграции с различными системами управления доступом (ролевыми, мандатными и др.).

В общем виде, как показано в [9], некоторая политика безопасности P_k определяется формулой логики предикатов, представимой в виде конъюнкции определенных дизъюнктов Хорна: $P_k \stackrel{\Delta}{=} C_1 \wedge C_2 \wedge \ldots$, где C_n – предложение политики вида: $\forall x_1,\ldots,x_m . l_1(\cdot) \wedge l_2(\cdot) \wedge l_3(\cdot)\ldots \Rightarrow Flow(u)$ или $\forall x_1,\ldots,x_m . \neg l_1(\cdot) \vee \neg l_2(\cdot) \vee \neg l_3(\cdot)\ldots \vee Flow(u)$, где $Flow(u)$ – предикат, обозначающий поток данных к u (u – связанная переменная или константа, обозначающая пользователя), $l_1 \ldots l_n$ – условия (блокировки), выполнение которых требуется для того, чтобы $Flow(u)$ принял значение $TRUE$. Предикаты $l_1 \ldots l_n$ могут быть параметрическими или непараметрическими.

Для дальнейших рассуждений удобно использовать для предложений политик клаузальную форму: $C_n \triangleq (\neg l_1(\cdot), \neg l_2(\cdot), \neg l_3(\cdot)\ldots, Flow(u))$.

Описание политик в модели (спецификации *TLA+*) требует введения ряда констант: UU – множество допустимых имен связанных переменных, U – множество актеров (конкретных пользователей системы), E_0 – множество имен непараметрических блокировок, E_1 – множество имен параметрических (с одним параметром) блокировок.

В качестве примера рассмотрим выражение политики: “Поток к произвольному пользователю x возможен, если а) открыта блокировка *guest* – для x задана роль *guest* – и открыта блокировка *t_expire* – истек заданный интервал времени **или** б) открыта блокировка *manager* – для x задана роль *manager*. Логически оно может быть представлено как $\forall x.(texpire \wedge guest(x) \Rightarrow Flow(x)) \wedge (manager(x) \Rightarrow Flow(x))$. В спецификации, полученной с помощью *PLIF* [7], данное выражение примет вид:

{⟨*x* , ⟨[*t_expire* ↦ {}],
[*guest* ↦ {*x*}, *reviewer* ↦ {*NONE*},
manager ↦ {*NONE*}, *organizer* ↦ {*NONE*}]⟩⟩,
⟨*x* , ⟨[*t expire* ↦ {*NONE*}],
[*guest* ↦ {*NONE*}, *reviewer* ↦ {*NONE*},
manager ↦ {x}, *organizer* ↦ {*NONE*}]⟩⟩}

Здесь: $x \in U$, $t\_expire \in E_0$, $\{guest, reviewer, manager, organizer\} \subseteq E_1$, $NONE$ – специальное значение, которое обозначает, что открытие блокировки не требуется. Политика определяется как множество двумерных кортежей. Каждый кортеж – отдельное предложение политики, его первым элементом является связанная переменная или константа, обозначающие пользователя, к которому разрешен информационный поток; второй элемент – это двумерный кортеж, с его помощью описывается множество блокировок, которые должны быть открыты. Элементами этого кортежа являются записи, их поля соответствуют именам блокировок, определяемых константами модели, значения полей задают множества фактических параметров блокировок. В *TLA+* записи, кортежи, массивы могут интерпретироваться как функции. Поскольку *TLA+* не поддерживает частично определенных функций, для блокировок, которые не участвуют в выражениях политики, приходится использовать специальные значения $NONE$. Для удобства анализа трасс,

приводящих к возникновению запрещенных информационных потоков, *PLIF* включает отдельную утилиту с графическим пользовательским интерфейсом – *plifparser*. Отображения политик в ней происходит в соответствии с упрощенной нотацией, в которой рассматриваемый пример выглядит так:

```
x: manager(x)
x: guest(x), t_expire
```

Отношение частичного порядка $\sqsubseteq$ на множестве политик задается следующим образом: $P_1 \sqsubseteq P_2$, если $\forall c_2 \in P_2 : \exists c_1 \in P_1 : c_1 \sqsubseteq c_2$ (1). С точки зрения логики $P_1 \sqsubseteq P_2$ можно интерпретировать, как $P_1 \vDash P_2$ (2). В [9] показано, что условие (1) является необходимым и достаточным для истинности выражения (2). Сравнение предложений политик в [9] описывается алгоритмически, через набор правил. Однако определение частичного порядка на данном множестве может иметь формальную замкнутую форму.

Пусть $C_1 = \{\neg l_1(\cdot), \neg l_2(\cdot), \ldots, \neg l_n(\cdot), Flow(u_1)\}$, $C_2 = \{\neg m_1(\cdot), \neg m_2(\cdot), \ldots, \neg m_k(\cdot), Flow(u_2)\}$, тогда $C_1 \sqsubseteq C_2$, если $\{\neg l_1(\cdot)\tau, \neg l_2(\cdot)\tau, \ldots, \neg l_n(\cdot)\tau, Flow(u_1)\tau\}\sigma \subseteq \{\neg m_1(\cdot), \neg m_2(\cdot), \ldots, \neg m_k(\cdot), Flow(u_2)\}\sigma$, где τ – операция переименования связанных переменных в термах (блокировках) соответствующего предложения политики, σ – наиболее общий унификатор (НОУ) $Flow(u_1)$ и $Flow(u_2)$ при условии, что u_1 и u_2 – связанные переменные или u_2 – константа. Операция определения НОУ является стандартной, ее подробное описание можно найти в литературе, посвященной методу резолюций.

Максимальная нижняя грань выражений политик определяется через объединение их предложений: $GLB(P_1, P_2) \triangleq P_1 \cup P_2$.

Минимальная верхняя грань в общем виде определяется следующим образом:

$$LUB(P_1, P_2) \triangleq$$

$$\triangleq \bigcup_{\substack{(\neg l_1(\cdot),\ldots\neg l_n(\cdot), Flow(u_1),\\ \neg m_1(\cdot),\ldots\neg m_k(\cdot), Flow(u_2)) \in P_1 \times P_2:\\ \exists\sigma.\sigma=(Flow(u_1), Flow(u_2))}} (\neg l_1(\cdot)\tau, \ldots, \neg l_n(\cdot)\tau, Flow(u_1)\tau, \neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(u_2))\sigma$$

В данном случае при поиске НОУ никаких ограничений на параметр $Flow(\cdot)$ не накладывается.

Описанные в [9–11] варианты использования *Paralocks* для кодирования политик элементов среды вычислений, преемственных к существующим механизмам управления доступом (например, ролевому), позволяют сделать вывод о возможности внесения допущений, которые упрощают описание соответствующих сущностей в спецификациях *TLA+*. Положим, что множество имен связанных переменных *UU* включает один элемент, также будем считать, что, если связанная переменная появляется в одной из блокировок некоторого предложения политики, то эта же переменная является аргументом и в литерале $Flow(\cdot)$ того же предложения, и, если связанная переменная является аргументом $Flow(\cdot)$ некоторого предложения политики, то все параметрические блокировки того же предложения также будут принимать эту переменную в качестве аргумента.

С учетом заданных предположений, а также того, что $Flow(\cdot)$ является однопараметрической блокировкой, и в качестве ее параметра может выступать либо связанная переменная, либо константа, имеем:

$$LUB(P_1, P_2) \triangleq$$

$$\cup\{(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), \neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(x)) :$$
$$\wedge(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), Flow(x)) \in P_1$$
$$\wedge(\neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(x)) \in P_2\}$$
$$\cup\{(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), \neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(a)) :$$
$$\wedge(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), Flow(a)) \in P_1$$
$$\wedge(\neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(a)) \in P_2\}$$
$$\cup\{(\neg l_1(\cdot), \ldots, \neg l_n(\cdot),$$
$$\neg m_1(\cdot)[x := a], \ldots, \neg m_k(\cdot)[x := a], Flow(a)) :$$
$$\wedge(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), Flow(a)) \in P_1$$
$$\wedge(\neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(x)) \in P_2\}$$
$$\cup\{(\neg l_1(\cdot)[x := a], \ldots,$$
$$\neg l_n(\cdot)[x := a], \neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(a)) :$$
$$\wedge(\neg l_1(\cdot), \ldots, \neg l_n(\cdot), Flow(x)) \in P_1$$
$$\wedge(\neg m_1(\cdot), \ldots, \neg m_k(\cdot), Flow(a)) \in P_2\}$$

В табл. 2 с использованием упрощенной нотации показано несколько примеров работы оператора сравнения на множестве политик, в табл. 3 – вычисление минимальной верхней грани.

Множество возможных предложений политик *ClausesSet* и множество самих политик *PoliciesSet* с

Таблица 2. Пример работы оператора $\sqsubseteq$

P_1	P_2	$P_1 \sqsubseteq P_2$
x: manager(x)	x: reviewer(x)	FALSE
x: reviewer(x)	x: manager(x)	FALSE
x: manager(x)	x: manager(x), t_expire	TRUE
x: manager(x)	bob: manager(bob)	TRUE

Таблица 3. Пример работы оператора LUB

P_1	P_2	$P_1 \sqcup P_2$
x: manager(x)	x: reviewer(x)	x: manager(x), reviewer(x)
alice	x: reviewer(x)	alice: reviewer(alice)
x: manager(x) x: reviewer(x)	x: t_expire	x: manager(x), t_expire x: reviewer(x), t_expire
bob	alice: t_expire	$\top$

учетом изложенного выше в спецификации *Paralocks.tla* [7] определяется как:

ClausesSet $\triangleq$
{$c \in \{\langle u, \langle e0, e1\rangle\rangle$
: $u \in (U \cup UU)$,
$e0 \in [E0 \to \text{SUBSET } \{NONE\}]$,
$e1 \in [E1 \to ((\text{Subset } (U \cup UU))$
$\setminus \{\{\}\}) \cup \{\{NONE\}\}]\}$:
$\vee\, c[1] \in UU$
$\vee \wedge\, c[1] \notin UU$
$\wedge$ (UNION *Range*(c[2][2]))
$\cap\, UU = \{\}$

PoliciesSet $\triangleq$
{$p \in$ subset *ClausesSet* :
$\forall\, c1 \in p$:
$\neg\exists\, c2 \in p$:
$\wedge\, c1 \neq c2$
$\wedge \vee$ (*compareClause*($c1$, $c2$)
$\vee$ *compareClause*($c2$, $c1$))}
$\cup \{\{\}\}$

Определение операторов: *compareClause*($\cdot$), *comparePol*($\cdot$), в данной работе не разбирается. Заданное описанным образом конечное множество выражений политик образует полную решетку. Доказательство соответствующих свойств проведено с использованием формальной системы для вывода доказательств *TLAPS* в */plif_specs/plif_lattice/Paralocks.tla* (см. [7]).

THEOREM *Reflexivity* $\triangleq$
$\forall\, p \in$ *PoliciesSet* : *comparePol*(p, p)

THEOREM *Transitivity* $\triangleq$
$\forall\, p1, p2, p3 \in$ *PoliciesSet* :
$\wedge$ *comparePol*($p1$, $p2$)
$\wedge$ *comparePol*($p2$, $p3$) $\Rightarrow$ *comparePol*($p1$, $p3$)

THEOREM *Antisymmetry* $\triangleq$
$\forall\, p1, p2 \in$ *PoliciesSet* :
$\wedge$ comparePol($p1$, $p2$)
$\wedge$ *comparePol*($p2$, $p1$) $\Rightarrow p1 = p2$

THEOREM *ParalocksLattice* $\triangleq$
$\forall\, p1, p2 \in$ *PoliciesSet* :
$\wedge$ *comparePol*($p1$,*LUB*($p1$, $p2$))
$\wedge$ *comparePol*($p2$,*LUB*($p1$, $p2$))
$\wedge \forall\, y \in$ *PoliciesSet* :
$\wedge$ *comparePol*($p1$, y)
$\wedge$ *comparePol*($p2$, y)
$\Rightarrow$ *comparePol*(*LUB*($p1$, $p2$), y)

Безопасность семантики. За основу принимается понятие прогресс-зависимого информационного невлияния [12]. Данная схема имеет несколько отличительных особенностей по сравнению со строгой схемой информационного невлияния, более подробные сведения можно найти в [4]. С целью сохранить целостность изложения приведем все же формальное определение.

Определение 2.1 *Программа P удовлетворяет свойству прогресс-зависимого информационного невлияния для начального и конечного отображений множества переменных на множество меток безопасности M_1 и M_2 – ПЗИН$(P)_{M_1,M_2}$, если для любых двух состояний S_1 и S_2 среды вычислений, состоящих в отношении низкой эквивалентности относительно некоторого уровня конфиденциальности L при отображении M_1: а) каждый шаг вычислений сопровождается генерацией одинаковых наблюдаемых значений относительно уровня L или приводит к "расхождению" для обоих состояний; б) соответствующие финальные состояния – S_1' и S_2' находятся в отношении низкой эквивалентности относительно уровня L при отображении M_2* (рис. 2):

$$\text{ПЗИН}(P)_{M_1,M_2} \triangleq \forall L, S_1, S_2, o_1, o_2 : S_1 \approx_{M_1,L} S_2 \wedge$$
$$\wedge\, P(S_1) \downarrow \langle S_1', o_1\rangle \wedge P(S_2) \downarrow \langle S_2', o_2\rangle \Rightarrow \qquad (2.1)$$
$$\Rightarrow o_1 \approx_{L(d)} o_2 \wedge S_1' \approx_{M_2,L} S_2'$$

Здесь $o_1 \approx_{L(d)} o_2$ означает эквивалентность наблюдаемых серий значений (поведений) относительно уровня L (с учетом деклассификации). Подробнее понятие деклассификации данных рассматривается далее, а также в [4]. Предполагается что, два состояния среды вычислений находятся в отношении низкой эквивалентности относительно заданного уровня конфиденциальности L, если все пары одноименных элементов с меткой, не превышающей L, обладают одинаковыми значениями. Условие *б*) легко проверяется для отдельных выражений и команд *PL/SQL* с ис-

пользованием заданных для них правил абстрактной семантики и является важным для формального доказательства безопасности вычислений в целом в условиях потокочувствительности и отсутствия ограничений на количество исполняемых блоков в одном сеансе [4]. Важную роль в определении также играют начальное и конечное отображения множества элементов на множество меток (политик) безопасности M_1 и M_2, назовем их абстрактными состояниями. Например, в простейшем случае, когда в системе имеется лишь один сеанс, и множество программных блоков ограничено лишь одной процедурой вида:

```
PROCEDURE proc_1 ( hi number) IS
    var 1 number;
    f1 utl__file.file_type;
BEGIN
        select col_1 into var_1 from Tab1;
    f1 :=
    utl__file.file_open('DIR', 'file name' ,wb);
    utl_file.put(f1, var_1);
        update Tab1
    set col_1 = hi
    where id = n;
END;
```

в которой внешний объект **file_name** обладает неизменяемой политикой $\bot$, начальная политика глобальной переменной **col_1** – $\bot$[1] и политика формального параметра **hi** – $\top$, отсутствие запрещенных информационных потоков при однократном выполнении процедуры позволяет сделать вывод о безопасности вычислений лишь для начального абстрактного состояния M_1, в котором глобальная переменная **col_1** имеет политику $\bot$, но не $\top$. Поэтому безопасная композиция программного блока с собой невозможна. Предложенный механизм проверки условий безопасности вычислений учитывает это обстоятельство см. Раздел 3.

Проверка условий безопасности. Как описано в [4], механизм проверки основан на формальной верификации программных блоков методом “проигрывания моделей” – *model checking*. В качестве языковой платформы спецификаций выбран язык *TLA+*. Важными его преимуществами являются: полноценная поддержка темпоральной логики (дает возможность более точно описывать поведение системы), а также возможность проверки свойств переходов, что важно в контексте предложенного алгоритма проверки модели вычислений (см. Раздел 3). Свойство перехода CompInv, используемое в данном алгоритме, может быть выражено как:

[1] Под глобальными переменными моделируемой среды вычислений понимаются атрибуты отношений.

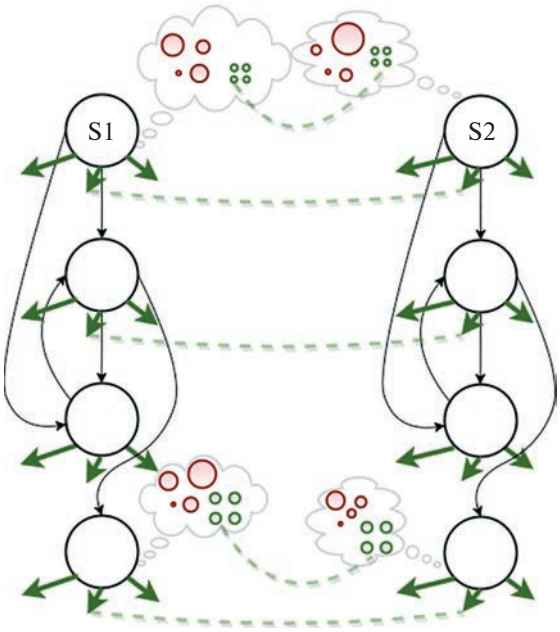


Рис. 2. Схема ПЗИН.

$$
\begin{array}{l}
VPolUnchanged \triangleq \\
\quad \text{LET } CompInv\_OP1(x, y) \triangleq \\
\qquad \wedge\ x \\
\qquad \wedge\ comparePol(VPol[y].policy,\ VPol'[y].policy) \\
\qquad \wedge\ comparePol(VPol'[y].policy,\ VPol[y].policy) \\
\quad \text{IN } FoldSet(CompInv\_\text{OP1}, \text{TRUE}, \text{DOMAIN } VPol) \\
\text{CompInv} \triangleq \Box[VPolUnchanged]_{vars}
\end{array}
$$

VPol – переменная состояния, содержащая политики глобальных переменных – столбцов. Кроме того, для проверки спецификаций *TLA+* методом “проигрывания моделей” специально разработан инструмент *TLC*.

