Пакет *BIBasis* для вычисления булевых инволютивных базисов и базисов Грёбнера в системах компьютерной алгебры *REDUCE* и *Macaulay2*

М. В. Зинин
Лаборатория Информационных Технологий
Объединенный Институт Ядерных Исследований
141980 Дубна, Россия
mzinin@gmail.com

Аннотация

В данной работе для двух систем компьютерной алгебры – *REDUCE* и *Macaulay2* – представлен пакет *BIBasis*, позволяющий вычислять булевы инволютивные базисы и базисы Грёбнера. В соответствующих разделах описаны реализации и пользовательские интерфейсы пакета для каждой из систем. Также приводятся результаты сравнения пакета *BIBasis* с другими доступными в данных системах компьютерной алгебры пакетами и алгоритмами для построения булевых базисов Грёбнера.

1 Введение

Инволютивные базисы в кольце многочленов [1] в общем случае являются избыточными базисами Грёбнера и обладают некоторыми полезными свойствами [2] по сравнению с редуцированными базисами Грёбнера. Широко используемые инволютивные базисы вычисляются с помощью различных вариаций инволютивного алгоритма [3], который также предоставляет эффективный способ построения редуцированного базиса Грёбнера. Искомый редуцированный базис Грёбнера легко получается из известного инволютивного базиса без каких-либо дополнительных редукций, поскольку является его строго определенным подмножеством. Все вышесказанное справедливо не только в общем случае колец коммутативных многочленов, но и в частном случае булевых колец.

Булевы базисы Грёбнера уже доказали свою практическую применимость. Первой впечатляющей демонстрацией возможностей булевых базисов Грёбнера стал "взлом" первого же вызова (challenge) для криптосистемы HFE (Hidden Fields Equations) с открытым ключом, описанный в [4] и представляющий собой вычисление булева базиса Грёбнера для системы квадратичных булевых многочленов от 80 переменных. С тех пор область применения булевых базисов Грёбнера значительно расширилась, и сейчас есть уже ряд примеров [5] успешного применения этих базисов к решению задач булевой выполнимости SAT (Boolean Satisfiability).

В последние годы разработаны [6, 7, 8], реализованы и использованы для вычисления булевых базисов Грёбнера две версии булева инволютивного алгоритма, основанные на делениях Жане и Поммаре. Практические исследования реализаций этих версий на языке

C++ выявили превосходство алгоритма, основанного на делении Поммаре, — в отличие от инволютивного алгоритма в кольце многочленов с рациональными коэффициентами, где деление Жане предпочтительнее, а использование деления Поммаре может привести к построению бесконечного базиса. Результатом этих работ и исследований стал пакет *BIBasis* (Boolean Involutive Basis), разработанный для двух открытых систем компьютерной алгебры — *REDUCE* [9] и *Macaulay2* [10]. Пакет реализует инволютивный алгоритм на основе деления Поммаре в булевом кольце многочленов от многих переменных и позволяет вычислять как инволютивные базисы, так и базисы Грёбнера.

В разделе 2 данной работы описаны основные особенности булева кольца, имеющие отношение к вычислению базисов Грёбнера, даны определение деления Поммаре и общий вид инволютивного алгоритма, а также приводятся краткие доводы в пользу применения инволютивного алгоритма и деления Поммаре в булевом кольце и против использования алгоритма Бухбергера. Разделы 3 и 4 посвящены описанию внутреннего устройства пакета *BIBasis* и его пользовательского интерфейса в каждой из упомянутых систем. В этих же разделах приведены примеры использования пакета. В разделе 5 представлены результаты сравнения *BIBasis* с другими доступными пакетами и алгоритмами, позволяющими получить булевы базисы Грёбнера. И в заключении делаются выводы о производительности пакета *BIBasis* и предположения о путях его дальнейшего развития.

2 Инволютивный алгоритм на основе деления Поммаре в булевом кольце

2.1 Булево кольцо

Булево кольцо есть коммутативное кольцо *булевых функций* от n переменных, т.е. отображений из $\{0,1\}^n$ на $\{0,1\}$. Пусть $\mathbf{X} := \{x_1, \ldots, x_n\}$ — множество переменных, $\mathbb{F}_2$ — конечное поле, состоящее из двух элементов $\{0,1\}$. Тогда булево кольцо может быть представлено в виде факторкольца

$$\mathbb{B}[\mathbf{X}] := \mathbb{F}_2[\mathbf{X}] \,/ < x_1^2 + x_1, \ldots, x_n^2 + x_n > \ .$$

Умножение в $\mathbb{B}[\mathbf{X}]$ *идемпотентно*, а сложение — *нильпотентно*:

$$\forall\, b \in \mathbb{B}[\mathbf{X}] \ : \ \ b^2 = b\,,\ b + b = 0\,.$$

Элементы $\mathbb{B}[\mathbf{X}]$ называются *булевыми многочленами* и представимы конечными суммами

$$\sum_j \prod_{x \in\, \Omega_j \subseteq \mathbf{X}} x$$

булевых мономов.

Каждый моном представляет собой произведение переменных. Если множество Ω пусто, то соответствующий моном является единичной булевой функцией 1. Многочлен, состоящий из 0 мономов, является нулевой булевой функцией 0.

Биномы $\varphi_i := x_i^2 + x_i \in \mathbb{F}_2[\mathbf{X}]$ $(1 \le i \le n)$ называются *полевыми многочленами* [11, 5], поскольку равенства $\varphi_i = 0$ определяют ограничения на $x_i \in \mathbb{F}_2$.

2.2 Деление Поммаре и инволютивный алгоритм

Как и всякое инволютивное деление, деление Поммаре определяется разбиением переменных на мультипликативные и немультипликативные для каждого многочлена $f \in F \subset \mathbb{F}_2[\mathbf{X}]$, где множество F предполагается конечным. Это разбиение переменных проводится для множества старших мономов $U := \mathrm{lm}(F)$ и фиксированного допустимого мономиального упорядочения $\succ$ [12]. Через $M_{\mathcal{P}}(f)$ обозначим множество мультипликативных по Поммаре переменных многочлена $f \in F$, через $NM_{\mathcal{P}}(f) := \{x_1, \ldots, x_n\} \setminus M_{\mathcal{P}}(f)$ — множество немультипликативных переменных.