Абстрактная семантика информационных потоков в программных блоках, выполняемых в параллельных пользовательских сеансах, представлена в [4]. Ее правила дополнены выражениями *inv*, символизирующими необходимые проверки на определенных этапах вычислений. Например, правило (C-IF):

$$
\frac{\langle e, M\rangle \Downarrow p \quad \text{inv} : e - local}{\left\langle PC, \begin{array}{l}\textbf{if } e \textbf{ then } c_1 \\ \textbf{else } c_2\end{array}, M\right\rangle \rightarrow \langle p :: PC, c_1 \vee c_2, M\rangle}
$$

говорит о том, что при выполнении условного оператора политика элемента *PC* – счетчика команд – дополняется политикой p условного выражения e, далее осуществляется равновероятный переход либо к инструкции c_1, либо к ин-

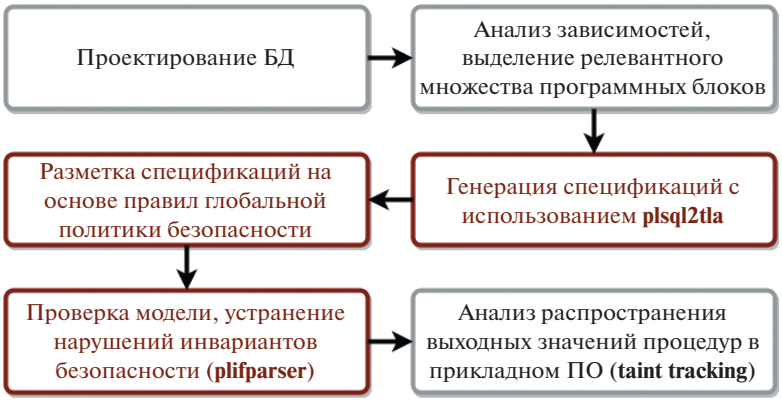


Рис. 3. Общая процедура исследования.

струкции c_2, абстрактное состояние среды вычислений при этом не изменяется. Для удобства манипулирования политикой PC с учетом возможной вложенности она представляется как кортеж отдельных значений. Очевидно, актуальное значение политики PC для каждого сеанса рассчитывается как минимальная верхняя грань входящих в соответствующий кортеж выражений. Ограничение $e - local$ (проверяется вручную) необходимо для исключения запрещенных вероятностных информационных потоков. Соответственно, правило (C-END-IF):

$$\frac{}{\langle p :: PC, \mathbf{endif}, M\rangle \rightarrow \langle PC, \mathbf{null}, M\rangle}$$

указывает лишь на ослабление политики PC при выходе из условия.

Общее описание системы переходов, моделирующей параллельные вычисления в сеансах работы с базой данных (БД), также можно найти в [4]. Реализованный анализ, как уже отмечалось, является потокочувствительным.

Важным этапом проверки является автоматическая трансформация кода PL/SQL в спецификации $TLA+$. Эта задача решается с использованием разработанной утилиты *plsql2tla*. В ее основе лежит подход, предложенный в [13].

В случае нарушения заданного инварианта безопасности или свойства перехода проблемную трассу и граф информационных потоков можно исследовать с помощью инструмента *plifparser*. Данный инструмент был разработан на основе [14]. В четвертом разделе показан пример его использования.

3. ОБЩАЯ ПРОЦЕДУРА ИССЛЕДОВАНИЯ

В общем виде этапы процедуры анализа информационных потоков в программном обеспечении промышленной информационной системы, построенной на основе СУБД, представлены на рис. 3.

Проектирование базы данных рекомендуется осуществлять таким образом, чтобы конфиденциальные данные размещались компактно – в ограниченном наборе таблиц. Данная рекомендация не противоречит общепринятым принципам безопасной разработки, и при этом позволит более эффективно изолировать критичные вычисления от общего кода PL/SQL.

Следующим этапом является анализ зависимостей и выделение релевантного множества программных блоков. На этом этапе предполагается определение тех блоков PL/SQL, которые имеют отношение к манипулированию конфиденциальными данными. Важной особенностью современных промышленных СУБД является наличие встроенных механизмов выявления прямых и косвенных зависимостей.

Ключевыми и наиболее сложными этапами процедуры являются: генерация и разметка спецификаций, проверка модели и устранение нарушений инвариантов безопасности. Данные этапы разберем далее на конкретном примере. Акцент будет сделан на алгоритме проверки модели вычислений (рис. 4). Предварительный шаг алгоритма Инициализация предполагает инициализацию констант, определяющих количество сеансов, множество конкретных пользователей, множество имен связанных переменных, множества параметрических и непараметрических блокировок и т.д. При проигрывании модели проверяются два свойства: главный инвариант безопасности *ParalocksInv* – гарантирует выполнение условия *a)* Определения 0.1 для отдельных команд и выражений – и свойство перехода *CompInv* – гарантирует эквивалентность начального и конечного отображений множества глобальных переменных на множество политик. В случае нарушения *ParalocksInv* принимается решение о деклассификации данных, в некоторых случаях требующей ис-

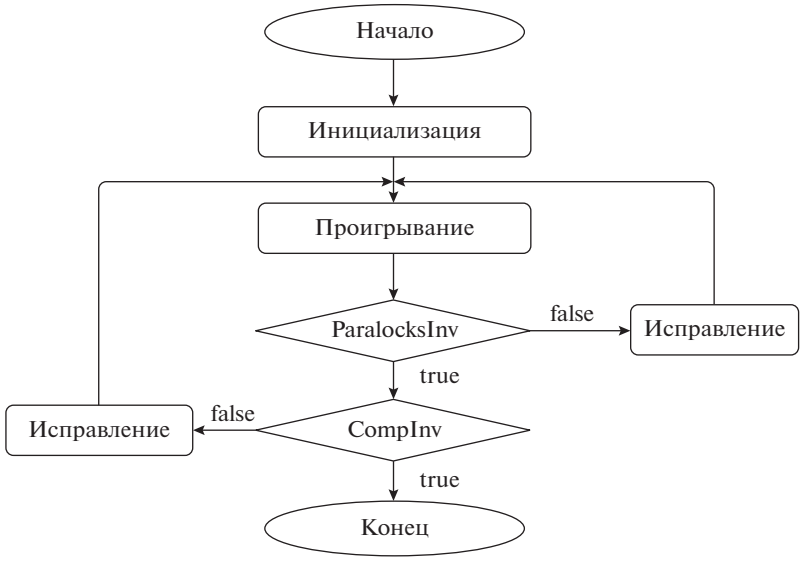


Рис. 4. Алгоритм проверки модели.

правления кода или изменения привилегий на доступ к объектам БД (рассматривается далее). При нарушении свойства *CompInv* в начальном состоянии модели повышается политика соответствующей глобальной переменной – столбца[2]. Приведение политик глобальных переменных к стационарным значениям позволяет проверить выполнение условия *a*) Определения 2.1 для отдельных команд и выражений *PL/SQL* для всех возможных начальных и конечных абстрактных состояний M_1 и M_2, что, в свою очередь, требуется для доказательства корректности алгоритма проверки модели вычислений.

В [4] доказано, что успешное завершение алгоритма гарантирует безопасность вычислений в смысле *ПЗИН* при неограниченном количестве программных блоков, выполняемых в каждом пользовательском сеансе.

Завершающим этапом процедуры является анализ распространения выходных значений критичных процедур и функций в прикладном программном обеспечении.

4. PLIF. ПРИМЕР

Для апробации *PLIF* использовался пример из [4]. Пусть имеется система управления конференциями. Пользователи системы регистрируют доклады, которые рецензируются и распределяются по секциям. На уровне СУБД *Oracle* действует ролевой механизм управления доступом. Иерархия ролей представлена на рис. 5.

Основные бизнес-функции в системе реализуются с помощью хранимых программных блоков *PL/SQL*. Общие требования по безопасности формулируются, например, следующим образом: любой зарегистрированный пользователь может подать заявку на участие в конференции с докладом. Анализ тезисов осуществляется рецензентами, которые, однако, не могут ассоциировать работы с авторами. Программа конференции формируется менеджером, ему доступен по чтению статус любого доклада (одобрен или нет), оставшимся пользователям статусы докладов могут быть доступны лишь по истечению заданного интервала времени. Программу конференции могут просматривать все пользователи. Авторы работ должны оставаться недоступными по чтению для всех пользователей до истечения определенного интервала времени.

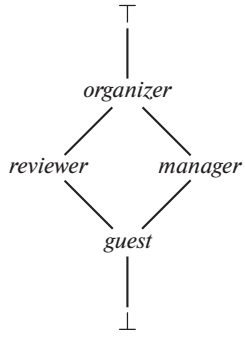


Рис. 5. Решетка ролей.

[2] Выполнения свойства CompInv можно добиться за конечное количество итераций, поскольку алфавит политик представляет собой конечную решетку, а переходные функции вычисления политик глобальных переменных являются монотонно возрастающими.

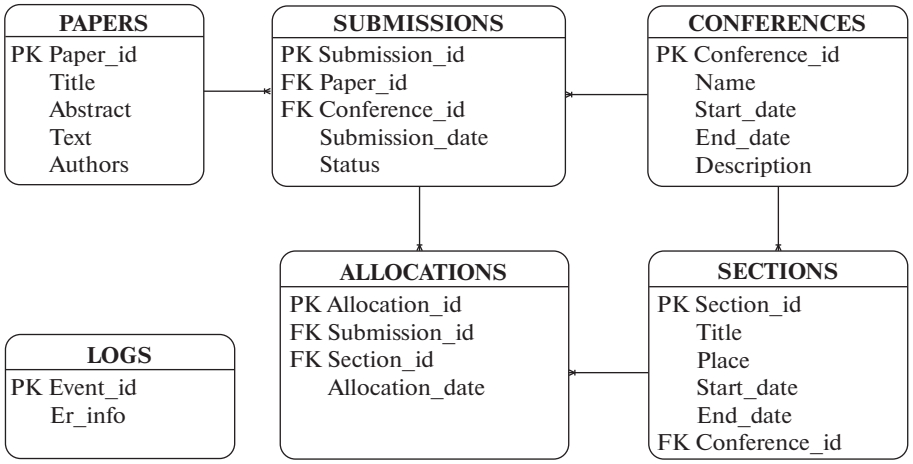


Рис. 6. Структура БД.

Пусть в системе запрещен прямой доступ к таблицам, взаимодействие возможно лишь с использованием хранимых процедур и функций. Положим также, что для поддержания описанных требований по безопасности на уровне БД заданы следующие объектные привилегии:

grant execute on p_add_paper to public;

grant execute on p_submit_paper to public;

grant execute on p_chahge_status to reviewer;

grant execute on p_allocate to manager;

grant execute on f_getsection_program to public;

grant execute on f_getpaper to public;

grant execute on f_is_accepted to public.

После завершения этапа **проектирования БД**, в соответствии с описанной процедурой (см. рис. 3), следует определить таблицы, предназначенные для хранения конфиденциальных данных, и выявить зависимые от них объекты БД. В СУБД *Oracle* для этих целей существует отдельный механизм. Например, для получения прямых и косвенных зависимостей, ассоциированных с таблицей *Submissions* – содержит статусы зарегистрированных для участия докладов – можно выполнить следующие команды:

execute deptree_fill('TABLE', 'CONFERENCE',
'SUBMISSIONS');

select * from deptree.

Положим, на **втором этапе** удалось выявить релевантное множество объектов БД. В него входят отношения: *Papers* – доклады, *Submissions* – зарегистрированные заявки на участие, *Conferences* – конференции, *Sections* – секции, *Allocations* – распределение докладов по секциям, *Logs* – журнал ошибок (рис. 6); и программные блоки: *p_add_paper* – добавляет новый доклад в БД, *p_submit_paper* – регистрирует доклад для участия в конференции (добавляет строку в таблицу *Submissions*), *p_chahge_status* – изменяет значение поля *Status* для заданной строки в таблице *Submissions* после рецензирования соответствующего доклада, *p_allocate* – добавляет сведения о докладе в таблицу *Allocations* после осуществления ряда проверок, *f_getsection_program* – возвращает программу заданной секции, для формирования отчета обращается к таблицам: *Papers* и *Allocations*, *f_get_paper* – возвращает сведения о заданном докладе, *f_is_accepted* – проверяет статус доклада и возвращает значение *TRUE*, если доклад одобрен. **Генерация спецификаций** осуществляется с помощью утилиты *plsql2tla*. В данной работе этот этап подробно не рассматривается.

Далее необходимо выполнить **разметку спецификаций** на основе правил глобальной политики безопасности. Как может выглядеть результат выполнения данного этапа, показано в табл. 4. Остановимся на этом немного подробнее.

Прежде всего, опираясь на заданные требования глобальной политики и иерархию ролей БД (рис. 5), определим множества параметрических и непараметрических блокировок E_1 и E_0. Пусть $E_0 \triangleq \{$“*t_expire*”$\}$, $E_1 \triangleq \{$“*guest*”,“*reviewer*”, “*manager*”, “*organizer*”$\}$. К процедуре *p_submit_paper* администратором БД предоставлен общий доступ, поэтому для всех формальных параметров справедлива политика *any_caller* ($\boxed{a}$) или формально:

$$any\_caller\,(a) \triangleq$$
$$\{\langle a, \langle [e1 \in E0 \mapsto \{NONE\}],$$
$$[e2 \in E1 \mapsto \{NONE\}]\rangle\rangle\}$$

Таблица 4. Разметка спецификаций

Программный блок	Входные значения	Формальные параметры	Локальные переменные	Выходные значения
create or replace procedure p_submit_paper s_id number p_id number, c_id number, sub_date_date, stat number) is begin ... end p_add_paper;	**i1:=** x **i2:=** x **i3:=** x **i4:=** x **i5:=** x	**s_id :=** a **p_id :=** a **c_id :=** a **sub_date :=** a **stat :=** a		
create or replace. procedure p_add_paper (p_id number, tit varchar2, abstr varchar2, t clob, auth varchar2) is begin ... end p_add_paper;	**i1:=** x **i2:=** x **i3:=** x **i4:=** x **i5:=** x : t_expire	**p_id:=** a **tit :=** a **abstr :=** a **t :=** a **auth :=** a		
create or replace procedure p_change_status (s_id number, stat number) is begin ...; end p_change_status;	**i1:=** x **i2:=** x : manager(x) x : t_expire	**s_id :=** a : reviewer(a) **stat :=** a : reviewer(a)		
create or replace procedure p_allocate (id number, s_id number, sec_id number, alloc_date number) is p_not_accepted exception v_p_id exception v_is_acc exception begin ... end p_allocate;	**i1:=** x **i2:=** x **i3:=** x **i4:=** x	id := a : manager(a) s_id := a : manager(a) **sec_id :=** a : manager(a) **alloc_date:=** a : manager(a)	**p_not_accepted:=** x **v_p_id :=** x **v_is_acc :=** x	

Таблица 4. Окончание

Программный блок	Входные значения	Формальные параметры	Локальные переменные	Выходные значения
create type paper_type ...; create or replace function f_get_paper (p_id number) return paper_type as v_paper paper_type begin ... return v_paper; end f_get_paper;	**i1:=** x	**p_id:=** a	**v_paper_c1:=** x **v_paper_c2:=** x **v_paper_c3:=** x **v_paper_c4:=** x **v_paper_c5:=** x	**r_rec_c1:=** a **r_rec_c2:=** a **r_rec_c3:=** a **r_rec_c4:=** a **r_rec_c5:=** a

create or replace function
f_is_accepted
(s_id number)
return boolean
is
v_status number;
begin
select status
into v_status
from submissions
where
submission_id = s_id;
if v_status = 1
then return true;
else return false;
end if;
end f_is_accepted;
	i1:= x	**s_id :=** a	**v_status:=** x	**r:=** a

create type paper_type
as object(...);
create type
program_arr_type
as varray(1000)
of paper_type;
create or replace
function
f_get_section_program
(s_id number)
return program_arr_type
as
(v_program)
program_arr_type;
begin
...
end;
	i1:= x	**s_id:=** a	**v_prog_ar_e1_c1:=** x
			v_prog_ar_e1_c2:= x
			v_prog_ar_e1_c3:= x
			v_prog_ar_e1_c4:= x
			v_prog_ar_e1_c5:= x
			v_prog_ar_e2_c1:= x
			v_prog_ar_e2_c2:= x
			v_prog_ar_e2_c3:= x
			v_prog_ar_e2_c4:= x
			v_prog_ar_e2_c5:= x
				r_ar_e1_c1:= a
				r_ar_e1_c2:= a
				r_ar_e1_c3:= a
				r_ar_e1_c4:= a
				r_ar_e1_c5:= a
				r_ar_e2_c1:= a
				r_ar_e2_c2:= a
				r_ar_e2_c3:= a
				r_ar_e2_c4:= a
				r_ar_e2_c5:= a

Здесь a – символизирует элемент множества U, представляет собой конкретного пользователя сеанса. Описание общей политики безопасности также не содержит ограничений на какие-либо данные, вводимые пользователем при вызове процедуры, поэтому все входные значения обладают минимальной политикой x:

$$min \triangleq \{\langle \text{CHOOSE } x \in UU : \text{TRUE}, \langle [e1 \in E0 \mapsto \{NONE\}], [e2 \in E1 \mapsto \{NONE\}]\rangle\rangle\}$$

Разметка программного блока *p_add_paper* во многом аналогична. Исключение составляет входное значение параметра *auth*, ему назначается политика x: t_expire, поскольку в общих требованиях сказано, что сведения об авторах работ должны оставаться недоступными до истечения заданного интервала времени. Исходя из заданных привилегий на вызов процедуры *p_change_status* политика ее формальных параметров определяется как a: reviewer(a) при этом передаваемое при ее вызове фактическое значение статуса доклада $i2$ должно быть доступно по чтению *manager* и всем остальным пользователям по истечению временного интервала, следовательно $i2 :=$ x : manager(x) / x : t_expire. Входные значения процедуры *p_allocate* не являются конфиденциальными в соответствии с представленными выше требованиями, соответственно, обладают минимальной политикой. Политика формальных параметров a: manager(a) продиктована заданными на уровне системы объектными привилегиями. Отметим, что данная процедура также включает локальные элементы. Локальным переменным всегда назначается минимальная начальная политика, политика исключения *p_not_accapted* в данном случае также минимальна. Общедоступные функциии *f_get_paper* и *f_get_section_program* представляют интерес в части описания составных и ссылочных локальных переменных и возвращаемых значений. Так переменная *v_paper* и возвращаемое значение функции *f_get_paper* обладают объектным типом, состоящим из пяти полей. Каждое поле объектного типа или записи в *PLIF* моделируется отдельным элементом. Аналогично переменная *v_program* и возвращаемое значение функции *f_get_section_program* являются коллекциями объектов. Любая коллекция, ассоциированная с набором кортежей БД, в текущей версии *PLIF* моделируется двухэлементным массивом. Потеря точности автоматического анализа, как ожидается, может быть компенсирована удобством графического интерпретатора трасс вычислений, приводящих к нарушению инварианта безопасности (см. далее).

Ключевым этапом процедуры, конечно, является **проверка модели** и устранение нарушений инвариантов безопасности. В данном примере для завершения алгоритма потребовалось 8 итераций. Показанные далее графы информационных потоков были сгенерированы с помощью утилиты *plifparser*. Проигрывание модели осуществлялось в *TLC* при следующих значениях основных параметров инициализации (констант):

$U \triangleq \{allen, bob, allex, john\}$

$UU \triangleq \{\text{x}\}$

$E0 \triangleq \{\text{“t_expire”}\}$

$E1 \triangleq \{\text{“guest”}, \text{“reviewer”}, \text{“manager”}, \text{“organizer”}\}$

$GPol \triangleq$ (“organizer” :> {“manager”, “reviewer”, “guest”})@@
(“manager” :> {“guest”})@@
(“reviewer” :> {“guest”})@@
(“guest” :> {“guest”})

$Session\_number \triangleq 1$

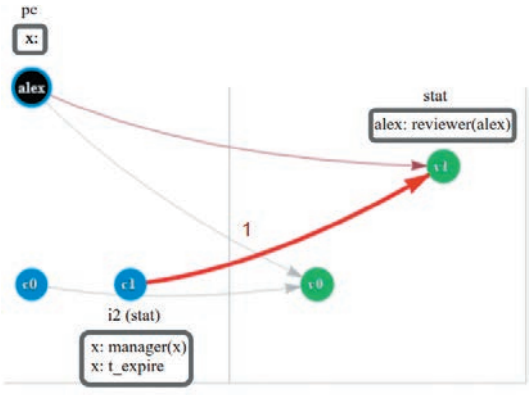
Здесь функция *GPol* задает отображение множества ролей на множество подмножеств ролей, соответствует иерархии на рис. 5 и позволяет осуществлять автоматическое открытие зависимых блокировок при проигрывании модели. *Session_number* – количество моделируемых параллельных сеансов.

ШАГ 1. Нарушение ParalocksInv

При первой попытке проигрывания модели возникает нарушение инварианта безопасности на этапе выполнения оператора *p_change_status_load* – загрузке в сеанс параметров и локальных переменных процедуры *p_change_status*[3] (рис. 7). Вычисления на данном шаге выполняются в соответствии с правилом (C-EXT-PRC) абстрактной семантики, описанной в [4]:

$$\frac{\langle e_1, M\rangle \Downarrow p_1 \ldots \langle e_n, M\rangle \Downarrow p_n \quad \langle x_1, M\rangle \Downarrow p_{x_1} \ldots \langle x_n, M\rangle \Downarrow p_{x_n} \quad \text{inv} : p_1 \sqsubseteq p_{x_1} \ldots p_n \sqsubseteq p_{x_n}}{\langle PC, x_f(x_1 \to e_1, \ldots x_n \to e_n), M\rangle \to \langle PC, \textbf{null}, M[x_1 \mapsto p_1, \ldots x_n \mapsto p_n]\rangle}$$

[3] Генерируемые *PLIF* спецификации описывают состояние каждого пользовательского сеанса, которое включает область локальных переменных и параметров – стек, указатели на текущий кадр стека и текущую выполняемую в сеансе инструкцию, а также область возвращаемых значений.



Session	TLA+	PLSQL
alex	p_change_status_load	change_status_load

Рис. 7. ParalocksInv is violated (p_change_status_load).

Причиной нарушения является несоответствие политик входного значения *i*2(*stat*) – `x : manager(x)` `x : t_expire` и формального параметра *stat* – `a : reviewer(a)`. Для устранения проблемы применяется процедура деклассификации. В литературе описано несколько принципов и способов деклассификации данных в программном обеспечении [15]. В данном случае предполагается, что рецензирование докладов осуществляется анонимно, и рецензент ограничен (на прикладном уровне) только текстом доклада (без имени автора и номера заявки), определяющим фактором является именно расположение проблемного выражения. Поэтому выявленное нарушение можно устранить, применив деклассификацию с ограничением по месту (*WHERE*). Для исправления модели достаточно в соответствующем операторе установить флаг *Ignore*, исправление кода самой процедуры не требуется.

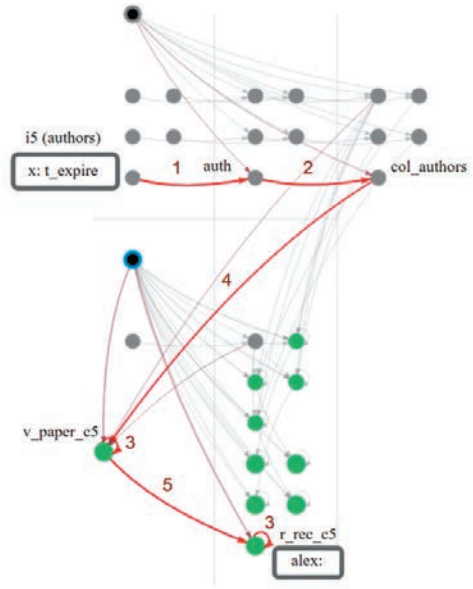
$p\_change\_status\_load(id) \triangleq \ldots \wedge Ignore' = 1$

$ParalocksInv \triangleq$ IF $Ignore \neq 1$
THEN ... ELSE TRUE

Другие допустимые в *PLIF* способы деклассификации применяются для исправления модели на следующих итерациях.

ШАГ 2. Нарушение ParalocksInv

После повторного запуска утилиты проигрывания модели инвариант безопасности нарушается при переходе в операторе *f_get_paper_8* (рис. 8). Анализ графа информационных потоков позволяет легко выявить проблемную траекторию: 1 – политика входного значения *i5 (authors)* –



bob	p_add_paper_load	add_paper_load
bob	p_add_paper4	insert into PAPERS (PAPER_ID, TITLE, ABSTRACT, TEXT, AUTHORS)values (p_id, tit, absr, t, auth)
bob	p_add_paper_exit	add_paper_exit
alex	f_get_paper_load	get_paper_load
alex	f_get_paper_5	select PAPER_ID, TITLE, ABSTRACT, TEXT, AUTHORS into v_paper from PAPERS where PAPER_ID = p_id
alex	f_get_paper_8	return v_paper

Рис. 8. ParalocksInv is violated (p_get_paper_8).

`x : t_expire` попадает в параметр *auth* процедуры *p_add_paper* (выполняется в процессе загрузки процедуры в сеансе *bob*), 2 – обновленная политика параметра *auth* передается в глобальную переменную (столбец) *col_authors*, 3 – происходит загрузка функции *f_get_paper* в сеансе *alex*, 4 – политика столбца *col_authors* передается во внутреннюю переменную *v_paper_c5*, 5 – политика переменной *v_paper_c5* передается в возвращаемое значение *r_rec_c5*, обладающее стационарной политикой `alex`. Таким образом, нарушается условие, заданное правилом (C-EXT-RET) абстрактной семантики:

$$\frac{\langle e, M\rangle \Downarrow p_1 \quad \langle x_{out}, M\rangle \Downarrow p_2 \quad \text{inv} : p_1 \sqcup pc_1 \ldots pc_n \sqsubseteq p_2}{\langle PC, \textbf{return}(x_{out} \to e), M\rangle \to \langle PC, \textbf{null}, M\rangle}$$

Для того, чтобы информационный поток в данной точке стал разрешенным, можно открыть

блокировку *t_expire*. Фактически это означает исправление кода процедуры путем добавления проверки условия:

```
create or replace function f_get_paper
...
is
   begin
      if is time expire()
         then ... else return null;
end p_get_paper;
```

Формально в этом случае применяется деклассификация *WHEN*. Исправление модели требует использования оператора *openLock*, который позволяет добавить во множество открытых блокировок сеанса новый элемент:

$f\_get\_paper\_load(id) \triangleq$

...

$\wedge\ openLock(id, \{[t\_expire \mapsto ALL]\})$

...

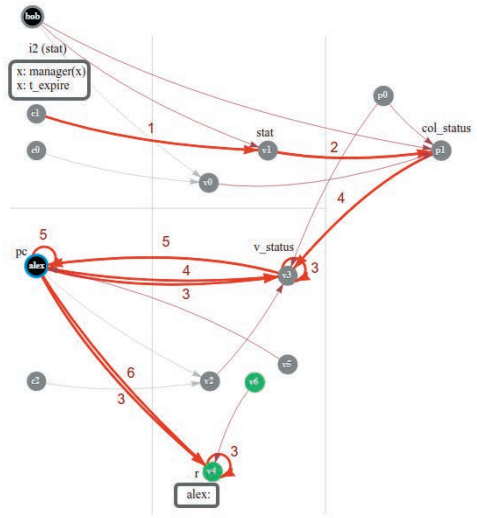
ШАГ 3. Нарушение ParalocksInv

Третье проигрывание модели также завершается нарушением инварианта безопасности, в этот раз ошибка возникает в операторе *f_is_accepted_9*. Поскольку доступ к этой функции предоставлен всем пользователям, ее возвращаемое значение имеет политику a (*any_caller*). Вариант трассы, приводящей к ошибке, показан на рис. 9.

1, 2 – политика входного значения *i2*(*stat*) `x: manager(x)` `x: t_expire` попадает в глобальную переменную *col_status* в сеансе *bob*. 3, 4 – политика глобальной переменной *col_status* передается во внутреннюю переменную *v_status* функции *f_is_accepted* в сеансе *alex*. 5 – в результате проверки условия *if* политика переменной *v_status* переходит в политику *pc_label* сеанса *alex*. 6 – политика *pc_label* передается в возвращаемое значение функции *f_is_accepted*, которая обладает стационарной политикой a. Таким образом, на последнем шаге трассы нарушается рассмотренное уже ранее условие, заданное правилом (C-EXT-RET) абстрактной семантики.

В данном случае, скорее всего, администратором БД была допущена ошибка при настройке правил разграничения доступа. Функция *f_is_accepted* не должна вызываться явно пользователями системы, или доступ к ней должен быть ограничен ролью *manager*. Корректировка объектных привилегий может выглядеть как:

```
revoke execute on f_is_accepted from PUBLIC;
grant execute on f_is_accepted to manager.
```

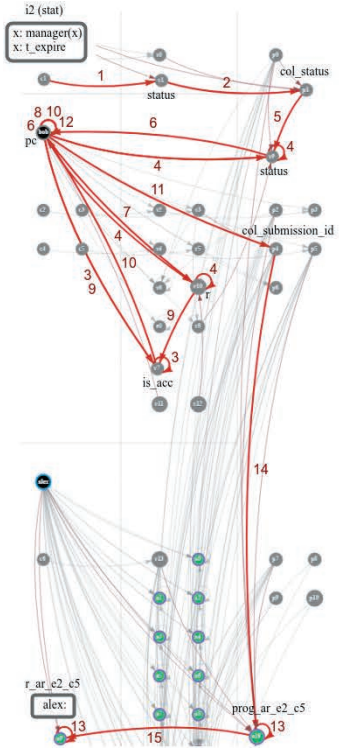


bob	p_change_status_load	change_status_load
bob	p_change_status4	update SUBMISSIONS set STATUS = stat where SUBMISSION_ID = s_id
bob	p_change_status_exit	change_status_exit
alex	f_is_accepted_load	is_accepted_load
alex	f_is_accepted5	select STATUS into v_status, from SUBMISSIONS where SUBMISSION_ID = s_id
alex	f_is_accepted8	if v_status = 1
alex	f_is_accepted9	then return TRUE

Рис. 9. ParalocksInv is violated (f_is_accepted_9).

Для исправления спецификации необходимо в формулу начального состояния *Init* внести изменения, при которых при запуске процедуры *f_is_accepted* в сеансе пользователя *s* во множество открытых блокировок сеанса *SLocks* добавлялись бы блокировки: *manager*(*s*) и *guest*(*s*):

$Init \triangleq \ldots \wedge SLocks =$
$[s \in S \mapsto$
$[e1 \in E0 \mapsto \{\}]$
$@@[e2 \in E1 \mapsto$
CASE . . .
if *f_is_accepted* is called
manager(*s*) and *guest*(*s*) are added
into the initial set of open locks
$\square \wedge Sessions[s][\text{“StateRegs”}][1][\text{“pc”}][1]$
$= \text{“ f\_is\_accepted”}$
$\wedge \vee e2 = \text{“manager”}$
$\vee e2 = \text{“guest”} \rightarrow \{s\}$
$\square \ldots$



allen	p_change_status_load	change_status_load
allen	p_change_status4	update SUBMISSIONS set STATUS = stat where SUBMISSION_ID = s_id
allen	p_change_status_exit	change_status_exit
bob	p_allocate_load	allocate_load
bob	f_is_accepted_call	is_accepted_call
bob	f_is_accepted5	select STATUS into v_status from SUBMISSIONS where SUBMISSION_ID = s_id
bob	f_is_accepted8	if v_status = 1
bob	f_is_accepted9	then return TRUE
bob	f_is_accepted11	endif
bob	f_is_accepted_exit	is_accepted_exit
bob	p_allocate7r	is_accepted_exit
bob	p_allocate_8	if v_is_acc
bob	p_allocate_10	then select paper_id into v_p_id from SUBMISSIONS where submission_id = s_id
bob	p_allocate_13	insert into ALLOCATIONS (ALLOCATION_ID, SUBMISSION_ID, SECTION_ID, ALLOCATION_DATE) values (id, s_id, sec_id, alloc_date)
bob	p_allocate16	endif
bob	p_allocate_exit	allocate_exit
alex	f_get_section_program_5	select paper_typ(PAPER_ID, TITLE, ABSTRACT, TEXT, 'UNKNOWN_AUTH' bulk collect into v_program from PAPERS from PAPERS where PAPER_ID IN (select PAPER_ID from ALLOCATIONS a join SUBMISSIONS s on a.submission_id = s.submission_id where a.SECTION_ID = s_id)
alex	f_get_section_program_9	return v_submissions

Рис. 10. ParalocksInv is violated (f_get_section_program_9).

При этом политику возвращаемого значения функции в *ParametersFS.tla* также следует изменить:

$$
\begin{aligned}
&f\_ia\_r(x) \triangleq \\
&\quad [loc \mapsto \text{“mem”}, \\
&\quad offs \mapsto 2, \\
&\quad policy \mapsto \text{replace } any\_caller(x) \text{ with} \\
&\qquad \{\langle x, \langle [t\_expire \mapsto \{NONE\}], \\
&\qquad\quad [guest \mapsto \{NONE\}, \\
&\qquad\quad reviewer \mapsto \{NONE\}, \\
&\qquad\quad manager \mapsto \{x\}, \\
&\qquad\quad organizer \mapsto \{NONE\}]\rangle\rangle\}, \\
&\quad name \mapsto \text{“ f\_ia\_r”}]
\end{aligned}
$$

ШАГ 4. Нарушение ParalocksInv

Четвертая попытка проигрывания модели также приводит к нарушению инварианта *ParalocksInv*. При выполнении оператора *f_get_section_program_9* (рис. 10) вновь нарушается условие, заданное правилом (C-EXT-RET).

Попробуем кратко описать проблемную траекторию распространения конфиденциальных данных. 1, 2 – политика входного значения $i2(stat)$ – `x: manager(x)` `x: t_expire` попадает в глобальную переменную *col_status* в сеансе *allen*. 3,4,5 – в сеансе *bob* загружается процедура *p_allocate* внутри нее происходит вызов функции *f_is_accepted*, и политика глобальной переменной *col_status* передается в локальную переменную *v_status*. 8 – политика переменной *v_status* добавляется к политике *pc_label* сеанса *bob*. 7, 8, 9 – политика `x: manager(x)` `x: t_expire` из элемента *pc_label* сначала передается в выходное значение функции *f_is_accepted*, а затем в локальную переменную *v_is_acc* процедуры *p_allocate*, при выходе из конструкции *if* указанная выше политика удаляется из общей политики элемента *pc_label*. 10 – политика `x: manager(x)` `x: t_expire` из переменной *v_is_acc* вновь попадает в *pc_label*. 11 – политика элемента *pc_label* сеанса *bob* передается в глобальную переменную *col_submission_id*. 13, 14 – в сеансе *alex* вызывается функция *f_get_section_program*, и в ее локальную переменную *v_program* (представляет собой массив объектов, состоящих из 5 полей) передается политика из глобальной переменной *col_submission_id*. 15 – политика одного из элементов массива *v_program* передается в один из элементов выходного массива функции *f_get_section_program*, который в соответствии с текущими настройками глобальной политики управления

доступом обладает стационарной политикой a. В данном случае аналитик легко заметит, что среди возвращаемых функцией значений имеются наименования статей. То есть любой пользователь независимо от выполнения заданного в описании требований по безопасности условия истечения временного интервала может получить список докладов, обладающих статусом “допущено”. Очевидно, выявленный запрещенный информационный поток не является ложным срабатыванием. Для устранения проблемы целесообразно применить ту же стратегию (деклассификация *WHEN*), что и на шаге 2 – внедрить в код процедуры проверку:

```
create or replace function f_get_section_program
...
is
begin
  if is time expire() or is manager()
    then ... else return null;
end p_get_section_program;
```

и исправить модель, добавив во множество открытых блокировок сеанса, в котором вызывается функция *f_get_section_program*, необходимые элементы:

$f\_get\_section\_program\_load(id) \triangleq$
...
$\wedge \vee openLock(id, \{[t\_expire \mapsto ALL]\})$
$\vee openLock(id, \{[manager \mapsto id]\})$
...