Произвольное $\mathcal{L}-$разбиение переменных порождает соответствующее деление следующим образом:

Определение 1. *[1]. (Старший) моном $u \in \mathrm{lm}(F)$ является $\mathcal{L}-$делителем монома w, если существует моном v такой, что $w = u \cdot v$, и v есть либо 1, либо произведение $\mathcal{L}-$мультипликативных переменных u. Обозначается $u \mid_{\mathcal{L}} w$.*

Общее определение деления Поммаре выглядит так:

Определение 2. *[1]. Для (старшего) монома $v = x_1^{d_1} \cdots x_k^{d_k}$, где $d_k > 0$ $(1 \leq k \leq n)$, переменные $x_k, \ldots, x_n$ являются $\mathcal{P}-$мультипликативными.*

Для произвольного инволютивного деления $\mathcal{L}$ соответствующая $\mathcal{L}$-редукция многочлена p по модулю многочлена $f \in F$ определяется аналогично редукции Грёбнера [12]. Единственное отличие состоит в том, что умножение f на соответствующий моном разрешено, только когда моном представляет собой произведение $\mathcal{L}-$ мультипликативных переменных f [1].

Обозначим через $NF_{\mathcal{L}}(p, F)$ $\mathcal{L}-$нормальную форму p по модулю F и определим инволютивный базис идеала $\mathcal{I} \subset \mathbb{F}_2[\mathbf{X}]$:

Определение 3. *[1]. Для данных идеала $\mathcal{I}$, инволютивного деления $\mathcal{L}$ и допустимого мономиального упорядочения $\succ$, конечное множество G, порождающее $\mathcal{I}$ (обозначается $\mathcal{I} =< G >$), называется $\mathcal{L}-$базисом, если*

$$\forall\, g \in G\,, \forall x \in NM_{\mathcal{L}}(g, G) \;\mid\; NF_{\mathcal{L}}(g \cdot x, G) = 0\,.$$

Подробное описание инволютивного алгоритма приведено в [3]. Здесь же приведем простейшую форму этого алгоритма, который для произвольного деления $\mathcal{L}$ и заданных $\succ$ и конечного $F \subset \mathbb{F}_2[\mathbf{X}]$ выдает $\mathcal{L}-$базис идеала $\mathcal{I} =< F >$.

Алгоритм: $\mathcal{L}-$**basis** $(F, \succ)$

Input: $F \subset \mathbb{F}_2[\mathbf{X}]$, конечное множество;
$\mathcal{L}$, инволютивное деление;
$\succ$, мономиальное упорядочение.
Output: G, $\mathcal{L}$-базис $< F >$.

1: **выбрать** $f \in F$ с наименьшим $\mathrm{lm}(f)$ относительно $\succ$
2: $G := \{f\}$;
3: $Q := F \setminus \{f\} \cup \{f \cdot x \mid x \in NM_{\mathcal{L}}(f, F)\}$;
4: **while** $Q \neq \emptyset$ **do**
5: $h := 0$;
6: **while** $Q \neq \emptyset$ и $h = 0$ **do**
7: **выбрать** $q \in Q$ с наименьшим $\mathrm{lm}(q)$ относительно $\succ$;
8: $Q := Q \setminus \{q\}$;
9: $h := NF_{\mathcal{L}}(q, G)$;
10: **end while**
11: **if** $h \neq 0$ **then**
12: **for all** $\{p \in G \mid \mathrm{lm}(h) \mid_{\mathcal{L}} \mathrm{lm}(p)\}$ **do**
13: $G := G \setminus \{p\}$;
14: $Q := Q \cup \{p\} \setminus \{p \cdot x \mid x \in NM_{\mathcal{L}}(p, G)\}$;
15: **end for**
16: $G := G \cup \{h\}$;
17: $Q := Q \cup \{h \cdot x \mid x \in NM_{\mathcal{L}}(h, G)\}$;
18: **end if**
19: **end while**
20: **return** G;

Известно, что результат работы как инволютивного, так и алгоритма Бухбергера зависит от выбранного мономиального упорядочения. И это упорядочение должно быть *допустимым* [3], т.е.

$$m \neq 1 \Longleftrightarrow m \succ 1 \quad \forall\, m,$$
$$m_1 \succ m_2 \Longleftrightarrow m_1 m \succ m_2 m \quad \forall\, m, m_1, m_2.$$

Но, как легко проверить, эти два условия в булевом кольце не совместимы:

$$x_1 \succ x_2 \quad \xrightarrow{*x_1} \quad x_1 * x_1 \succ x_1 * x_2 \quad \longrightarrow \quad x_1 \succ x_1 x_2$$

К тому же, хотя $\mathbb{B}[\mathbf{X}]$ и является кольцом главных идеалов, множество, состоящее из одного булева многочлена $\{p\}$, не обязательно является базисом Грёбнера идеала $< p >$, порожденного этим многочленом. Например:

$$x_1, x_2 \in < x_1 x_2 + x_1 + x_2 > \subset \mathbb{B}[x_1, x_2].$$

Поэтому алгоритм Бухбергера неприменим непосредственно в булевом кольце, а чтобы получить с его помощью булев базис Грёбнера, приходится выполнять все вычисления в кольце $\mathbb{F}_2[\mathbf{X}]$ и использовать *полевые биномы* $x_1^2 + x_1, \ldots, x_n^2 + x_n$.

Инволютивный алгоритм, основанный на делении Жане, обладает этими же недостатками в отличие от инволютивного алгоритма на основе деления Поммаре [6]. Последний можно применять непосредственно в булевом кольце, что в свою очередь допускает использование эффективных структур данных для внутреннего представления мономов.

3 Пакет *BIBasis* в системе компьютерной алгебры *REDUCE*

Система компьютерной алгебры *REDUCE* является интерактивной системой общего назначения для математиков, ученых и инженеров. В настоящее время ее возможности в области символьных вычислений включают в себя [9]:

- Все основные символьные алгоритмические операции над многочленами, рациональными функциями и отрезками степенных рядов;
- Автоматическое и управляемое пользователем упрощение выражений;
- Вычисления с символьными матрицами;
- Аналитические дифференцирование и интегрирование;
- Решение достаточно широкого класса алгебраических, трансцендентных и дифференциальных уравнений;
- Вычисления с основными специальными функциями;
- Построения базисов Грёбнера в коммутативной алгебре;
- Оптимизация программ для численных расчетов с символьными входными данными.