Нарушение инварианта не всегда свидетельствует о наличии действительно опасного информационного потока. Предложенный анализ, как уже отмечалось, не является абсолютно точным. Однако предполагается, что любую проблемную траекторию можно легко исследовать с использованием графической утилиты *plifparser*. Например, если в последнем случае (рис. 10) запрос *SELECT* вместо полных сведений о докладе возвращал бы только идентификатор *paper_id*, который по мнению аналитика невозможно ассоциировать с конкретной работой, или общее количество докладов в заданной секции, то возвращаемое значение можно было бы деклассифицировать без дополнительных проверок (деклассификация *WHAT*):

---

select(*id*, ...,
Replace the value calculated using *LUB*
LUB4Seq
(⟨*VPol*[“*col_papers_paper_id*”].*policy*,
VPol[“*col_allocations_submission_id*”].*policy*,
VPol[“*col_allocations_section_id*”].*policy*,
VPol[“*col_submissions_paper_id*”].*policy*,
VPol[“*col_submissions_submission_id*”].*policy*,

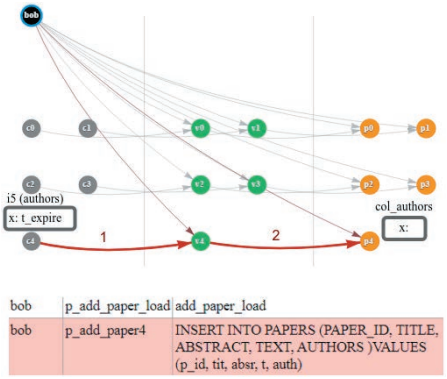


Рис. 11. Action property CompInv is violated (p_add_paper_4).

load(*id*, *f_gsp_p_s_id*(*id*))))
with *MIN* policy (WHAT declassification)
min, ...)

ШАГ 5. Нарушение свойства CompInv

После исправления модели на шаге 4 нарушений основного инварианта безопасности более не происходит. Однако включение проверки свойства перехода *CompInv* приводит к ошибке при выполнении оператора *f_is_accepted_9* (рис. 11). Стратегия исправления ошибки проста (см. Раздел 3) – обновление начальной политики соответствующей глобальной переменной:

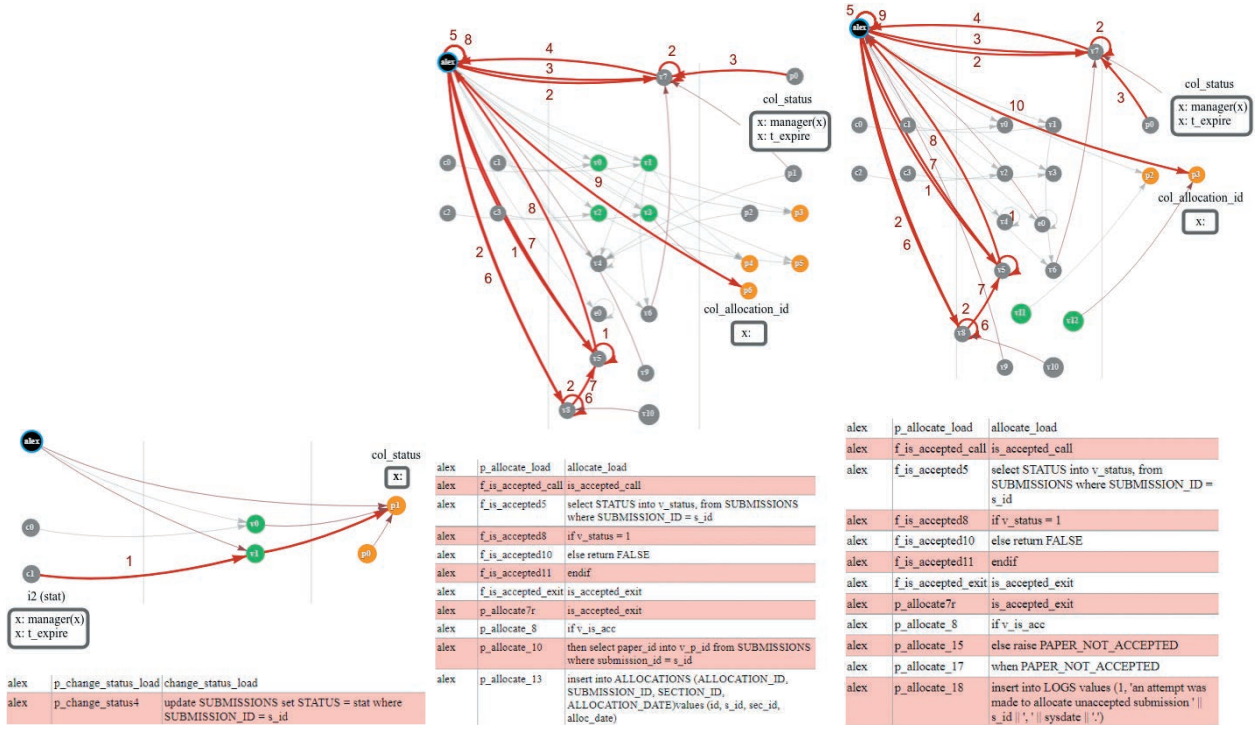
$Init \triangleq ...$
$\wedge VPol =$
[... *col_papers_authors* $\mapsto$
[*ext* $\mapsto$ 0,
policy $\mapsto$ min
{⟨*u*1, ⟨[*t_expire* $\mapsto$ {}],
[*guest* $\mapsto$ {*NONE*},
reviewer $\mapsto$ {*NONE*},
manager $\mapsto$ {*NONE*},
organizer $\mapsto$ {*NONE*}]⟩⟩},
name $\mapsto$ “`col papers authors`”], ...]

Исправление кода *PL/SQL* или глобальной политики управления доступом не требуется [4].

ШАГИ 6 – 8. Нарушение свойства CompInv

Последующие три прогона модели приводят к нарушению свойства перехода *CompInv* в операторах: *p_change_status_4*, *p_allocate_13*, *p_allocate_14* (рис. 15). Исправление модели осуществляется также как и на Шаге 5.

После исправления ошибки на восьмом шаге проигрывание модели завершается успешно.



(а) p_change_status 4 (б) p_allocate_13 (в) p_allocate_18

Рис. 12. Action property CompInv is violated.

Контроль распространения выходных значений критичных процедур и функций в прикладном программном обеспечении предполагается осуществлять с использованием стандартного анализа помеченных данных (Taint Tracking), например в среде *CodeQL* [16]. В данной работе этот этап подробно не рассматривается.

ЗАКЛЮЧЕНИЕ

В данной работе детализирована процедура выявления запрещенных информационных потоков в программных блоках *PL/SQL*. Возможно несколько избыточный иллюстративный материал, подготовленный с использованием утилиты *plifparser*, на наш взгляд имеет существенное значение для доказательства практической осуществимости предложенного в [4] подхода. В условиях некоторой потери точности анализа, за счет неизбежного для формальной верификации абстрагирования, возможность графической интерпретации проблемных трасс вычислений значительно упрощает задачу выявления “ложных” срабатываний.

Наряду с иными исследованиями логической семантики политик *Paralocks* [17], в данной работе раскрываются некоторые особенности возможной реализации языка в спецификациях *TLA+*. Существенным в этой части является наличие формальных доказательств свойств алгебраической решетки, построенной на заданном множестве политик безопасности.

В дальнейшем требуется провести отдельные исследования в отношении последнего этапа предложенной процедуры контроля информационных потоков – анализа распространения выходных значений критичных процедур и функций в прикладном программном обеспечении.

Кроме того, в работах, посвященных КИП, достаточно широко охвачена проблема *деклассификации* данных, но практически не уделяется внимание обратной процедуре – *классификации* данных. Уровень конфиденциальности используемых программой данных может не только понижаться, но и повышаться в процессе их обработки.

СПИСОК ЛИТЕРАТУРЫ

1. *Latham D.C.* Department of defense trusted computer system evaluation criteria // Department of Defense, 1986. T. 198.
2. *Infrastructure P.K., Profile T.P.* Common criteria for information technology security evaluation // National Security Agency, 2002.
3. *Девянин П.Н., Леонова М.А.* Применение подтипов и тотальных функций формального метода Event-B для описания и верификации МРОСЛ ДП-моде-

ли // Программная инженерия. 2020. Т. 11. № 4. С. 230–241.

4. *Тимаков А.А.* Контроль информационных потоков в программных блоках баз данных на основе формальной верификации // Программирование. 2022. № 4. С. 27–49
5. *Denning E. Dorothy* A lattice model of secure information flow // Communications of the ACM. 1976. № 5. P. 236–243.
6. *Шайтура С.В., Питкевич П.Н.* Методы резервирования данных для критически важных информационных систем предприятия // Российский технологический журнал. 2022. Т. 10 (1). С. 28–34.
7. *Timakov A.* PLIF. 2021. GitHub. https:// github.com/timimin/plif.
8. *Konnov I., Kukovec J., Tran T.-H.* TLA+ model checking made symbolic // Proceedings of the ACM on Programming Languages. 2019. V. 3. OOPSLA. P. 1–30.
9. *Broberg Niklas, Sands David* Paralocks: Role-based information flow control and beyond // Conference Record of the Annual ACM Symposium on Principles of Programming Languages. 2010. P. 431–444.
10. *Broberg Niklas, Sands David* Flow locks: Towards a core calculus for dynamic flow policies // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2006. No March. P. 180–196.
11. *Broberg N.* Thesis for the Degree of Doctor of Engineering Practical, Flexible Programming with Information Flow Control. 2011.
12. *Hedin Daniel, Sabelfeld Andrei* A Perspective on Information-Flow Control // Software safety and securit. 2012. P. 319–347.
13. *Methni A., Lemerre M., Hedia B.B., Barkaoui K., Haddad S.* An Approach for Verifying Concurrent C Programs // 8th Junior Researcher Workshop on Real-Time Computing. 2014. P. 33–36.
14. *Fernandes A.* tlaplus-graph-explorer. 2021. GitHub. https:// github.com/afonsonf/tlaplus-graph-explorer.
15. *Hedin Daniel, Sabelfeld Andrei* A Perspective on Information-Flow Control // Software safety and security. 2012. P. 319–347.
16. *Kristensen E.* CodeQL. 2022. GitHub. https:// github.com/github/codeql.
17. *Bart V. Delfit, Broberg Niklas, Sands David* A Datalog semantics for Paralocks // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2013. P. 305–320.
18. *Harrison M.A ., Ruzzo W.L., Ullman J.D.* Protection in operating systems // Communications of the ACM. 1976. V. 19. № 8. P. 461–471.

ПРОГРАММИРОВАНИЕ, 2023, № 4, с. 58–74

АНАЛИЗ ДАННЫХ

УДК 004.622

ПРИМЕНЕНИЕ ИМИТАЦИОННОГО КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ К ЗАДАЧЕ ОБЕЗЛИЧИВАНИЯ ПЕРСОНАЛЬНЫХ ДАННЫХ. ОЦЕНКА СОСТОЯНИЯ И ОСНОВНЫЕ ПОЛОЖЕНИЯ

© 2023 г. А. В. Борисов[a],* (ORCID: 0000-0002-3124-2147),
А. В. Босов[a], (ORCID: 0000-0001-7163-341X),**
А. В. Иванов[a],* (ORCID: 0000-0001-7811-7645)**

[a]Федеральный исследовательский центр "Информатика и управление" РАН
119333, Москва, ул. Вавилова, д. 44, кор. 2, Россия

**E-mail: aborisov@ipiran.ru*

***E-mail: avbosov@ipiran.ru*

****E-mail: aivanov@ipiran.ru*

Поступила в редакцию 20.01.2023 г.
После доработки 25.02.2023 г.
Принята к публикации 02.03.2023 г.

В статье представлена первая часть исследования по проблеме автоматизированной обработки персональных данных с целью их обезличивания и анализа. Эта часть носит обзорный характер и ставит целью анализ состояния исследований в данной области и систематизацию имеющихся результатов. Представлены результаты анализа широкого круга вопросов обезличивания, сформировавшие системное понимание состояния исследований и обосновавшие выбор направления для дальнейшего изучения. Вначале сформулированы определения основных терминов и понятий, используемых в связи с обезличиванием персональных данных, в т.ч. в увязке с законодательством РФ. Направления исследований сгруппированы по четырем разделам: методы обезличивания, проблемы реализации, приложения обработки обезличенных данных, вопросы деобезличивания. По каждой из групп методов обезличивания – рандомизации, группировке, распределению данных и контролю приложений – даны описания основных алгоритмов, проанализированы их достоинства и недостатки. Проблемы реализации затрагивают такие понятия как полезность обезличенных данных, ограничения применимости универсальных алгоритмов и надежность в отношении сохранения анонимности субъектов персональных данных. В числе прикладных решений, сформировавших востребованность обработки обезличенных данных, обсуждаются медицинские, биологические, генетические исследования и охрана правопорядка. В заключительной части упоминаются наиболее резонансные факты деобезличивания и дается небольшой обзор прессы.

DOI: 10.31857/S0132347423040040, **EDN:** RDBXOS

1. МЕТОДЫ И АЛГОРИТМЫ ОБЕЗЛИЧИВАНИЯ

1.1. Основные термины и определения

К настоящему времени для исследований, проводимых научным сообществом в области обезличивания персональных данных (ПД), не сформирована единая общепринятая терминология. Во многом это объясняется различными правовыми режимами ПД в разных государствах. Основные понятия, использованные в данной работе, опираются на законодательную базу РФ и ориентированы на решение информационных и алгоритмических вопросов, а не на правовое регулирование. Принципиальное понимание "персональных данных" дается Федеральным законом "О персональных данных" от 27.07.2006 № 152-ФЗ. Для технического описания методов и алгоритмов далее используются следующие понятия:

• необезличенные ПД – множество ПД в исходном виде,

• персональная идентификационная информация (персональные идентификаторы) – любые цифровые идентификаторы человека (субъекта ПД), указывающие на него напрямую,

• обезличенные ПД – множество ПД, полученное в результате обработки необезличенных ПД в соответствии с некоторым алгоритмом обезличивания с целью препятствования деобезличиванию ПД,

• деобезличивание ПД – процесс обработки обезличенных ПД с целью (возможно, частично-

го) восстановления соответствующего набора необезличенных ПД и/или установления любого соответствия ПД и субъекта ПД (нарушения анонимности),

• чувствительная информация – ПД, которые не являются персональными идентификаторами, но представляют угрозу нарушения анонимности,

• нечувствительная информация – ПД, которые не представляют сколь-либо существенной угрозы нарушения анонимности и не могут быть использованы для установления любого соответствия ПД и субъекта ПД,

• несущественная информация – ПД, которые не содержат семантически значимых данных в отношении заявленной цели обезличивания,

• квазиидентификаторы – это любые элементы ПД, которые позволяют исключить из имеющегося набора обезличенных ПД данные, не относящиеся к конкретному субъекту ПД,

• псевдоданные (синтетические ПД) – это данные, полученные в результате компьютерного моделирования, реализующего модель, воспроизводящую статистические характеристики соответствующего набора необезличенных данных,

• уровень анонимности обезличенных ПД – оценка сложности проведения деобезличивания субъектов ПД,

• уровень полезности обезличенных ПД – свойство множества обезличенных ПД, заключающееся в возможности решения прикладных задач с использованием обезличенных ПД.

1.2. Направления исследований в области обезличивания ПД

Вопросы и задачи, затрагивающие тематику ПД, уже довольно длительное время привлекают исследовательский интерес научного сообщества. Пристальное внимание к области последние двадцать лет наблюдается в РФ: периодические сообщения об очередных утечках ПД привлекают внимание общества в целом и инспирируют весьма ответственные и актуальные правовые шаги госорганов. Но наше внимание направлено не на социальные и правовые аспекты, а на вопросы, составляющие исследовательский интерес. И здесь надо констатировать, что внимание российского научного сообщества к области минимально. Вместе с тем проблематика алгоритмов обезличивания и деобезличивания очень богата и предоставляет хорошие возможности для приложения исследовательских усилий.

К настоящему времени опубликовано огромное число работ, с той или иной степенью математической строгости представляющих соответствующие алгоритмы. Представленный далее материал в целом написан в контексте систематического обзора [1] с современными дополнениями [2–6]. Особо надо упомянуть отдельный класс задач, связанных с обезличиванием фото- и видеоматериалов [7–10], специфика которых очевидно связана с медийностью элементов ПД.

Большинство методов обезличивания представляют собой преобразования данных с целью обеспечения требуемого уровня анонимности (далее будет обсуждаться формальное определение этого понятия, пока ограничимся его интуитивно понятным смыслом). Такие методы основаны на уменьшении детализации данных, приводящей к потере эффективности последующих операций поиска и обработки информации, если они направлены на деобезличивание, и потенциально не препятствуют обработке данных, не носящей идентифицирующий характер, например агрегации, статистике и прочим обобщениям. Уменьшение детализации – это плата за обеспечение анонимности. Основная задача разработчиков алгоритмов обезличивания заключается в нахождении компромисса между требуемым уровнем приватности и остаточной полезностью обезличенных данных. Большинство результатов, имеющих адекватное математическое описание, посвящаются следующим вопросам:

• методы обезличивания ПД,

• проблемы реализации,

• целевые приложения обработки обезличенных ПД,

• возможности и варианты деобезличивания.

Наиболее распространенными подходами к обезличиванию являются следующие.

1. Рандомизация: преобразование данных путем добавления в них шума для маскировки значений записей (чаще всего числовых) [11, 12]. Считается, что уровень шума должен быть настолько велик, чтобы сделать невозможным восстановление исходных значений записей по имеющимся зашумленным данным.

2. K-анонимность и L-разнообразие: метод K-анонимности разработан как возможное средство борьбы с косвенным деобезличиванием по информации из открытых баз данных, которое оказывается возможным, если некоторые комбинации элементов ПД позволяют точно идентифицировать всю запись. Метод K-анонимности предполагает уменьшение детализации данных за счет группировки, достигаемой различными приемами: удалением информации, обобщением, маскингом и пр. Уменьшение детализации должно обеспечивать такое преобразование, чтобы каждой обезличенной записи соответствовало не менее K исходных записей [13–16]. Модель L-разнообразия разработана, чтобы нейтрализовать имеющийся у метода K-анонимности недостаток – возможность распознавания субъекта с точностью до группы размером K не обеспечивает защиту чувствительной информации до такого же

уровня, в особенности, когда чувствительная информация неоднородна внутри данной группы. Для обеспечения этого свойства предлагается концепция внутригруппового разнообразия чувствительной информации за счет специальной схемы обезличивания [17, 18].

3. Распределение данных: выполняется разбиение данных на несколько наборов с последующей распределенной работой с этими наборами. Разбиение может быть как горизонтальным (наборы данных разбиваются на некоторые подмножества), так и вертикальным (наборы данных разбиваются на некоторые подмножества атрибутов) [19–21]. Подобное разбиение обеспечивает приложениям доступ только к разделам данных, необходимым этим приложениям для выполнения целевых функций, без возможности обработки всего массива ПД.

4. Контроль работы приложений: нередкой является ситуация, когда анонимность нарушается из-за результатов работы приложений. Это выражается в нахождении новых правил ассоциации в процессе работы с данными, возможности классификации целей поиска при обработке поисковых запросов и пр. Данные обстоятельства вынуждают выполнять модификацию либо данных, либо приложений, работающих с данными. Примеры подобных исследований связаны с сокрытием ассоциативных правил [22], снижением чувствительности классификаторов [23] и контролем выполняемых запросов [24].

1.3. Методы и их свойства

1.3.1. Рандомизация. Традиционное применение находит при проведении анонимных опросов [25, 26], а также распространена на задачу безопасного поиска данных [11]. Метод заключается в следующем. Пусть набор значений одного элемента ПД представляет собой реализацию числовой выборки $X = \{x_1,\ldots,x_N\}$. Обезличивание заключается в добавлении к X случайного шума $Y = \{y_1,\ldots,y_N\}$ – набора независимых одинаково распределенных случайных величин с публично известным распределением $f_Y(y)$. В результате обезличивания для последующего статистического анализа доступна реализация выборки $Z = \{x_1 + y_1,\ldots,x_N + y_N\}$. Соответственно, задача обработки обезличенных данных заключается в восстановлении неизвестного распределения $f_X(y)$ по зашумленным наблюдениям Z. Для решения этой задачи используются различные методы [11, 12], которые можно разделить на эмпирические, методы непараметрического оценивания (построение ядерных оценок плотности распределения), методы параметрического оценивания (метод максимального правдоподобия, его реализация в виде EM-алгоритма). Задача значительно осложняется в случае необходимости восстановления совместного распределения нескольких полей [27], т.к. для достижения постоянной относительной точности восстановления многомерной плотности с ростом размерности требуется экспоненциальный рост длины выборки.

Очевидным преимуществом метода рандомизации является его вычислительная экономичность, а существенным недостатком то, что зашумление малоэффективно для маскировки выбросов [28]. Свойства этого метода позволяют довольно эффективно использовать его в разных аналитических задачах. Так, в [11, 29, 30] они применялись для построения классификаторов, в [31, 32] – для построения безопасных ассоциативных правил. Подобная техника рандомизации также применялась при создании безопасных алгоритмов OLAP [33] и SVD-коллаборативной фильтрации в рекомендательных системах [34].

Для определения уровня анонимности при рандомизации используются различные числовые характеристики. В качестве простейшего показателя [11] в случае аддитивного шума с непрерывным распределением используются пара “длина доверительного интервала – уровень доверительной вероятности”. Например, если после статистических выводов оказалось, что оцениваемый параметр с вероятностью 0.8 принадлежит интервалу (1, 10), то в качестве этого показателя выступает пара (9, 0.8). Такой подход к анонимности имеет серьезный недостаток, состоящий в том, что он не учитывает собственное распределение значений поля. В некоторых случаях неудачный выбор распределения шума может привести даже не к повышению, а к снижению уровня анонимности.

В качестве альтернативных показателей уровня анонимности [12] могут выступать следующие характеристики:

• безусловная дифференциальная энтропия $h(B)$ обезличенных данных B с плотностью распределения вероятностей $f_B(b)$

$$h(B) = -\int_{\Omega_B} f_B(b)\log_2 f_B(b)\,db,$$

• величина $\Pi(B) = 2^{h(B)}$, имеющая смысл длины носителя равномерного распределения с тем же значением $h(B)$ (энтропия $h(B)$ случайной величины B, имеющей произвольную плотность распределения, совпадает с энтропией случайной величины с равномерным распределением на отрезке длиной $\Pi(B)$),

• условная дифференциальная энтропия $h(A|B)$ исходных данных A относительно обезли-

ченных данных B, и соответствующий показатель $\Pi(A|B) = 2^{h(A|B)}$,

$$h(A|B) = -\int_{\Omega_{A,B}} f_{A,B}(a,b)\log_2 f_{A|B}(a|b)\,dadb,$$

• условная потеря анонимности A при условии B: $\mathcal{P}(A|B) = 1 - \dfrac{\Pi(A \mid B)}{\Pi(A)}$,

• совместная информация A и B: $I(A, B) = h(A) - h(A|B)$.

Помимо исследований, сфокусированных на характеризации уровня анонимности, также рассматривалась задача поиска паллиатива между уровнем анонимности и потерей информации при обезличивании [35].

Кроме упомянутой неспособности скрыть аномальные значения в необезличенных ПД, рандомизация имеет и другие слабые стороны, которые можно использовать для деобезличивания. Основным подходом здесь является корреляционный анализ, который позволяет найти "линейную" зависимость между элементами обезличенных данных даже при добавлении аддитивного шума и уменьшить тем самым "размерность" обезличенных ПД, что и составляет угрозу нарушения анонимности. Наиболее распространенными являются алгоритмы главных компонент и спектральная фильтрация [36–38]. Как только корреляционная структура обезличенных данных прикидочно оценена (по выборке большого объема), можно пытаться очистить имеющиеся данные от шума для того, чтобы добиться деобезличивания. Другой вид атак на анонимность заключается в использовании метода максимального правдоподобия для оценки потенциальных шумов в данной выборке обезличенных данных по известному распределению шума.

В связи с рандомизацией нельзя не упомянуть использование вместо аддитивного шума мультипликативного. Техника здесь в основном базируется на аппарате многомерных проекций для сокращения размерности данных [39]. Он приближенно сохраняет "расстояние" между отдельными записями ПД, и поэтому обезличенные данные могут быть использованы для последующего статистического анализа. Этот подход использован в [40, 41] для кластерного анализа обезличенных ПД. Эта же техника применялась в [42] для решения задачи классификации, в [43] – для обеспечения безопасного распределенного анализа данных, в [44] – для обезличивания результатов переписи населения. Еще один вариант применения данного подхода связан с использованием преобразования Фурье, сохраняющего расстояние [45]. Как и использование аддитивных шумов, использование мультипликативных шумов не может рассматриваться как панацея при обезличивании. Опасности имеются в случае, если атакующий обладает некоторым априорным знанием о данных [46].

Наконец, аддитивное или мультипликативное зашумление – не единственные варианты случайного возмущения данных. Еще один возможный способ – перемешивание с целью достижения обезличивания [47]. Явным преимуществом перемешивания является то, что сохраняется разброс выборки (т.е. пара "минимум-максимум"), да и сама выборка тоже. В этом же и самый серьезный недостаток – актуальные, не измененные цифровые ПД могут оказаться квазиидентификаторами и стать источником для деобезличивания всего набора ПД или его части. Кроме того, разные перемешивания разных элементов ПД нарушают имеющиеся зависимости между ними, что уменьшает информативность обезличенных ПД (например, из-за нарушения корреляции между элементами ПД при их независимом перемешивании).

1.3.2. Группировки (K-анонимность, L-разнообразие, t-близость). Добавление шума – эффективный метод обезличивания, который может применяться в процессе сбора данных, т.к. отдельные элементы ПД получаются и зашумляются последовательно, независимо друг от друга. Но в этом состоит и слабость данного метода, поскольку выбросы в исходных данных можно определить только после окончания их сбора. Кроме того, аномальные измерения сложно замаскировать шумами. Таким образом, при обезличивании нужно обеспечивать одинаковую степень приватности всем элементам ПД. Другой слабой чертой рандомизации является наличие обширных открытых данных, с использованием которых оказывается возможным идентифицировать субъекта, к которому данная запись относится. В [28] показано, что совместная обработка открытых данных и многомерных обезличенных данных способна серьезным образом скомпрометировать последние. И особо уязвимыми при этом являются значения элементов ПД (выбросы). Именно эти обстоятельства подтолкнули исследователей к созданию методов обезличивания, основанных на группировке исходных ПД.

Группировки (K-анонимность). Во многих приложениях обезличивание проводилось путем простого исключения из записей ключевых идентификаторов типа имен, номеров документов и пр. Однако некоторые другие атрибуты (выше они определены как квазиидентификаторы) могут быть использованы для точной идентификации обезличенных записей. К таким атрибутам относятся, например, набор "возраст/дата рождения-индекс-пол". Если эти данные одновременно присутствуют в неискаженном виде в обез-

личенных и открытых данных, то по ним легко идентифицировать субъекта ПД. Идея метода *K*-анонимности заключается в снижении детализации квазиидентификаторов с использованием операций обобщения/группировки и исключения. Обобщение означает замену конкретного значения некоторым интервалом, например, замена даты рождения годом или интервалом лет. Исключение подразумевает отсутствие значения в обезличенных данных, например, исключение номеров банковских счетов. Очевидно, что снижение детализации одновременно с обеспечением высокого уровня анонимности, снижает уровень полезности обезличенных данных.

Концепция *K*-анонимности предполагает, что каждый кортеж квазиидентификаторов в наборе обезличенных данных неотличим от кортежей как минимум еще *K* субъектов ПД. Первый алгоритм *K*-анонимности был предложен в [48]. Он основан на некоторой иерархии обобщения предметной области квазиидентификаторов. Однако с практической реализацией метода, несмотря на кажущуюся простоту идеи, возникают проблемы, связанные с его ресурсозатратностью. В [16] представлено утверждение о том, что задача оптимальной *K*-анонимности (т.е. обеспечения набору данных *K*-анонимности с минимальной потерей информации) является NP-сложной, т.е. характеризуется полиномиальным ростом числа операций. Поэтому были разработаны достаточно эффективные с вычислительной точки зрения эвристические алгоритмы. Так, в [49] предложен алгоритм *K*-Optimize, использующий идею упорядочения полей квазиидентификатора. Алгоритм Incognito [15] разработан для *K*-минимальной группировки иерархии обобщения предметной области квазиидентификаторов путем ее агрегации “снизу вверх”. Еще два интересных метода нисходящей детализации и восходящего обобщения реализации *K*-анонимности были предложены в [50] и [51]. В [50] предлагается эвристический алгоритм обработки иерархии “сверху вниз”, в процессе которого некоторые поля детализируются, что увеличивает информацию, но снижает уровень анонимности, а в [51] предложен смежный метод “обобщения снизу вверх”.

Следует заметить, что обобщение и исключение не являются единственными возможными преобразованиями данных, используемых концепцией *K*-анонимности. Например, в [52] предлагается подход, реализующий агрегацию записей в кластеры с использованием средней величины по каждому полю кластера. Аналогичная кластеризация, но с применением факторного анализа представлена в [53]. Очень важна для дальнейшего исследования работа [54], в которой кластеризация дополнена техникой генерации псевдоданных из кластеризованных групп *K*-записей. Такой подход оказался эффективным для решения задач классификации. Кроме того, использование псевдоданных представляет собой дополнительный уровень защиты, т.к. сложно организовать атаки на синтетические данные.

Поскольку процесс обеспечения набору данных *K*-анонимности по своей сути является многошаговым итеративным поиском в пространстве многомерных решений, стандартные эвристические методы поиска также могут быть использованы достаточно эффективно. Так, в [55] применен алгоритм имитации отжига, а в [56] – генетический алгоритм.

Анализ уровня анонимности, обеспечиваемой концепцией *K*-анонимности, был проведен в [57]. Эффективность защиты была проверена с помощью сценария атаки, когда атакующему была доступна некоторая дополнительная информация, например часть оригинальных данных. Для этого сценария предложена методика, позволяющая оценить среднее число записей, которые возможно идентифицировать. Эта методика может быть полезна в тех ситуациях, когда необходимо ответить на вопрос, нужно или нет использовать *K*-анонимность в той или иной конкретной задаче.

Не все субъекты одинаково относятся к уровню анонимности. Это означает, что параметр *K*-анонимности может варьироваться от записи к записи. Алгоритм с переменным параметром *K* был предложен в [58]. Он основан на создании групп различного объема, для которых гарантировано, что размер группы не меньше, чем максимальный параметр *K* в данной группе. А затем, для каждой группы генерируются синтетические данные с характерным распределением.

Другой любопытный подход к обезличиванию предложен в [59], когда субъекту предлагается указать уровень защиты чувствительной информации, например, запретить выдавать определенные атрибуты ПД. Соответствующая техника предполагает, что субъект может указать узел в иерархии обобщения для определения своего уровня анонимности.

Отдельным классом приложений для методов на основе операций обобщения является обезличивание динамических данных, таких как потоковые. Очевидно, что как только блок данных обработан и передан в открытое использование, его нельзя преобразовать заново, например изменить уровень обобщения. В то же время следующий опубликованный блок данных может скомпрометировать предыдущие. Такой пример приведен в [60], где предлагается сделать невозможным создание связей между квазиидентификаторами, присутствующими в обоих блоках. Для этой задачи в [61] предлагается метод управляемого уменьшения степени детализации с целью сохранения уровня *K*-анонимности на всем объединении имеющихся обезличенных данных.

Еще одна группа специфических задач для методов обобщения образуется для атрибутов, имеющих текстовые, бинарные или строковые типы. В отличие от потоковых данных здесь нет особенностей, связанных с последовательным поступлением информации, но есть выраженный характер чувствительности и значительные размерности. В качестве примера можно упомянуть чеки супермаркетов и электронные резюме. Типовой объект для K-анонимизации – это числовые и/или категорийные данные. Чеки представляют собой совокупность числовых и строковых данных, обладают весьма высокой размерностью, при этом данные разрежены (лишь малое число значений в строковой составляющей отличны от 0). Некоторые методы обезличивания, базирующиеся именно на свойстве разреженности данных, предложены в [62]. Непосредственное применение алгоритмов K-анонимизации к строковым и тем более бинарным данным невозможно хотя бы по причине их переменной длины. В [63] предложен метод, кластеризующий исходные строковые данные с последующим моделированием синтетических данных, имеющих те же статистические характеристики, что и строки в отдельных кластерах.

Группировки (L-разнообразие, t-близость). Концепция K-анонимности выглядит интересной из-за простоты понимания и большого числа алгоритмов реализации. Тем не менее, ПД, удовлетворяющие условию K-анонимности, могут стать объектом атак, связанных с однородностью (когда значения всех атрибутов чувствительной информации в группе совпадают) и атак, связанных со знанием фоновых значений (когда можно использовать ассоциацию между одним/несколькими квазиидентификаторами для сужения множества возможных значений атрибутов чувствительной информации). Так, в [17] приведен пример, в котором фоновое знание о малой подверженности японцев сердечным приступам может быть использовано для сужения информационной неопределенности чувствительного атрибута “Заболевание”. Более детальное обсуждение влияния фоновых значений приведено в [64].

Можно сказать, что K-анонимность эффективно предотвращает деобезличивание конкретного субъекта ПД, но не всегда защищает отдельные значения атрибутов чувствительной информации. Концепция L-разнообразия гарантирует не только минимальный размер группы, равный K, но и обеспечивает внутригрупповое разнообразие чувствительных значений. Связанные с L-разнообразием понятия и определения предложены в [17]. Принципиальная идея этой концепции состоит в том, что при любом обобщении атрибутов нечувствительной информации остающиеся атрибуты чувствительной информации содержат не менее, чем L различных значений. Дополнительное требование L-разнообразия в случае многомерных атрибутов чувствительной информации делает задачу K-анонимизации особо сложной из-за проблемы “проклятия размерности” [14].

t-близость является дальнейшим развитием концепции L-разнообразия. Дело в том, что в модели, используемой концепцией L-разнообразия, все значения одного атрибута трактуются одинаково, независимо от их распределения. Тем самым косвенно предполагается, что реализации значений атрибута равновероятны. В реальности такая ситуация исключительна. Неравномерность распределения значений может затруднить построение обобщений, обладающих свойством L-разнообразия. Более того, знание глобальных распределений различных атрибутов может позволять делать выводы, снижающие уровень анонимности, особенно для атрибутов чувствительной информации. Кроме того, не все возможные значения атрибута обладают одинаковой “степенью чувствительности”. Например, логическое значение поля, соответствующее какому-либо заболеванию, более чувствительно, если оно имеет положительное значение, чем отрицательное. В [18] предложена концепция t-близости, согласно которой распределение значений чувствительного поля внутри любой группы анонимности не должно отличаться от глобального распределения более, чем на порог t. В качестве характеристики близости распределений часто используется метрика Вассерштейна.