REDUCE полностью написана на своем собственном диалекте языка LISP, называемом Standard LISP. Для большинства операционных систем (Unix, Linux, Microsoft Windows, Apple Macintosh) *REDUCE* доступна в двух версиях, отличающихся реализациями используемого диалекта языка LISP — Portable Standard LISP [13] (PSL) и Codemist Standard LISP [14] (CSL). Отметим, что PSL версия является более быстрой в большинстве случаев, а потому чаще используемой.

С декабря 2008 года *REDUCE* стала бесплатной системой компьютерной алгебры с открытым исходным кодом, ее разрабатываемая ветка доступна по адресу [15].

REDUCE включает в себя множество разработанных ее пользователями дополнительных пакетов, реализующих функции и алгоритмы в различных областях математики. Каждый пакет является вполне самостоятельной программой, его исходные коды находятся в отдельной директории вместе с документацией и тестами этого пакета. Краткое руководство по созданию пакетов для системы *REDUCE* можно найти в заголовке файла `/packages/package.map` в дереве исходных кодов.

Одним их таких пакетов является пакет *BIBasis*.

3.1 Реализация и представление данных

Для реализации пакета *BIBasis* в системе компьютерной алгебры *REDUCE* был выбран используемый в этой системе диалект языка LISP — Standard LISP.

В процессе работы инволютивного алгоритма наиболее часто используемыми, и, следовательно, отнимающими большую часть процессорного времени операциями являются операции над базовыми объектами — мономами и многочленами [8]. В пакете *BIBasis* для

системы *REDUCE*, как и в других реализациях инволютивного алгоритма, о которых идет речь в данной статье, используется плотное дистрибутивное представление как мономов, так и многочленов.

Для языка Standard LISP это означает, что моном представляет собой упорядоченный список произвольной длины. Элементами этого списка — типа `integer` — являются номера переменных, входящих в моном в степени 1. Переменные, входящие в моном в степени 0, в формировании списка, представляющего моном, не участвуют. Других степеней, кроме 0 и 1, переменные в булевом кольце иметь не могут, что очень удобно с вычислительной точки зрения, т.к. не приходится тратить память на хранение показателя степени каждой переменной. Для удобства итерации по моному, каждый такой список завершается двумя элементами NIL.

Аналогичное представление имеют и многочлены: каждый многочлен есть упорядоченный список, завершающийся двумя NIL. Элементами списка являются только мономы, входящие в многочлен с коэффициентом 1. Мономы с коэффициентом 0 в список не входят, а других коэффициентов в булевом кольце нет. Это не только позволяет нам экономить память на хранении коэффициентов мономов, но и ускоряет работу инволютивного алгоритма в целом, поскольку в случае полной арифметики на вычисление коэффициентов тратится значительная часть процессорных ресурсов.

Как будет видно из следующего подраздела, пакет *BIBasis* позволяет выбрать мономиальное упорядочение в процессе выполнения программы. Функциональный язык программирования Standard LISP не предоставляет простора для реализации такого выбора: функция сравнения двух мономов в зависимости от выбранного упорядочения (т.е. в зависимости от значения соответствующей глобальной переменной) вызывает специализированную для данного упорядочения функцию сравнения. Вероятно (см. подраздел 4.2), такая реализация не самая быстрая из всех принципиально возможных, но этот недостаток вполне компенсируется другими преимуществами инволютивного алгоритма, деления Поммаре и булева кольца, что видно из графиков в разделе 5.

3.2 Пользовательский интерфейс

Текущая версия пакета *BIBasis* предоставляет пользователям только 2 функции: `bibasis` и `bibasis_print_statistics`.

Функция `bibasis` вычисляет булев инволютивный базис или базис Грёбнера по заданному базису и имеет следующий синтаксис:

```
bibasis(initial_polynomial_list,
        variables_list,
        monomial_ordering,
        reduce_to_groebner);
```

Параметры:

- `initial_polynomial_list`
  Список многочленов, составляющих известный базис исходного булева идеала. Данные многочлены считаются булевыми независимо от того, какому кольцу они принадлежат. См. Пример 3.3.1.

- `variables_list`
  Список независимых переменных в порядке убывания старшинства.

- `monomial_ordering`
  Выбранное мономиальное упорядочение. Поддерживаются следующие значения:

`lex`	чисто лексикографическое упорядочение
`deglex`	градуированное лексикографическое упорядочение
`degrevlex`	градуированное обратное лексикографическое упорядочение

  См. Примеры 3.3.2—3.3.4, иллюстрирующие зависимость редуцированного базиса Грёбнера от мономиального упорядочения.

- `reduce_to_groebner`
  Булева величина, если этот параметр имеет значение `t`, то функция вернет редуцированный булев базис Грёбнера, если `nil`, то результатом будет редуцированный булев базис Поммаре. На Примерах 3.3.5 и 3.3.6 показана разница между этими двумя базисами.

Возвращаемое значение:

- Список многочленов, составляющих редуцированный булев базис Грёбнера или Поммаре.

Функция `bibasis_print_statistics` выводит на печать краткую статистику для предыдущего вызова функции `bibasis`. Имеет следующий синтаксис:

```
bibasis_print_statistics();
```

Эта функция не принимает никаких параметров и ничего не возвращает. См. Пример 3.3.7.

3.3 Примеры

Пример 3.3.1:

```
1: load_package bibasis;
2: bibasis({x+2*y}, {x,y}, lex, t);
{x}
```

Пример 3.3.2:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, lex, t);
{x0 + x2*x4,x2*(x1 + x3*x4)}
```

Пример 3.3.3:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, deglex,
   t);
{x1*x2*(x3 + 1),
 x1*(x0 + x2),
 x0*(x2 + 1),
 x0*x3 + x1*x2,
 x0*(x4 + 1),
 x2*x4 + x0}
```

Пример 3.3.4:

```
1: load_package bibasis;
2: variables :={x0,x1,x2,x3,x4}$
3: polynomials := {x0*x3+x1*x2,x2*x4+x0}$
4: bibasis(polynomials, variables, degrevlex,
   t);
{x0*(x1 + x3),
 x0*(x2 + 1),
 x1*x2 + x0*x3,
 x0*(x4 + 1),
 x2*x4 + x0}
```