1.3.3. Распределение данных. Целью большинства распределенных методов обезличивания ПД является обеспечение возможности выполнения статистических выводов по всему массиву данных без нарушения уровня анонимности отдельных записей. Идея заключается в разделении имеющегося массива данных таким образом, чтобы обеспечить корректность выполнения агрегированных статистических выводов, не имея одновременного доступа ко всей совокупности данных. Используемые для этого методы бывают как горизонтальными, так и вертикальными. В случае горизонтального разделения все множество записей разбивается на несколько подмассивов с одинаковым набором полей. При вертикальном разбиении отдельные массивы имеют различные атрибуты одного общего набора записей. Задача обеспечения анонимности распределенных данных по своей природе близка к задачам криптографии и протоколов конфиденциальных вычислений (secure multi-party computations, приемлемый обзор представлен в [65]). Подход к обезличиванию на основе распределенной обработки данных достаточно традиционен. Он базируется на вычислении функций по их аргументам, принадлежащим разным массивам, без допуска владельцев массивов к данным друг друга. Многие алгоритмы анализа и обработки данных могут быть пред-

ставлены в этом контексте как наборы повторений вычислительных примитивов: скалярных произведений, “безопасных” сумм, экстремумов и пр. Сами алгоритмы и обеспечиваемая ими анонимность зависят от уровня доверия между владельцами наборов данных. В литературе можно обнаружить такие характеристики как “получестные” и нечестные владельцы информации.

“Получестные” владельцы заинтересованы в получении информации из других массивов и пытаются извлечь пользу из полученной ими информации во время безопасных вычислений, но при этом не отклоняются от протокола. То есть целью их действий является не выполнение вычислений, а получение информации и чужих данных в процессе вычислений, например, путем большого числа запросов на выполнение разных вычислительных операций и последующего анализа полученных результатов. Во многих ситуациях это можно считать реалистичной моделью состязательного поведения. Нечестные владельцы информации могут отклоняться от протокола безопасных вычислений, поставлять на вход протокола специально синтезированные данные с целью получения информации о других информационных массивах.

Ключевым элементом выполнения безопасных вычислений является протокол передачи данных ([66, 67]), его называют “забывчивым”. Согласно ему отправитель предоставляет пару (x_0, x_1), а получатель – одно из битовых значений $\sigma \in \{0,1\}$. В результате получатель знает только x_σ, а отправитель не знает ничего. Для реализации этого протокола есть простые решения. В одном из них [66] отправитель генерирует два случайных открытых ключа, K_0, K_1, а получатель генерирует ключ K_σ. Получатель отправляет открытую часть своего ключа, шифруя ее одним из ключей, полученных от отправителя. Отправитель расшифровывает этот ключ, используя закрытые части своих ключей. При этом он получает два варианта ключа, один из которых действителен, а второй содержит мусорные данные. Отправитель шифрует x_0 с K_0, x_1 с K_1 и возвращает зашифрованные данные получателю. На этом этапе получатель может расшифровать только x_σ, т.к. он имеет только этот ключ дешифрования. Это “получестный” алгоритм, т.к. все промежуточные шаги требуют доверия между сторонами. Например, предполагается, что, когда получатель отправляет два ключа отправителю, он действительно знает ключ дешифрования только для одного из них. В случае нечестных владельцев информации необходимо убедиться, что отправитель выбирает открытые ключи согласно протоколу. Эффективный метод проверки этого факта приведен в [68]. В этой же работе протокол обобщен на случай нескольких владельцев информации. Забывчивый протокол является одним из простейших блоков в алгоритмах безопасных вычислений и может повторяться для вычисления заданной функции, поэтому важна его вычислительная эффективность. Численно эффективные методы для обоих видов владельцев информации также представлены в [68]. Более сложные проблемы в этой области включают вычисление различных вероятностных функций по аргументам с различными владельцами [69]. Алгоритмический аппарат этих действий развит как для случая двух владельцев информации, так и для произвольного числа владельцев [70]. Протокол забывчивой передачи данных может использоваться при выполнении некоторых вычислительных “примитивов”, связанных с вычислением расстояний между многомерными векторами, а также скалярных произведений [71]. Достаточно полный набор этих методов представлен в [72]. Многие из них основаны на отправке измененных или зашифрованных аргументов друг другу для вычисления функции с различными альтернативными значениями, последующей забывчивой передачей данных и получения в конце концов корректного конечного результата. В [72] в этом ключе рассмотрены задачи кластеризации, классификации, поиска ассоциативных правил, суммаризации и обобщения данных. Другой набор безопасных “примитивов” распределенного анализа данных представлен в [73]: он включает операции суммирования, объединения, пересечения, вычисления мощности множества и скалярного произведения. Все эти операции могут быть успешно применены при безопасной статистической обработке как вертикально, так и горизонтально распределенных наборов данных.

Горизонтальное разделение. В этом случае разные владельцы имеют разные наборы записей ПД с одинаковым (или очень близким) набором атрибутов. Большое число методов безопасного статистического анализа горизонтально распределенных данных основываются на общих методах, предложенных в [72, 73]. В других работах исследуются более тонкие методы: алгоритм ID3 построения дерева решений с использованием приближений наилучших атрибутов разбиения [74], наивный байесовский классификатор [75], классификатор Вапника–Червоненкиса с нелинейными ядрами [76]. “Предельный” случай рассмотрен в [77]: каждая запись, участвующая в совместной обработке, принадлежит отдельному владельцу. Кроме того, на случай горизонтально распределенных данных были обобщены такие алгоритмы анализа данных как поиск ассоциативных правил [78], кластеризация [79–81], коллаборативная фильтрация [82].

С обработкой горизонтально распределенных данных связана задача совместной индексации данных различных провайдеров. Сложность за-

ключается в том, что, с одной стороны, для ее выполнения нужна информационная кооперация, а с другой – провайдеры являются конкурентами друг друга. В [83] исследован алгоритм, выполняя который злоумышленник по результатам контекстных запросов к поисковой системе может восстановить интересующие его ПД. Решение возникшей проблемы предложено искать с помощью создания централизованного безопасного индекса в комплексе с распределенным механизмом контроля доступа. Предложенные меры позволяют обеспечить строгую приватность даже в случае сговора нескольких злоумышленников и в ситуации полного раскрытия индекса.

Вертикальное разделение. В этом случае остаются полезными многие вычислительные примитивы, например скалярное произведение, пересечение множеств и пр. Например, в [71] предложено использовать скалярное произведение для часто выполняемой операции пересчета элементов множества. Она же может быть выполнена с помощью безопасной процедуры нахождения пересечения множеств [73]. Один из методов поиска ассоциативных правил [84] использует скалярное произведение по вертикальному битовому столбцу, представляющему использование набора элементов в транзакциях, чтобы вычислить частоту соответствующих наборов элементов. Эта же вычислительно эффективная процедура может использоваться для многократного пересчета элементов множества.

Многие методы анализа данных распространены на случай их вертикальной распределенности: построение дерева решений [20], наивный байесовский классификатор [85], алгоритм Вапника–Червоненкиса [21], метод кластеризации k-средних [86]. Теоретическим вопросам реализации вычисления различных функций от вертикально распределенных аргументов с использованием методов криптографии посвящена работа [19].

Распределенные алгоритмы K-анонимности. Во многих практических задачах имеется необходимость обеспечить K-анонимность информационных массивов с различными собственниками. В [87] предложен K-анонимный протокол для вертикально распределенных данных с двумя владельцами. Общая идея заключается в определении для данных обоих владельцев некоторого общего квазиидентификатора, поскольку нужно атрибуты разных массивов каким-то образом синхронизировать для соответствующих субъектов ПД. Похожая идея предложена в [88], где собственникам предлагается предварительно договориться о правилах обобщения/объединения. В [89] задача K-анонимности рассматривалась для горизонтально распределенных данных. При этом исследован максимально распределенный вариант, т.е. в предположении, что каждая запись общего массива данных имеет своего собственника. Запись содержала квазиидентификаторы и атрибуты чувствительной информации, которые в итоге предполагается шифровать.

Проблема обеспечения K-анонимности важна и в других задачах безопасной обработки распределенных обезличенных данных. Так, известна задача сокрытия идентификационной информации при работе сервисов, связанных с определением местоположения [90–92]. Для такого контента K-анонимность идентификатора пользователя нужна даже в случае, когда информация о его местоположении становится доступной. В частности, это может быть в том случае, когда пользователь отправляет сообщение из некоторой точки данной локации/окрестности. Аналогичные проблемы могут возникать и в телекоммуникационных протоколах, в которых анонимность отправителя/получателя может быть не защищена. Здесь можно говорить о K-анонимности по отправителю K-анонимности по получателю.

1.3.4. Контроль работы приложений. Часто деобезличивание ПД оказывается возможным без прямого доступа к данным, а “благодаря” возможностям, предоставляемым приложениями, которые должны обеспечивать защищенный доступ к ПД. Все обсуждение здесь тесно связано с известной проблемой контроля разглашения [24] в статистических базах данных, хотя надо учитывать, что современные достижения в методах интеллектуального анализа данных предоставляют все более изощренные способы, чтобы злоумышленники могли делать выводы о ПД. В случае, когда ПД передаются в какой-либо форме оператором-собственником другому оператору, правила ассоциации могут представлять собой чувствительную информацию, например для целевого маркетинга, и эта информация должна быть защищена. К проблемам контроля раскрытия информации (в нашей терминологии – нарушению анонимности ПД) в приложениях относятся сокрытие ассоциативных правил, снижение эффективности классификации и обработки запросов.

Сокрытие ассоциативных правил. Ассоциативные правила являются весьма важными в сфере бизнеса при построении таргетированной рекламы. Для сокрытия ассоциативных правил используются два метода. Первый – искажение – описан в [93], где предложено параметр для транзакции заменять другим значением, а в битовом случае – инвертировать. Второй метод – блокировка [94] – оставляет параметр транзакции пустым. Оба метода имеют ряд побочных эффектов, влияющих на нечувствительные правила в данных. Многие из них могут быть искажены или утеряны вместе с чувствительными правилами

(атрибутами чувствительной информации) или могут появиться новые фантомные правила.

В [95] представлено формальное доказательство того факта, что применение метода искажения для сокрытия ассоциативных правил является NP-сложной задачей. В работе предлагается метод битового инвертирования $1 \to 0$ для снижения уровня поддержки чувствительных правил. Полезность предложенного подхода характеризуется числом нечувствительных правил, которые были нейтрализованы одновременно с чувствительным. Этот подход в [96] был усовершенствован. Детальное описание различных методов искажения данных для сокрытия ассоциативных правил можно найти в [93].

Основная идея метода блокировки представлена в [97]. Его главное качество заключается в том, что вместо искажения исходных данных он не дает доступа к ним. Некоторые интересные алгоритмы были представлены в [98], а затем расширены в [94] путем рассмотрения задачи реконструкции скрытых правил. Другой подход к построению алгоритмов сокрытия ассоциативных правил, ставящий целью снижение потерь нечувствительных правил/появления фантомных правил, представлен в [99]. В [100] обсуждалось влияние техники блокировки на определение злоумышленником атрибутов чувствительной информации – скрывались только те правила, которые могли раскрыть именно чувствительные атрибуты.

Снижение эффективности классификаторов. Одними из важнейших приложений, результаты которого могут понижать уровень анонимности, являются различные классификаторы. Основной сложностью противодействия здесь является такое преобразование данных, при котором точность классификации будет снижена без снижения уровня полезности для других видов приложений. В [23] и [101] были предложены методики снижения качества классификации в контексте правил классификации и приложений, связанных с построением дерева решений. Определение “скупого снижения” (parsimonious downgrading) введено в [101] в контексте блокировки выдачи результатов классификации с последующим изучением ее влияния на общую полезность. Система Rational Downgrader [23] была разработана с использованием этих принципов.

К снижению эффективности классификаторов могут адаптироваться методы сокрытия ассоциативных правил. В [102], предложен метод снижения эффективности классификаторов, базирующихся на правилах, и показана его эффективность.

Обработка запросов (и контроль логических выводов). Многие массивы ПД не являются полностью открытыми, но имеют открытый интерфейс. В этом есть очевидная угроза того, что злоумышленник может получить чувствительную информацию путем построения “правильной” серии запросов. Уровень такой угрозы может быть характеризован как “полное раскрытие”, когда злоумышленник получает точные значения чувствительных атрибутов, или “частичное раскрытие”, когда удается только сузить множество возможных значений атрибута. Есть два подхода противодействия такой угрозе. Первый – аудит запросов – состоит в том, что один или несколько запросов из последовательности могут быть отклонены с тем, чтобы сохранить анонимность исходных данных (см. примеры в [103, 104]). Второй подход состоит в управлении логическими заключениями. В этом случае либо исходные данные, либо результат запроса искажаются таким образом, чтобы сохранить уровень анонимности исходных данных (см. примеры в [33, 105, 106], методы искажения результатов запросов – в [19, 107–110]).

2. ВОПРОСЫ РЕАЛИЗАЦИИ АЛГОРИТМОВ ОБЕЗЛИЧИВАНИЯ

2.1. Сохранение полезности обезличенных данных

Потеря части информации является естественным следствием обезличивания ПД. Вопрос состоит в том, является ли утерянная часть полезной с точки зрения конечного потребителя обезличенной информации. В [14] представлен негативный ответ: обеспечение требуемого уровня анонимности приводит к исключению слишком большого числа атрибутов ПД. Вопрос об оценке уровня полезности часто сопровождает конкретную процедуру обезличивания. Так, для некоторых методов обезличивания [17, 49, 50, 111] предложены вполне конкретные показатели для потери информации от процесса обезличивания. Примерами таких показателей являются “высота обобщения” (общее число шагов обобщения, которые приводят исходные данные к нынешнему обезличенному виду), размеры анонимных групп в терминах K-анонимизации, меры различимости значений полей, относительная потеря информации при обезличивании и пр. Ряд предлагаемых метрик типа классификационной метрики [56] были специально разработаны для оценки полезности обезличенных данных для решения конкретных прикладных задач статистического анализа.

Задача обезличивания, сохраняющего полезность, и последующего безопасного интеллектуального анализа данных впервые была поставлена в [112]. Основная идея заключалась в отдельном опубликовании частных таблиц, имеющих полезность, но при этом обеспечивающих нужный уровень анонимности.

Еще один интересный подход к обезличиванию с сохранением полезности представлен в [113]. Основная идея состоит в том, чтобы зада-

вать для разных атрибутов разную полезность. В отличие от обсуждаемых выше методов обезличивания, которые называют глобальными из-за того, что они каждый атрибут исходных данных отображают на то же множество значений, этот подход предлагает разбиение исходного пространства данных на несколько областей, учитывающих полезность, с последующим отображением значения атрибута в зависимости от принадлежности той или иной области. Концептуально схожий подход, но на основе адаптации уровня анонимности данных в зависимости от частоты обращения к ним пользователей, предложен в [114].

Надо отметить, что всегда уровень полезности обезличенных данных рассматривается только в контексте конкретных прикладных задач и выбор функции для определения уровня полезности адаптируется именно к этим задачам. Например, в [50] для анализа свойств обезличенных данных, удовлетворяющих условию K-анонимности, рассматривалась величина потери информации (т.е. разности информационных энтропий до и после анонимизации). Подобный показатель позволяет оценить полезность данных для решения задач классификации. В [115] был предложен метод, сохраняющий точность базовых запросов (например, сохранение коэффициентов корреляции).

2.2. Ограничение применимости методов обезличивания

Возможности применения многих методов анализа обезличенных данных изначально ограничены “проклятием размерности” из-за наличия общедоступной информации. Например, в [14] представлен анализ одного метода K-анонимизации с ростом размерности данных. Рост размерности начинает оказывать значимое влияние, когда доступен большой объем справочной информации, в результате применения которой граница между квазиидентификаторами и атрибутами чувствительной информации становится размытой. Как уже упоминалось, подобный факт является стимулом для развития таких методов обезличивания, как L-разнообразие и t-близость. Но и без этого с ростом размерности данных обеспечение K-анонимности становится фактически невозможным из-за соответствующего роста групп анонимности. При постоянном числе записей субъектов ПД в исходном массиве данных появление новых атрибутов неизбежно ведет к разреженному характеру заполнения групп анонимности, поскольку чем больше атрибутов, тем меньшее число субъектов ПД (записей) попадает в ту или иную группу. В [14] делается важный пессимистический вывод о том, что для сохранения уровня анонимности может потребоваться исключение большого числа атрибутов. Такой же вывод в [17] был сделан в отношении модели L-разнообразия. Ясно, что такие операции снижают уровень полезности обезличенных данных.

Свойства методов рандомизации были исследованы в [28]. Предполагалось, что распределение исходных данных известно и они обезличены путем добавления шума. Задача восстановления данных решалась в постановке оценки максимального правдоподобия. Уровень шума предполагалось выбрать таким образом, чтобы оценка максимального правдоподобия указывала на другие записи, нежели на истинные. В работе показано, что с ростом размерности данных вероятность “обмануть” таким образом метод максимального правдоподобия также уменьшается.

Таким образом, проблема роста размерности данных является фундаментальным вызовом для методов обезличивания ПД, обеспечивающих высокий уровень анонимности. Маловероятно, что можно найти более эффективные методы для сохранения уровня анонимности, в случае доступности не только детальной информации о фоновых значениях, но и о некоторых выбранных группах субъектов ПД. Косвенные примеры таких нарушений приведены в [116, 117]. Описаны ситуации, когда информация из нескольких источников может быть скомпилирована для создания высокоразмерного представления, нарушающего ограничения анонимности для субъекта ПД. Очень ярким примером такой компиляции может служить совместная обработка потоковых данных о текущем геопозиционировании группы субъектов и одновременно получаемых данных из социальных сетей, раскрывающая текущее местоположение конкретного субъекта.