Пример 3.3.5:

```
1: load_package bibasis;
2: variables :={x,y,z}$
3: polinomials := {x, z}$
4: bibasis(polinomials, variables, degrevlex,
   t);
{x,z}
```

Пример 3.3.6:

```
1: load_package bibasis;
2: variables :={x,y,z}$
3: polinomials := {x, z}$
4: bibasis(polinomials, variables, degrevlex,
   nil);
{x,z,y*z}
```

Пример 3.3.7:

```
1: load_package bibasis;
2: variables := {u0,u1,u2,u3,u4,u5}$
3: polynomials := {u0+u1+u2+u3+u4+u5+1,
3:         u0*u1+u1+u1*u2+u2*u3+u3*u4+u4*u5,
3:         u0*u2+u2+u1*u3+u2*u4+u3*u5+u1,
3:         u0*u3+u3+u1*u4+u2*u5+u1*u2,
3:         u0*u4+u4+u1*u5+u1*u3+u2}$
4: bibasis(polynomials, variables, degrevlex,
```

```
   t);
{u2*u4,
 u2*(u5 + 1),
 u4*u5,
 u0 + u4 + u5 + 1,
 u1 + u2 + u4,
 u3 + u4}
5: bibasis_print_statistics();
Variables order = u0 > u1 > u2 > u3 > u4 > u5
Normal forms calculated = 48
  Non-zero normal forms = 16
        Reductions made = 276
Time: 0 ms
GC time: 0 ms
```

4 Пакет *BIBasis* в системе компьютерной алгебры *Macaulay2*

Цель системы компьютерной алгебры *Macaulay2* — помощь и поддержка исследований в области алгебраической геометрии, коммутативной алгебры и их приложений [10]. *Macaulay2* позволяет производить эффективные вычисления с разнообразными математическими объектами, такими, как [16] :

- поля Галуа,
- кольца многочленов,
- алгебры Вейля,
- кольца вычетов,
- идеалы,
- гомоморфизмы колец,
- и другие.

Пользователям *Macaulay2* предоставляет интерактивный интерфейс и свой собственный мощный, но простой в использовании язык программирования, синтаксис которого максимально приближен к стандартной алгебраической системе записи. Но для большей эффективности основополагающие алгоритмы написаны на языке C++ и скомпилированы, т.е. не интерпретируются при их вызове пользователем [16]. К таким алгоритмам относятся арифметические операции над кольцами, модулями и матрицами, алгоритм вычисления базиса Грёбнера и функции Гильберта, разложение на множители и прочие.

Изначально, с 1993 года, *Macaulay2* является свободной программой с открытым исходный кодом и распространяется под лицензией GNU General Public License версии 2. Разрабатываемая ветка доступна по адресу [17].

Как и система *REDUCE*, *Macaulay2* включает в себя множество дополнительных пакетов. И хотя добавить пакет в эту систему несколько сложнее, чем в ту же *REDUCE*, в *Macaulay2* есть гораздо более подробная инструкция с примером, которую можно найти в файле `/Macaulay2/packages/PackageTemplate.m2` в дереве исходных кодов. Почти все пакеты обладают собственными документацией и набором тестов, в том числе и *BIBasis*.

4.1 Реализация и представление данных

Пакет *BIBasis* в системе компьютерной алгебры *Macaulay2* реализован на языке C++, как и все ядро системы *Macaulay2*.

Как и в пакете *BIBasis* для *REDUCE*, здесь мономы и многочлены имеют плотное дистрибутивное представление. Моном — упорядоченный односвязный список, его элементы — номера переменных, входящих в моном в степени 1 — имеют тип `short int`, что позволяет работать с кольцами булевых многочленов от 32767 переменных. Такого числа переменных вполне достаточно для любых задач SAT, решаемых за приемлемое время, и оно легко может быть увеличено в случае необходимости.

Аналогично и многочлены представляют собой упорядоченные односвязные списки мономов, входящих в соответствующие многочлены с коэффициентом 1.

Объектно-ориентированный язык программирования C++ предоставляет множество способов решения той или иной задачи, в том числе и задачи выбора мономиального упорядочения для построения инволютивного базиса во время выполнения программы. В процессе разработки пакета *BIBasis* для системы *Macaulay2* были опробованы следующие варианты:

- Имеется единственный класс мономов, выбор мономиального упорядочения проявляется только в функциях- членах класса, отвечающих за сравнение мономов. Т.е. это решение аналогично тому, что используется в пакете *BIBasis* для системы *REDUCE*.

- Имеется базовый абстрактный класс мономов и несколько его потомков (по числу поддерживаемых мономиальных упорядочений). Каждый класс мономов-потомков имеет свой набор функций-членов этого класса для сравнения мономов этого же класса. Экземпляры классов мономов создаются с помощью фабрики объектов, которая, зная выбранное упорядочение, вызывает конструктор соответствующего класса-потомка.

- Так же имеется базовый абстрактный класс мономов и его потомки. Но экземпляры классов-потомков создаются непосредственно вызовами конструкторов, а все прочие классы, использующие классы мономов, являются шаблонными. Параметрами шаблонов таких классов являются типы мономов, соответствующие выбранному мономиальному упорядочению. Таким образом, зависимость от упорядочения проявляется лишь один раз в начале работы программы — при выборе специализации шаблона класса, зависящего от типа монома.

Практика показала, что лучшим с точки зрения скорости работы алгоритма является последний вариант, именно он сейчас реализован в пакете *BIBasis* для системы *Macaulay2*.

Еще одним преимуществом использования языка C++ является возможность использования специализированный менеджеров памяти для отдельных классов. В пакете *BIBasis* такими менеджерами обладают классы мономов и многочленов, что в сочетании с использованием во всей системе *Macaulay2* своего аллокатора памяти вместо стандартного позволяет заметно ускорить вычисление булевых базисов.

4.2 Пользовательский интерфейс

В системе *Macaulay2* пакет *BIBasis* предоставляет пользователям только одну функцию — `biBasis` (*Macaulay2*, в отличие от *REDUCE*, чувствительна к регистру команд). Как и

аналогичная функция в версии для системы *REDUCE*, `biBasis` вычисляет требуемый булев базис по заданному своим базисом идеалу:

```
biBasis(ideal,
        toGroebner)
```

Параметры:

- `ideal`
  Исходный булев идеал, заданный неким своим базисом. Многочлены его базиса считаются булевыми независимо от того, какому кольцу они принадлежат. См. Пример 4.3.1.