2.3. Чувствительность методов обезличивания к угрозам деобезличивания

В литературе обсуждается весьма широкий спектр возможных атак на обезличенные данные. Ограничиваясь кратким качественным анализом этих атак, сгруппируем их по результатам, которые могут быть достигнуты. Обобщенно их можно назвать угрозами, и исследовать каждый метод обезличивания по отношению к этим угрозам. Более детальный анализ приведен в работе [118], предлагающей именно такой подход. Для наших целей достаточно выделить три типа возможных угроз.

Угроза выделения (Singling Out) заключается в возможности изолировать некоторые или все записи, идентифицирующие субъект ПД в имеющемся наборе данных.

Угроза связывания (Linkability) заключается в возможности сопоставить как минимум две записи относительно одного и того же субъекта ПД или группы субъектов (в одном и том же или разных массивах данных). Если злоумышленник мо-

жет определить (например, с помощью корреляционного анализа), что две записи относятся к одной и той же группе лиц, но не может выделить отдельных участников в этой группе, то метод устойчив к выделению, но неустойчив к связыванию.

Угроза умозаключения, угроза логических выводов (Inference) заключается в возможности выделить, со значительной вероятностью, значение атрибута из общего множества значений атрибутов, для одной записи или их группы.

Все методы обезличивания, примененные отдельно, в той или иной степени подвержены всем перечисленным типам угроз. Некоторые из угроз могут быть устранены полностью или частично в конкретной ситуации, например, путем использования комбинации методов. Гарантированно противодействовать всем угрозам для некоторого универсального массива ПД не представляется возможным, поэтому такое большое внимание уделяется проектированию каждого конкретного приложения обезличивания ПД.

3. ПРИЛОЖЕНИЯ И ПРИМЕРЫ ОБЕЗЛИЧИВАНИЯ

3.1. Медицинские базы данных

Медицинская информация – возможно, самый очевидный источник ПД, обладающих высоким уровнем полезности. И при этом так же очевидна чувствительность этой информации к возможным утечкам. В качестве примеров систем обезличивания медицинской информации приведем Scrub [119] и Datafly [120], чей заслуженный возраст подчеркивает внимание к теме.

Scrub была создана для обезличивания клинических записей и писем, представленных в текстовой форме. Подобные записи обычно содержат ссылки на пациентов, членов их семей, адреса, телефоны и пр., но вместе с этим часто встречаются “загадочные” ссылки в виде сокращений, понятных только специалистам. Scrub параллельно использует несколько алгоритмов обнаружения текстов имен, адресов или номеров телефонов, применяет конкурентно локальные источники данных, специфические для данной области. В [119] показано, что система способна удалить из данных более 99% персональной идентификационной информации.

Datafly – одно из первых практических приложений преобразований, сохраняющих уровень анонимности. Эта система была разработана для предотвращения идентификации собственников медицинских карт, которые могут содержать множество атрибутов, в том числе персональные идентификаторы и квазиидентификаторы. Система была разработана в качестве ответа на опасения, что процесс удаления только персональных идентификаторов недостаточен для гарантии анонимности. Система имеет те же истоки, которые породили концепцию *K*-анонимности, но при создании Datafly самой концепции не было. Довольно простой подход в Datafly использует установку минимального размера ячейки для каждого атрибута. Уровень анонимности (отметим, что именно здесь уже формулируется это понятие) определяется в Datafly относительно именно установленного размера. Таким образом, значения в записях обобщаются до уровня неоднозначности размера ячейки, а не до точных значений. Идентифицирующие поля удаляются из данных. Кроме того, исключаются выбросы значений. Пользователь имеет возможность задать уровень анонимности в зависимости от профиля соответствующего получателя данных. Общий уровень анонимности определяется числовым параметром, лежащим в пределах от 0 до 1. Значение 0 соответствует исходным (неизмененным) данным, значение 1 – максимальному уровню обобщения. С момента создания Datafly прошло много времени и было выполнено большое число исследований в области обезличивания, в настоящий момент можно составить солидный список улучшений/модернизаций. И, конечно, к настоящему времени накопилось огромное число приложений медицинских ПД. Мы привели эти два примера с той лишь целью, чтобы подчеркнуть солидный исторический возраст проблематики.

3.2. Выявление возможного применения биологического оружия

Одна из актуальных задач применения обезличенных медицинских данных – своевременное обнаружение применения биологического оружия. Наверное, уместно учесть события с COVID-19 последних лет и добавить сюда “а также обнаружения и мониторинга опасных вирусных заболеваний”. Применение биологического оружия внешне, вызывает симптомы, сходные с симптомами других распространенных респираторных заболеваний: кашель, головная боль и жар. При отсутствии предварительных сведений о факте биологической атаки поставщики медицинских услуг могут диагностировать у пациента симптомы одного из наиболее распространенных и относительно безопасных респираторных заболеваний. Главная задача заключается в быстрой идентификации настоящей биологической атаки или пандемии в отличие от вспышки обычного респираторного заболевания. Во многих случаях необычное число или сочетание типовых симптомов в данной местности может указывать на такую атаку. Следовательно, для выявления необходимо отслеживать частоту появления “безопасных распространенных” заболеваний, а значит соответствующие данные необходимо обезличе-

но собирать/регистрировать/обрабатывать в органах общественного здравоохранения. Решение, предложенное в [8] – это "выборочное раскрытие" (снижение уровня анонимности по интересующим атрибутам), которое изначально допускает только ограниченный доступ к данным. Однако в случае подозрительной активности оно позволяет "углубиться" в базовые данные для уверенной идентификации вспышек опасных эпидемий.

3.3. Приложения по охране общественного порядка

Функционирование приложений, используемых для обеспечения общественного порядка, часто связано с обработкой результатов различных систем наблюдения. В [121] дается широкий обзор того, как эффективно использовать результаты наблюдения с сохранением уровня анонимности. Вот некоторые примеры таких приложений.

Система проверки учетных данных пытается сопоставить субъект данных с лицом, представляющим учетные данные. Кража паспортных данных или номера СНИЛС представляет серьезную угрозу для общественного порядка. В подходе к проверке учетных данных в [121] делается попытка использовать семантику, связанную с этими номерами, чтобы определить, действительно ли лицо, представляющее учетные данные, владеет им.

Система профилактики краж личных данных "ангел идентичности" (identity angel) [9] предлагает более активный подход. В ней анализируются данные данные из онлайн источников и обнаруживаются люди, которым грозит опасность кражи личных данных. Эта информация может быть использована для уведомления.

Система веб-камер наблюдения (система видеонаблюдения). Одним из возможных методов наблюдения является использование общедоступных веб-камер, которые можно использовать для обнаружения необычной активности [10, 121]. Надо заметить, что это гораздо больше нарушает анонимность, чем другие виды наблюдения, потому что зафиксированная в веб-камерах информация о субъекте включает изображение его лица, которое очевидно является персональным идентификатором. При этом исключить данную информацию гораздо сложнее в сравнении с исключением некоторой совокупности атрибутов из записи о субъекте ПД. Обезличивание фото- и видеоинформации предлагается проводить, извлекая из изображений только информацию о числе лиц и используя только ее для обнаружения необычной активности. В [10] была выдвинута гипотеза, что необычную активность можно обнаружить только по числу лиц, а не с использованием детальной информации о конкретных людях. Фактически, такой подход использует понижение уровня информации, доступной в веб-камерах, в зависимости от предметной области, чтобы сделать подход безопасным с точки зрения сохранения уровня анонимности.

Если в системе, связывающей камеры, есть функциональность по распознаванию лиц, угроза нарушения анонимности становится существенно выше. Для обезличивания самым простым решением было бы полное затемнение всех лиц, что выглядело бы странно для конечного пользователя. Более сбалансированный подход [7] заключается в использовании выборочного понижения качества изображения таким образом, чтобы ограничить способность надежного применения программного обеспечения распознавания лиц, сохраняя при этом детали лица на изображениях. Алгоритм называется k-Same и его ключевая идея состоит в том, чтобы идентифицировать лица, которые в некоторой степени похожи, а затем синтезировать "новые лица", как комбинации из похожих. Таким образом, личность владельца лица до некоторой степени анонимна, но видео остается полезным. Этот метод чем-то похож на K-анонимность, за исключением того, что он создает новые синтезированные данные для приложения.

3.4. Генетические данные

В последние годы были достигнуты успехи в области секвенирования ДНК и судебно-медицинской экспертизы с использованием ДНК. В результате базы данных ДНК очень быстро растут как в медицинском, так и в правоохранительном сообществе. Данные ДНК считаются крайне чувствительными, так как представляют собой информацию, почти однозначно идентифицирующую отдельного человека, а также важную медико-биологическую информацию о нем. Как и в случае роста размерности ПД, простого удаления непосредственно идентификационных данных, сопровождающих ДНК, недостаточно для предотвращения деобезличивания. В [122] показано, что программное обеспечение, обрабатывающее данные ДНК, может идентифицировать субъекта ПД независимо от другой информации. Такое программное обеспечение обрабатывает общедоступные медицинские данные и знания о различных конкретных заболеваниях, чтобы идентифицировать записи ДНК. В [122] показано, что идентифицировать можно 98–100% субъектов. Идентификация осуществляется путем взятия образцов ДНК человека и последующего построения генетического профиля, соответствующего полу, генетическим заболеваниям, месту сбора ДНК и т.д. Этот генетический профиль весьма эффективен при деобезличивании. Один из способов обеспе-

чить анонимность генетических данных – это использование решеток обобщения [123], являющихся, по сути, реализацией концепции *K*-анонимизации. Другой подход, обсуждаемый в [62], создает синтетические данные, которые имеют совокупные характеристики исходных данных, но сохраняют уровень анонимности исходных записей.

Одним из способов нарушения конфиденциальности геномных данных является последовательная реидентификация (trial re-identification), при которой уникальность шаблонов пациентов [116, 117] используется для деобезличивания. Предпосылка этих работ заключается в том, что пациенты часто посещают и оставляют генетические данные в "информационно разнесенных" местах: клиниках, лабораториях и пр. Больницы обычно отделяют клинические данные от генетических и предоставляют генетические данные для исследовательских целей. При этом данные кажутся обезличенными, а схема посещения пациентов кодируется на сайте, с которого они получены. В [116, 117] показано, что эта информация может быть объединена с общедоступными данными, чтобы выполнить деобезличивание.

4. ЗАКЛЮЧЕНИЕ

Выполненный обзор проблематики автоматизации обезличивания ПД был бы неполным без упоминания хотя бы некоторых резонансных случаев нарушения анонимности обезличенных ПД. Именно отдельные, уникальные примеры скандального характера инициировали и поддержали усилия исследователей в этой области, дали основу для придания этим исследованиям фундаментального характера. Заметим, кстати, что далеко не все резонансные ситуации носили и носят противоправный характер. Не всегда источником атаки на уровень конфиденциальности оказываются злоумышленники, действующие с целью незаконного извлечения какой-то прибыли. Чаще виновник оказывается таковым по причине некомпетентности в области ПД вообще и обезличивания в частности. К сожалению, также не всегда даже резонансные случаи нарушения анонимности подвергаются методическому исследованию. Информация о таких историях поступает чаще из газетных публикаций и по большей части направлена на привлечение внимания, чем на обстоятельное доказательное и аналитическое исследование. Но любопытно хотя бы обозначить фигурантов этих историй. Так, статья [124] посвящена истории скандала между подразделением Google, компанией DeepMind и медицинским фондом Royal Free London NHS Foundation Trust. Нашумевший случай сбора персональной информации профилей Facebook компанией Cambridge Analytica описан в [125]. Статья [126] касается случая с небольшим индейским племенем хавасупай и исследователями Университета Аризоны. Случай, инспирированный специалистом в области ПД Латаньей Суинни, продемонстрировавшей возможность деобезличивания в отношении губернатора Массачусетса Уильяма Вэлда, приведен в [127, 128]. Довольно аккуратные статистические исследования данных в отношении граждан США выполнены в [130] и [131]. В статье [132] декларирован факт деобезличивания медицинской информации, опубликованной департаментом здравоохранения Австралии, а в [133] – возможность деобезличивания субъектов ПД по общедоступному набору данных об аренде велосипедов в Лондоне. Проблемы возможного деобезличивания субъектов по данным электронных проездных документов в Риге представлены в статье [134]. Наконец, одному из наиболее громких и "математически проработанных" случаев деобезличивания, связанному с базой данных Netflix, посвящена работа [135].

Эти и многие другие факты угроз нарушения анонимности обезличенных ПД, прежде всего, вызывают большой общественный резонанс. Но и научное сообщество, как следует из представленного материала, уделяет весьма большое внимание проблеме обезличивания ПД. Причем для исследований используется вполне современный математический аппарат для решения задач, которые можно сгруппировать по следующим направлениям:

- обеспечение безопасного сбора персональных данных,
- построение надежных алгоритмов обезличивания персональных данных (существующих и новых типов),
- оценка полезности обезличенных данных,
- поиск уязвимостей существующих и перспективных алгоритмов обезличивания,
- поиск квазиидентификаторов в обезличенных данных,
- поиск ассоциативных правил в обезличенных данных,
- разработка методов безопасной обработки персональных данных,
- разработка методов безопасной распределенной обработки персональных данных и безопасных вычислений.

В этих исследованиях имеются возможности для применения интеллектуальных усилий специалистов самых разных компетенций. Так, авторы обратили внимание на два аспекта, характерных для всех методов обезличивания. Во-первых, в рамках каждого метода обсуждается некоторый показатель его надежности – границы, в рамках которых можно говорить о поддержании уровня анонимности. Но некоторая вероятность деобезличивания имеется всегда. Говорить об исключе-

нии возможности нарушения анонимности можно лишь для методов, основанных на искусственном синтезе обезличенных данных. Во-вторых, компьютерному синтезу обезличенных данных внимания уделяется довольно мало, практически отсутствуют сколь-либо универсальные методики. Вторая часть нашего исследования посвящена применению различных методов обезличивания, в том числе компьютерного синтеза данных. В ней будет представлена математическая модель процесса, даны формальные определения, предложен метод обезличивания, основанный на комбинации простых статистических характеристик и ядерных оценок Розенблатта–Парзена, выполнены численные эксперименты.

6. БЛАГОДАРНОСТИ

Работа выполнялась с использованием инфраструктуры Центра коллективного пользования "Высокопроизводительные вычисления и большие данные" (ЦКП "Информатика" ФИЦ ИУ РАН, Москва).

СПИСОК ЛИТЕРАТУРЫ

1. *Aggarwal C.C., Yu P.S.* A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In: *Aggarwal C.C., Yu P.S.* (eds) Privacy-Preserving Data Mining. Advances in Database Systems. 2008. V. 34. Springer, Boston, MA.
2. *Domingo-Ferrer J., Farràs O., Ribes-González J., Sánchez D.* Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges // Computer Communications. 2019. V. 140–141. P. 38–60.
3. *Sahi M.A. et al.* Privacy Preservation in e-Healthcare Environments: State of the Art and Future Directions // IEEE Access. 2018. V. 6. P. 464–478. https://doi.org/10.1109/ACCESS.2017.2767561
4. *Spiekermann S., Cranor L.F.* Engineering Privacy // IEEE Transactions on Software Engineering. 2009. V. 35. № 1. P. 67–82. https://doi.org/10.1109/TSE.2008.88
5. *Verykios V.S., Bertino E., Fovino I.N., Provenza L.P., Saygin Y., Theodoridis Y.* State-of-the-art in privacy preserving data mining // ACM SIGMOD Record. 2004. V. 33. № 1.
6. *Guide to Basic Data Anonymization Technique*. Personal Data Protection Commission, Singapore. 2018.
7. *Newton E., Sweeney L., Malin B.* Preserving Privacy by De-identifying Facial Images // IEEE Transactions on Knowledge and Data Engineering. 2005.
8. *Sweeney L.* Privacy-Preserving Bio-terrorism Surveillance // AAAI Spring Symposium, AI Technologies for Homeland Security. 2005.
9. *Sweeney L.* AI Technologies to Defeat Identity Theft Vulnerabilities // AAAI Spring Symposium, AI Technologies for Homeland Security. 2005.
10. *Sweeney L., Gross R.* Mining Images in Publicly-Available Cameras for Homeland Security // AAAI Spring Symposium, AI Technologies for Homeland Security. 2005.
11. *Agrawal R., Srikant R.* Privacy-Preserving Data Mining // Proceedings of the ACM SIGMOD Conference. 2000.
12. *Agrawal D., Aggarwal C.C.* On the Design and Quantification of Privacy-Preserving Data Mining Algorithms // ACM PODS Conference. 2002.
13. *Aggarwal G., Feder T., Kenthapadi K., Motwani R., Panigrahy R., Thomas D., Zhu A.* Approximation Algorithms for k-anonymity. Journal of Privacy Technology. 2005. № 20051120001.
14. *Aggarwal C.C.* On k-anonymity and the curse of dimensionality // VLDB Conference. 2005.
15. *LeFevre K., DeWitt D., Ramakrishnan R.* Incognito: Full Domain K-Anonymity // ACM SIGMOD Conference. 2005.
16. *Meyerson A., Williams R.* On the complexity of optimal k-anonymity // ACM PODS Conference. 2004.
17. *Machanavajjhala A., Gehrke J., Kifer D., Venkitasubramaniam M.* L-Diversity: Privacy Beyond k-Anonymity // ICDE Conference. 2006.
18. *Li N., Li T., Venkatasubramanian S.* t-Closeness: Privacy beyond k-anonymity and l-diversity // ICDE Conference. 2007.
19. *Dwork C., Nissim K.* Privacy-Preserving Data Mining on Vertically Partitioned Databases // CRYPTO. 2004.
20. *Vaidya J., Clifton C.* Privacy-Preserving Decision Trees over vertically partitioned data // Lecture Notes in Computer Science. 2005. V. 3654.
21. *Yu H., Vaidya J., Jiang X.* Privacy-Preserving SVM Classification on Vertically Partitioned Data // PA-KDD Conference. 2006.
22. *Verykios V.S., Elmagarmid A., Bertino E., Saygin Y., Dasseni E.* Association Rule Hiding // IEEE Transactions on Knowledge and Data Engineering. 2004. V. 16. № 4.
23. *Moskowitz I., Chang L.* A decision theoretic system for information downgrading // Joint Conference on Information Sciences. 2000.
24. *Adam N., Wortmann J.C.* Security-Control Methods for Statistical Databases: A Comparison Study // ACM Computing Surveys. 1989. V. 21. № 4.
25. *Liew C.K., Choi U.J., Liew C.J.* A data distortion by probability distribution // ACM TODS. 1985. V. 10. № 3. P. 395–411.
26. *Warner S.L.* Randomized Response: A survey technique for eliminating evasive answer bias // Journal of American Statistical Association. 1965. V. 60. № 309. P. 63–69.
27. *Silverman B.W.* Density Estimation for Statistics and Data Analysis. Chapman and Hall. 1986.
28. *Aggarwal C.C.* On Randomization, Public Information and the Curse of Dimensionality // ICDE Conference. 2007.
29. *Gambs S., Kegl B., Aimeur E.* Privacy-Preserving Boosting // Knowledge Discovery and Data Mining Journal. 2007. V. 14. № 1. P. 131–170.
30. *Zhang P., Tong Y., Tang S., Yang D.* Privacy-Preserving Naive Bayes Classifier // Lecture Notes in Computer Science. 2005. V. 3584.