- `toGroebner`
  Булева величина, если этот параметр имеет значение `true`, то функция вернет редуцированный булев базис Грёбнера, если `false`, то результатом будет редуцированный булев базис Поммаре. Значением по умолчанию является `true`. См. Пример 4.3.5.

Возвращаемое значение:

- Булев идеал, совпадающий с исходным, представленный своим инволютивным базисом или базисом Грёбнера.

Выбор мономиального упорядочения происходит при задании кольца, в котором определяется исходный идеал, передаваемый функции `biBasis` в качестве ее первого параметра. Как и в версии для системы *REDUCE* поддерживаются 3 упорядочения:

`Lex`	чисто лексикографическое упорядочение
`GLex`	градуированное лексикографическое упорядочение
`GRevLex`	градуированное обратное лексикографическое упорядочение

См. Примеры 4.3.2—4.3.4. При выборе какого-либо другого мономиального упорядочения функция `biBasis` вызовет ошибку.

4.3 Примеры

Пример 4.3.1:

```
i1 : loadPackage "BIBasis";
i2 : R = ZZ/2[x, y, MonomialOrder => GRevLex];
i3 : I = ideal(x + 2*y);
o3 : Ideal of R
i4 : biBasis(I)
o4 = ideal(x)
o4 : Ideal of R
```

Пример 4.3.2:

```
i1 : loadPackage "BIBasis";
i2 : R = ZZ/2[x0, x1, x2, x3, x4,
              MonomialOrder => Lex];
i3 : I = ideal(x0*x3 + x1*x2, x2*x4 + x0);
o3 : Ideal of R
i4 : biBasis(I)
o4 = ideal (x0 + x2*x4, x1*x2 + x2*x3*x4)
o4 : Ideal of R
```

Пример 4.3.3:

```
i1 : loadPackage "BIBasis";
i2 : R = ZZ/2[x0, x1, x2, x3, x4,
              MonomialOrder => GLex];
i3 : I = ideal(x0*x3 + x1*x2, x2*x4 + x0);
o3 : Ideal of R
i4 : biBasis(I)
o4 = ideal (x1*x2*x3 + x1*x2, x0*x1 + x1*x2,
            x0*x2 + x0, x0*x3 + x1*x2,
            x0*x4 + x0, x2*x4 + x0)
o4 : Ideal of R
```

Пример 4.3.4:

```
i1 : loadPackage "BIBasis";
i2 : R = ZZ/2[x0, x1, x2, x3, x4,
              MonomialOrder => GRevLex];
i3 : I = ideal(x0*x3+x1*x2, x2*x4+x0);
o3 : Ideal of R
i4 : biBasis(I)
o4 = ideal (x0*x1 + x0*x3, x0*x2 + x0,
            x1*x2 + x0*x3, x0*x4 + x0,
            x2*x4 + x0)
o4 : Ideal of R
```

Пример 4.3.5:

```
i1 : loadPackage "BIBasis";
i2 : R = ZZ/2[x, y, z,
              MonomialOrder => GRevLex];
i3 : I = ideal(x, z);
o3 : Ideal of R
i4 : G = biBasis(I, toGroebner => true)
o4 = ideal (x, z)
o4 : Ideal of R
i5 : P = biBasis(I, toGroebner => false)
o5 = ideal (x, z, y*z)
o5 : Ideal of R
```

5 Результаты тестов

На графиках ниже приведены времена счета (в секундах) для нескольких серий тестовых примеров, полученные на машине 2xXeon 5410 (4 ядра) 2.33Ghz с 16Gb RAM под управлением ОС Gentoo Linux. Обе рассматриваемые системы компьютерной алгебры, *REDUCE* и *Macaulay2*, были собраны из исходных кодов своих разрабатываемых веток([15] и [17] соответственно) с помощью набора компиляторов gcc 4.4.3. Времена счета были ограничены 6 часами, доступная оперативная память — 10 гигабайтами. Из всех рассмотренных серий примеров только *ezfact* вызвала переполнение выделенной памяти, все остальные вычисления прерывались, когда заканчивалось отведенное на них время.

В качестве тестовых примеров были выбраны стандартные серии примеров [18], не содержащие переменных в степени выше 1: *cyclic, eco, redcyc(lic), redeco.*В дополнение к этим известным сериям еще одна была взята из работы [19]. Она была названа *life*, поскольку была получена из анализа известной игры «Жизнь» Дж. Конуэя. Каждый пример из серии *life* состоит из одного многочлена от переменных $x_0, \ldots, x_i$ следующего вида:

$$x_i + x_{i-1}(\sigma_{i-2} + \sigma_{i-3} + \sigma_3 + \sigma_2) + \sigma_{i-2} + \sigma_3\,,$$

где σ_k — k-ый симметрический многочлен от $x_0, \ldots, x_{i-2}$.

Примеры SAT задач взяты из коллекции SAT-02 Challenge Set [20].

5.1 *REDUCE*, сравнение пакетов *BIBasis* и *Groebner*

В системе компьютерной алгебры *REDUCE* есть пакет *Groebner* для вычисления базисов Грёбнера. Он не рассчитан на работу в булевом кольце, поэтому, для того, чтобы получить с его помощью булев базис Грёбнера, исходный базис необходимо дополнить полевыми биномами.

Для сравнения использовалась PSL версия *REDUCE* как более быстрая по сравнению с CSL.

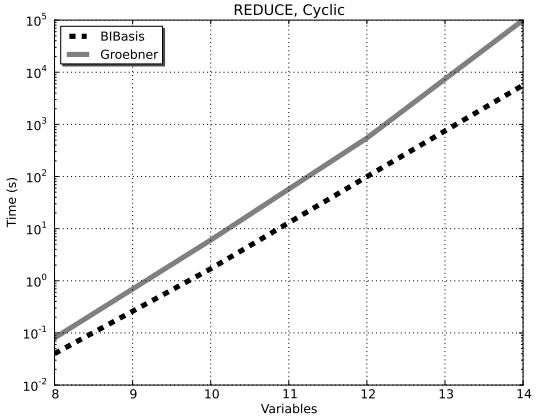

Рис. 1: Зависимость времени работы системы *REDUCE* от числа переменных для тестов *cyclic* DegRevLex

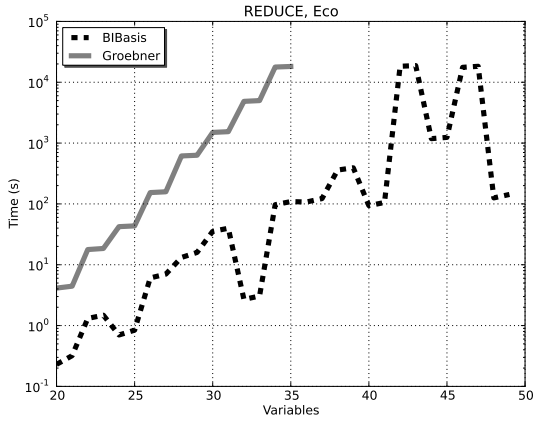

Рис. 2: Зависимость времени работы системы *REDUCE* от числа переменных для тестов *eco* DegRevLex

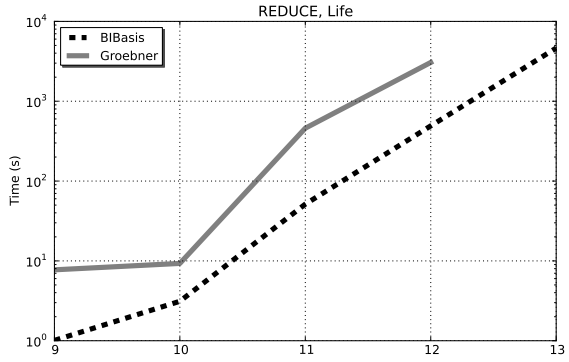

Рис. 3: Зависимость времени работы системы *REDUCE* от числа переменных для тестов *life* DegRevLex

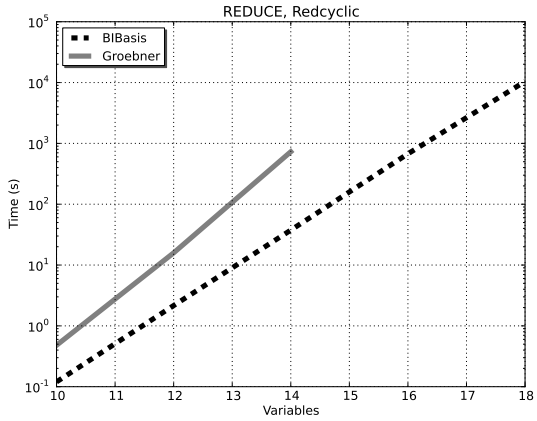

Рис. 4: Зависимость времени работы системы *REDUCE* от числа переменных для тестов *redcyclic* DegRevLex

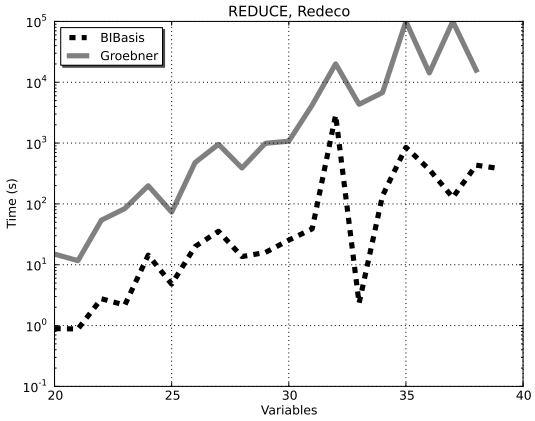

Рис. 5: Зависимость времени работы системы *REDUCE* от числа переменных для тестов *redeco* DegRevLex

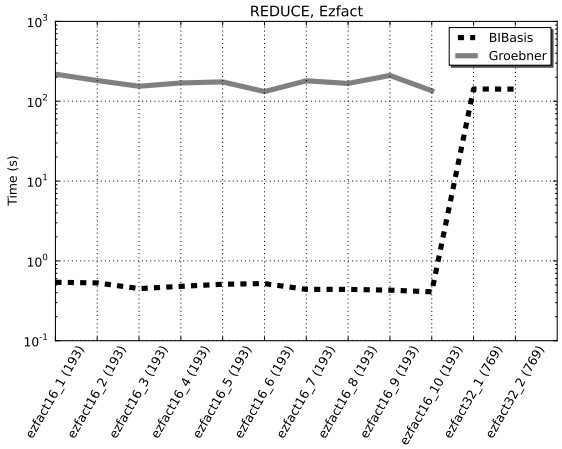

Рис. 6: Зависимость времени работы системы *REDUCE* для тестов *ezfact* DegRevLex. Число переменных указано в круглых скобках для каждого примера

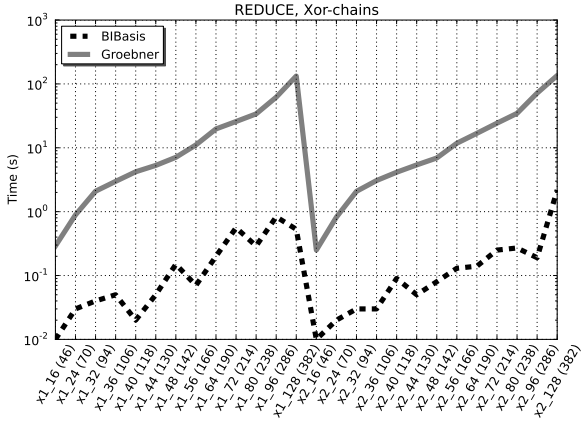

Рис. 7: Зависимость времени работы системы *REDUCE* для тестов *xor-chains* DegRevLex. Число переменных указано в круглых скобках для каждого примера

5.2 *Macaulay2*, сравнение пакета *BIBasis* и стандартного алгоритма *gb*

В системе компьютерной алгебры *Macaulay2* для вычисления базисов Грёбнера используется стандартный алгоритм *gb*. Для получения с его помощью булева базиса Грёбнера необходимо добавить к исходному базису полевые биномы.

В сравнении не участвует пакет *BooleanGB* [21], доступный в *Macaulay2*, поскольку данный пакет позволяет вычислять булевы базисы Грёбнера только для чисто лексикографического упорядочения и только в кольце булевых многочленов не более, чем от 64 переменных, что недостаточно для большинства SAT задач.