31. *Evfimievski A., Srikant R., Agrawal R., Gehrke J.* Privacy-Preserving Mining of Association Rules // ACM KDD Conference. 2002.
32. *Rizvi S., Haritsa J.* Maintaining Data Privacy in Association Rule Mining // VLDB Conference. 2002.
33. *Agrawal R., Srikant R., Thomas D.* Privacy-Preserving OLAP // Proceedings of the ACM SIGMOD Conference. 2005.
34. *Polat H., Du W.* SVD-based collaborative filtering with privacy // ACM SAC Symposium. 2005.
35. *Bertino E., Fovino I., Provenza L.* A Framework for Evaluating Privacy-Preserving Data Mining Algorithms // Data Mining and Knowledge Discovery Journal. 2005. V. 11. P. 121–154.
36. *Evfimievski A., Gehrke J., Srikant R.* Limiting Privacy Breaches in Privacy Preserving Data Mining // ACM PODS Conference. 2003.
37. *Huang Z., Du W., Chen B.* Deriving Private Information from Randomized Data // ACM SIGMOD Conference. 2005. P. 37–48.
38. *Kargupta H., Datta S.,Wang Q., Sivakumar K.* On the Privacy Preserving Properties of Radom Data Perturbation Techniques // ICDM Conference. 2003. P. 99–106.
39. *Johnson W., Lindenstrauss J.* Extensions of Lipshitz Mapping into Hilbert Space // Contemporary Math. 1984. V. 26. P. 189–206.
40. *Oliveira S.R.M., Zaiane O.* Privacy Preserving Clustering by Data Transformation // Proc. 18th Brazilian Symp. Databases. 2003. P. 304–318.
41. *Oliveira S.R.M., Zaiane O.* Data Perturbation by Rotation for Privacy-Preserving Clustering // Technical Report TR04-17, Department of Computing Science, University of Alberta, Edmonton, AB, Canada. 2004.
42. *Chen K., Liu L.* Privacy-preserving data classification with rotation perturbation // ICDM Conference. 2005.
43. *Liu K., Kargupta H., Ryan J.* Random Projection Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining // IEEE Transactions on Knowledge and Data Engineering. 2006. V. 18. № 1.
44. *Kim J., Winkler W.* Multiplicative Noise for Masking Continuous Data // Technical Report Statistics 2003-01, Statistical Research Division, US Bureau of the Census, Washington D.C. 2003.
45. *Mukherjee S., Chen Z., Gangopadhyay S.* A privacy-preserving technique for Euclidean distance-based mining algorithms using Fourier based transforms // VLDB Journal. 2006.
46. *Liu K., Giannella C. Kargupta H.* An Attacker's View of Distance Preserving Maps for Privacy-Preserving Data Mining // PKDD Conference. 2006.
47. *Fienberg S., McIntyre J.* Data Swapping: Variations on a Theme by Dalenius and Reiss // Technical Report, National Institute of Statistical Sciences. 2003.
48. *Samarati P.* Protecting Respondents' Identities in Microdata Release // IEEE Trans. Knowl. Data Eng. 2001. V. 13. № 6. P. 1010–1027.
49. *Bayardo R.J., Agrawal R.* Data Privacy through Optimal k-Anonymization // Proceedings of the ICDE Conference. 2005. P. 217–228.
50. *Fung B., Wang K., Yu P.* Top-Down Specialization for Information and Privacy Preservation // ICDE Conference. 2005.
51. *Wang K., Yu P., Chakraborty S.* Bottom-Up Generalization: A Data Mining Solution to Privacy Protection // ICDM Conference. 2004.
52. *Domingo-Ferrer J., Mateo-Sanz J.* Practical data-oriented micro-aggregation for statistical disclosure control // IEEE TKDE. 2002. V. 14. № 1.
53. *Aggarwal G., Feder T., Kenthapadi K., Khuller S., Motwani R., Panigrahy R., Thomas D., Zhu A.* Achieving Anonymity via Clustering // ACM PODS Conference. 2006.
54. *Aggarwal C.C., Yu P.S.* A Condensation approach to privacy preserving data mining // EDBT Conference. 2004.
55. *Winkler W.* Using simulated annealing for k-anonymity // Technical Report 7, US Census Bureau, Washington D.C. 20233. 2002.
56. *Iyengar V.S.* Transforming Data to Satisfy Privacy Constraints // KDD Conference. 2002.
57. *Lakshmanan L., Ng R., Ramesh G.* To Do or Not To Do: The Dilemma of Disclosing Anonymized Data // ACM SIGMOD Conference. 2005.
58. *Aggarwal C.C., Yu P.S.* On Variable Constraints in Privacy-Preserving Data Mining // SIAM Conference. 2005.
59. *Xiao X., Tao Y.* Personalized Privacy Preservation // ACM SIGMOD Conference. 2006.
60. *Wang K., Fung B.C.M.* Anonymization for Sequential Releases // ACM KDD Conference. 2006.
61. *Pei J., Xu J., Wang Z., Wang W., Wang K.* Maintaining k-Anonymity against Incremental Updates // Symposium on Scientific and Statistical Database Management. 2007.
62. *Aggarwal C.C., Yu P.S.* On Privacy-Preservation of Text and Sparse Binary Data with Sketches // SIAM Conference on Data Mining. 2007.
63. *Aggarwal C.C., Yu P.S.* On Anonymization of String Data // SIAM Conference on Data Mining. 2007.
64. *Martin D., Kifer D., Machanavajjhala A., Gehrke J., Halpern J.* Worst-Case Background Knowledge // ICDE Conference. 2007.
65. *Pinkas B.* Cryptographic Techniques for Privacy-Preserving Data Mining // ACM SIGKDD Explorations. 2002. V. 4. № 2.
66. *Even S., Goldreich O., Lempel A.* A Randomized Protocol for Signing Contracts // Communications of the ACM. 1985. V. 28.
67. *Rabin M.O.* How to exchange secrets by oblivious transfer // Washington D.C. 20233TR-81, Aiken Corporation Laboratory. 1981.
68. *Naor M., Pinkas B.* Efficient Oblivious Transfer Protocols // SODA Conference. 2001.
69. *Yao A.C.* How to Generate and Exchange Secrets // FOCS Conference. 1986.
70. *Chaum D., Crepeau C., Damgard I.* Multiparty unconditionally secure protocols // ACM STOC Conference. 1988.
71. *Ioannidis I., Grama A., Atallah M.* A secure protocol for computing dot-products in clustered and distributed

environments // International Conference on Parallel Processing. 2002.

72. *Du W., Atallah M.* Secure Multi-party Computation: A Review and Open Problems // CERIAS Technical Report 2001-51, Purdue University. 2001.
73. *Clifton C., Kantarcioglou M., Lin X., Zhu M.* Tools for privacy preserving distributed data mining // ACM SIGKDD Explorations. 2002. V. 4. № 2.
74. *Lindell Y., Pinkas B.* Privacy-Preserving Data Mining // CRYPTO. 2000.
75. *Kantarcioglu M., Vaidya J.* Privacy-Preserving Naive Bayes Classifier for Horizontally Partitioned Data // IEEE Workshop on Privacy-Preserving Data Mining. 2003.
76. *Yu H., Jiang X., Vaidya J.* Privacy-Preserving SVM using nonlinear Kernels on Horizontally Partitioned Data // SAC Conference. 2006.
77. *Yang Z., Zhong S., Wright R.* Privacy-Preserving Classification of Customer Data without Loss of Accuracy // SDM Conference. 2006.
78. *Kantarcioglu M., Clifton C.* Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data // IEEE TKDE Journal. 2004. V. 16. № 9.
79. *Inan A., Saygin Y., Savas E., Hintoglu A., Levi A.* Privacy-Preserving Clustering on Horizontally Partitioned Data // Data Engineering Workshops. 2006.
80. *Jagannathan G., Wright R.* Privacy-Preserving Distributed k-means clustering over arbitrarily partitioned data // ACM KDD Conference. 2005.
81. *Jagannathan G., Pillaipakkamnatt K., Wright R.* A New Privacy-Preserving Distributed k-Clustering Algorithm // SIAM Conference on Data Mining. 2006.
82. *Polat H., Du W.* Privacy-Preserving Top-N Recommendations on Horizontally Partitioned Data // Web Intelligence. 2005.
83. *Bawa M., Bayardo R.J., Agrawal R.* Privacy-Preserving Indexing of Documents on the Network // VLDB Conference. 2003.
84. *Vaidya J., Clifton C.* Privacy-Preserving Association Rule Mining in Vertically Partitioned Databases // ACM KDD Conference. 2002.
85. *Vaidya J., Clifton C.* Privacy-Preserving Naive Bayes Classifier over vertically partitioned data // SIAM Conference. 2004.
86. *Vaidya J., Clifton C.* Privacy-Preserving k-means clustering over vertically partitioned Data // ACM KDD Conference. 2003.
87. *Jiang W., Clifton C.* Privacy-preserving distributed k-Anonymity // Proceedings of the IFIP 11.3 Working Conference on Data and Applications Security. 2005.
88. *Wang K., Fung B.C.M., Dong G.* Integrating Private Databases for Data Analysis // Lecture Notes in Computer Science. 2005. V. 3495.
89. *Zhong S., Yang Z., Wright R.* Privacy-enhancing k-anonymization of customer data // Proc. of the ACM SIGMOD-SIGACT-SIGART Principles of Database Systems, Baltimore, MD. 2005.
90. *Bettini C., Wang X.S., Jajodia S.* Protecting Privacy against Location Based Personal Identification // Proc. of Secure Data Management Workshop, Trondheim, Norway. 2005.
91. *Gedik B., Liu L.* A customizable k-anonymity model for protecting location privacy // ICDCS Conference. 2005.
92. *Mimoto T., Kiyomoto Sh., Miyaji A.* Secure Data Management Technology // In Security Infrastructure Technology for Integrated Utilization of Big Data (*T. Mimoto and A. Miyaji* eds.), Singapore, Springer Open. 2020.
93. *Oliveira S.R.M., Zaiane O., Saygin Y.* Secure Association-Rule Sharing // PAKDD Conference. 2004.
94. *Saygin Y., Verykios V., Clifton C.* Using Unknowns to prevent discovery of Association Rules // ACM SIGMOD Record. 2001. V. 30. № 4.
95. *Atallah M., Elmagarmid A., Ibrahim M., Bertino E., Verykios V.* Disclosure limitation of sensitive rules // Workshop on Knowledge and Data Engineering Exchange. 1999.
96. *Dasseni E., Verykios V., Elmagarmid A., Bertino E.* Hiding Association Rules using Confidence and Support // 4th Information Hiding Workshop. 2001.
97. *Chang L., Moskowitz I.* An integrated framework for database inference and privacy protection. Data and Applications Security. Kluwer. 2000.
98. *Saygin Y., Verykios V., Elmagarmid A.* Privacy-Preserving Association Rule Mining // 12th International Workshop on Research Issues in Data Engineering. 2002.
99. *Wu Y.-H., Chiang C.-M., Chen A.L.P.* Hiding Sensitive Association Rules with Limited Side Effects // IEEE Transactions on Knowledge and Data Engineering. 2007. V. 19. № 1.
100. *Aggarwal C., Pei J., Zhang B.* A Framework for Privacy Preservation against Adversarial Data Mining // ACM KDD Conference. 2006.
101. *Chang L., Moskowitz I.* Parsimonious downgrading and decision trees applied to the inference problem // New Security Paradigms Workshop. 1998.
102. *Natwichai J., Li X., Orlowska M.* A Reconstruction-based Algorithm for Classification Rules Hiding // Australasian Database Conference. 2006.
103. *Kenthapadi K., Mishra N., Nissim K.* Simulatable Auditing // ACM PODS Conference. 2005.
104. *Nabar S., Marthi B., Kenthapadi K., Mishra N., Motwani R.* Towards Robustness in Query Auditing // VLDB Conference. 2006.
105. *Chawla S., Dwork C., McSherry F., Smith A., Wee H.* Towards Privacy in Public Databases // TCC. 2005.
106. *Mishra N., Sandler M.* Privacy vs Pseudorandom Sketches // ACM PODS Conference. 2006.
107. *Blum A., Dwork C., McSherry F., Nissim K.* Practical Privacy: The SuLQ Framework // ACM PODS Conference. 2005.
108. *Dinur I., Nissim K.* Revealing Information while preserving privacy // ACM PODS Conference. 2003.
109. *Dwork C., Kenthapadi K., McSherry F., Mironov I., Naor M.* Our Data, Ourselves: Privacy via Distributed Noise Generation // EUROCRYPT. 2006.
110. *Dwork C., McSherry F., Nissim K., Smith A.* Calibrating Noise to Sensitivity in Private Data Analysis // TCC. 2006.

111. *Wang K., Fung B.C.M., Yu P.* Template based Privacy-Preservation in classification problems // ICDM Conference. 2005.

112. *Kifer D., Gehrke J.* Injecting utility into anonymized datasets // SIGMOD Conference. 2006. P. 217–228.

113. *Xu J., Wang W., Pei J., Wang X., Shi B., Fu A.W.C.* Utility Based Anonymization using Local Recoding // ACM KDD Conference. 2006.

114. *LeFevre K., DeWitt D., Ramakrishnan R.* Workload Aware Anonymization // KDD Conference. 2006.

115. *Koudas N., Srivastava D., Yu T., Zhang Q.* Aggregate Query Answering on Anonymized Tables // ICDE Conference. 2007.

116. *Malin B., Sweeney L.* Re-identification of DNA through an automated linkage process // Proc. AMIA Symp. 2001. P. 423–427.

117. *Malin B.* Why methods for genomic data privacy fail and what we can do to fix it // AAAS Annual Meeting, Seattle, WA. 2004.

118. ARTICLE 29 DATA PROTECTION WORKING PARTY. Opinion 05/2014 on Anonymisation Techniques. Adopted on 10 April 2014.

119. *Sweeney L.* Replacing Personally Identifiable Information in Medical Records, the Scrub System // Proc. AMIA Annu Fall Symp. 1996. P. 333–337.

120. *Sweeney L.* Guaranteeing Anonymity while Sharing Data, the Datafly System // Proc. AMIA Annu Fall Symp. 1997. P. 51–55.

121. *Sweeney L.* Privacy Technologies for Homeland Security // Testimony before the Privacy and Integrity Advisory Committee of the Department of Homeland Security, Boston, MA, June 15. 2005.

122. *Malin B., Sweeney L.* Detrmining the identifiability of DNA database entries // Proc. AMIA Symp. 2000. P. 537–541.

123. *Malin B.* Protecting DNA Sequence Anonymity with Generalization Lattices // Methods of Information in Medicine. 2005. V. 44. № 5. P. 687–692.

124. *Hodson H.* Revealed: Google AI has access to huge haul of NHS patient data // New Scientist, 29 Apr 2016.

125. *Cadwalladr C., Graham-Harrison E.* Revealed: 50 million facebook profiles harvested for Cambridge Analytica in major data breach // The Guardian, 17 Mar 2018.

126. Harmon A. Indian tribe wins fight to limit research of its DNA // New York Times. 2010, April, 22.

127. *Meyer M.* Law, Ethics & Science of Re-identification Demonstrations // Bill of Health: Examining the Intersection of Health Law, Biotechnology and Bioethics, Petrie Flom Center at Harvard University. 2021.

128. *Ohm P.* Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization // UCLA Law Review. 2010. V. 57. P. 1700–1777.

129. *de Montjoye Y.-A., Radaelli L., Singh V.K., Pentland A.* Unique in the shopping mall: on the reidentifiability of credit card metadata // Science. 2015. V. 347. P. 536–539.

130. *Golle P.* Revisiting the uniqueness of simple demographics in the U.S. population // Workshop on privacy in the electronic society, New York, Association for Compute Machinery. 2006.

131. *Rocher L., Hendrickx J.M., de Montjoye Y.-A.* Estimating the success of re-identifications in incomplete datasets using generative models // Nat. Commun.. 2019. V. 10. № 1 (3069).

132. *Culnane C., Rubinstein B.I.P., Teague V.* Health data in an open world // Preprint at: https://arxiv.org/abs/1712.05627. 2017.

133. *Siddle J.* I know where you were last summer: London's public bike data is telling everyone where you've been // vartree.blogspot.com. 2014.

134. *Lavrenovs A., Podins K.* Privacy violations in Riga open data public transport system // 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Vilnius, Lithuania. 2016. P. 1–6.

135. *Narayanan A., Shmatikov V.* Robust De-anonymization of Large Sparse Datasets // IEEE Symposium on Security and Privacy. 2008. P. 111–125.