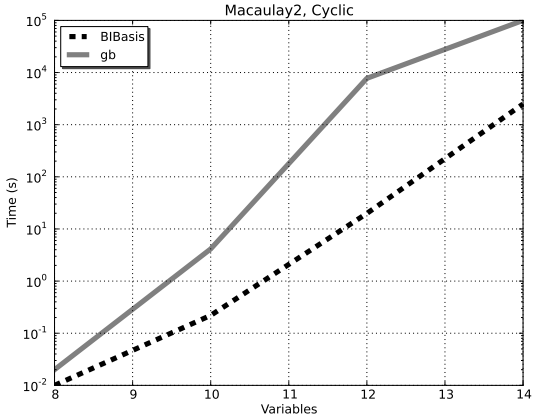

Рис. 8: Зависимость времени работы системы *Macaulay2* от числа переменных для тестов *cyclic* DegRevLex

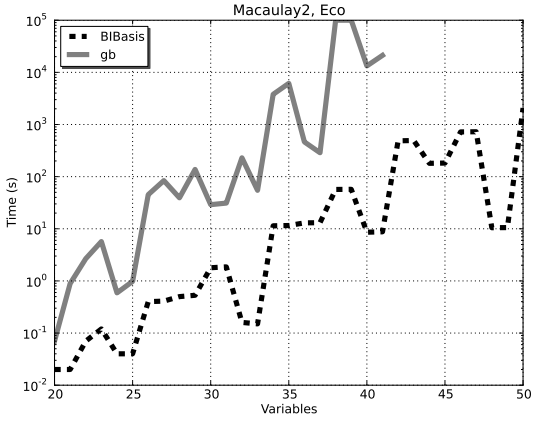

Рис. 9: Зависимость времени работы системы *Macaulay2* от числа переменных для тестов *eco* DegRevLex

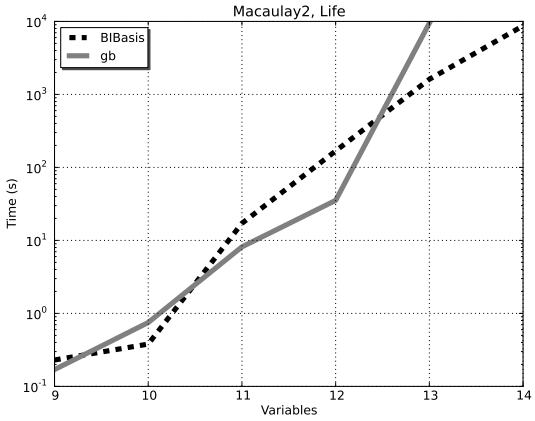

Рис. 10: Зависимость времени работы системы *Macaulay2* от числа переменных для тестов *life* DegRevLex

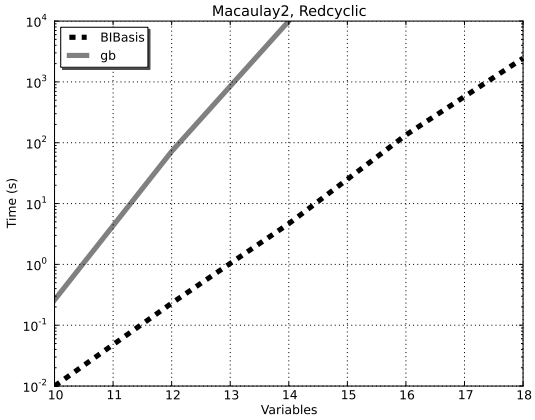

Рис. 11: Зависимость времени работы системы *Macaulay2* от числа переменных для тестов *redcyclic* DegRevLex

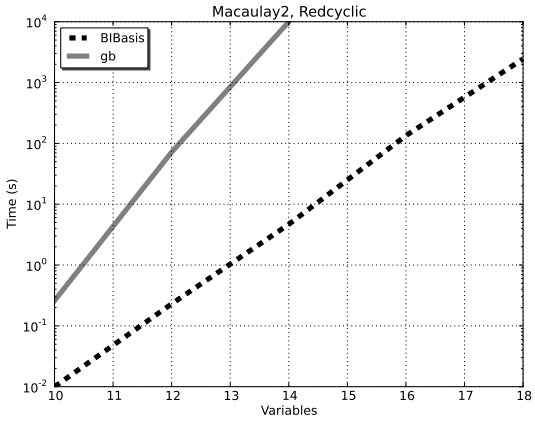

Рис. 12: Зависимость времени работы системы *Macaulay2* от числа переменных для тестов *redeco* DegRevLex

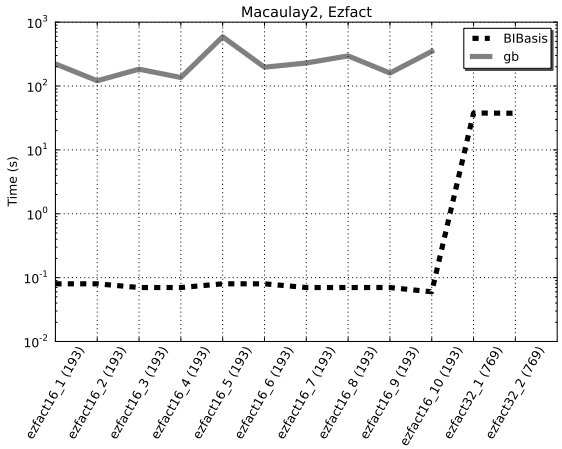

Рис. 13: Зависимость времени работы системы *Macaulay2* для тестов *ezfact* DegRevLex. Число переменных указано в круглых скобках для каждого примера

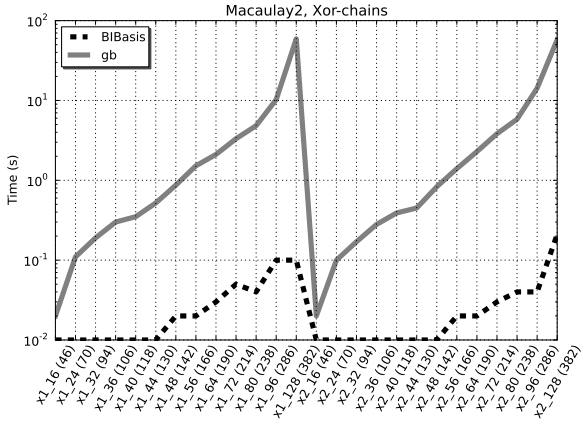

Рис. 14: Зависимость времени работы системы *Macaulay2* для тестов *xor-chains* DegRevLex. Число переменных указано в круглых скобках для каждого примера

6 Заключение

Из приведенных графиков видно, что пакет *BIBasis* на большинстве рассмотренных примеров быстрее стандартных реализаций алгоритма Бухбергера в системах компьютерной алгебры *REDUCE* и *Macaulay2*. Также отметим, что очень немногие из нескольких сотен удачных и неудачных попыток вычислить булев базис Грёбнера заканчивались нехваткой оперативной памяти, как для пакета *BIBasis*, так и для реализаций алгоритма Бухбергера. Таким образом, можно сказать, что преимущество в скорости пакета *BIBasis* достигается при сравнимых (а иногда и заметно более слабых) требованиях к объему оперативной памяти, и что в целом для вычисления булевых базисов Грёбнера более значимым параметров является производительность процессора, а не доступная оперативная память.

Одним из наиболее вероятных путей дальнейшего развития как рассмотренной реализации булева инволютивного алгоритма в целом, так и пакета *BIBasis* в частности станет использование многопоточной модели в их архитектуре. Это позволит задействовать возможности нескольких ядер/процессоров одновременно и таким образом ускорить вычисления требуемых булевых базисов практически на каждом современной компьютере.

Исходные коды пакета *BIBasis* доступны в разрабатываемых ветках соответствующих систем, а именно:

- для системы *REDUCE* — [15], относительный путь `/packages/bibasis/`,
- для системы *Macaulay2* — [17], относительный путь `/Macaulay2/kernel/bibasis/`.

Автор выражает благодарность Юрию Блинкову за помощь в создании пакета *BIBasis* для системы *REDUCE*, Винфриду Нойну (Winfried Neun) и Дэниэлу Грейсону (Daniel Grayson) за согласие и помощь в размещении исходных кодов пакета *BIBasis* в разрабатываемых ветках систем компьютерной алгебры *REDUCE* и *Macaulay2* соответственно.

Список литературы

[1] V.P.Gerdt and Yu.A.Blinkov. *Involutive Bases of Polynomial Ideals*. Mathematics and Computers in Simulation, 45, pp.519–542, 1998; *Minimal Involutive Bases*, ibid. pp.543–560.

[2] W.M.Seiler. *Involution: The Formal Theory of Differential Equations and its Applications in Computer Algebra*. Algorithms and Computation in Mathematics, 24, Springer, 2010. arXiv:math.AC/0501111

[3] Vladimir P. Gerdt. *Involutive Algorithms for Computing Gröbner Bases*. Computational Commutative and Non-Commutative Algebraic Geometry. IOS Press, Amsterdam, 2005, pp.199–225.

[4] J.-C.Faugère and A.Joux. *Algebraic Cryptanalysis of Hidden Field Equations (HFE) using Gröbner Bases*. LNCS 2729, Springer-Verlag, 2003, pp.44–60.

[5] M.Brickenstein, A.Dreyer, G.-M.Greuel and O.Wienand. *New developments in the theory of Gröbner bases and applications to formal verification*. Journal Pure and Applied Algebra, 213, pp.1612–1635, 2009. arXiv:math.AC/0801.1177

[6] V.P.Gerdt and M.V.Zinin. *A Pommaret Division Algorithm for Computing Gröbner Bases in Boolean Rings*. Proceedings of ISSAC 2008, ACM Press, 2008, pp.95–102.

[7] В. П. Гердт, М. В. Зинин. *Инволютивный метод вычисления базисов Грёбнера над* F_2. Программирование, т. 34, в. 4, 2008, с. 191–203.

[8] В. П. Гердт, М. В. Зинин, Ю. А. Блинков. *О вычислении булевых инволютивных базисов*. Программирование, т. 36, в. 2, 2010, с. 117–128.

[9] Anthony C. Hearn. *REDUCE, a portable general-purpose computer algebra system*. `http://www.reduce-algebra.com/`, 2009.

[10] Grayson, Daniel R. and Stillman, Michael E. *Macaulay2, a software system for research in algebraic geometry*. `http://www.math.uiuc.edu/Macaulay2/`, 2011.

[11] M. Bardet, J.-C.Faugère and B.Salvy. *Complexity of Gröbner Basis computation for Semi-regular Overdetermined sequences over* $\mathbb{F}_2$ *with solutions in* $\mathbb{F}_2$. INRIA report RR-5049, 2003.

[12] T.Becker, V.Weispfenning and H.Kredel. *Gröbner Bases. A Computational Approach to Commutative Algebra*. Graduate Texts in Mathematics 141, Springer-Verlag, New York, 1993.

[13] Winfried Neun. *Portable Standard Lisp*. `http://www2.zib.de/Symbolik/reduce/`, 1999.

[14] Codemist Ltd. *Codemist Standard LISP*. `http://www.codemist.co.uk`, 2002.

[15] `https://reduce-algebra.svn.sourceforge.net/svnroot/reduce-algebra/trunk/`

[16] David Eisenbud, Daniel R. Grayson, Michael E. Stillman and Bernd Sturmfels. *Computations in algebraic geometry with Macaulay 2*. Algorithms and Computations in Mathematics, 8, Springer-Verlag, 2001. `http://www.math.uiuc.edu/Macaulay2/Book/`.

[17] `svn://svn.macaulay2.com/Macaulay2/trunk/M2/`

[18] D. Bini and B. Mourrain. *Polynomial test suite*. `http://www-sop.inria.fr/saga/POL`, 2006 Jan Verschelde. *The database of polynomial systems*. `http://www.math.uic.edu/~jan/demo.html`, 2011

[19] V.V.Kornyak. *On Compatibility of Discrete Relations*. LNCS 3718, Springer-Verlag, 2005, pp.272–284. arXiv:math-ph/0504048

[20] Holger H. Hoos and Thomas Stutzle. *SATLIB: An Online Resource for Research on SAT*. I.P.Gent, H.v.Maaren, T.Walsh, editors, SAT 2000, pp.283-292, IOS Press, 2000. SATLIB is available online at `http://www.satlib.org/`.

[21] Franziska Hinkelmann, Elizabeth Arnold. *Fast Gröbner Basis Computation for Boolean Polynomials*. arXiv:1011.6064v1