

УДК 512

# ПРОСТОЙ ИТЕРАТИВНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ БАЗИСОВ ГРЁБНЕРА, ОСНОВАННЫЙ НА СИГНАТУРАХ

В. В. Галкин<sup>1</sup>

Данная работа описывает алгоритм вычисления базисов Грёбнера, основанный на использовании отмеченных многочленов и идеях из алгоритма F5. Отличительными особенностями алгоритма по сравнению с аналогами являются простота самого алгоритма и доказательства его корректности, достигнутые без потери эффективности. Это позволяет создать простую реализацию, не уступающую более сложным аналогам по производительности

*Ключевые слова:* базис Грёбнера, алгоритм F5, отмеченные многочлены

This paper presents an algorithm for computing Groebner bases based upon labeled polynomials and ideas from the algorithm F5. The main highlights of this algorithm compared with analogues are simplicity both of the algorithm and of the its correctness proof achieved without loss of the efficiency. This leads to simple implementation which performance is in par with more complex analogues

*Key words:* Groebner basis, F5 algorithm, labeled polynomials

## Определения

Рассмотрим кольцо многочленов  $P = k[x_1, \dots, x_n]$  над полем  $k$ . Будем предполагать, что на моноиде его мономов  $\mathbb{T}$  задан допустимый мономиальный порядок  $<$ . В этом кольце может быть поставлена задача вычисления базиса Грёбнера для произвольного идеала  $(f_1, \dots, f_l)$ . Один из способов её решения инкрементальный: последовательно вычисляются базисы идеалов  $(f_1, \dots, f_i)$ ,  $i = 2 \dots l$  на основе уже вычисленного для идеала  $(f_1, \dots, f_{i-1})$  базиса  $R_{i-1}$  и многочлена  $f_i$ . Представляемый алгоритм позволяет выполнить шаг такого вычисления. Таким образом, входные данные для алгоритма – это некоторый многочлен  $f$  и множество многочленов, обозначаемое  $\{g_1, \dots, g_m\}$ , являющееся базисом Грёбнера идеала  $I_0 = (g_1, \dots, g_m)$ . В качестве результата своей работы алгоритм должен построить множество многочленов  $R$ , являющееся базисом Грёбнера идеала  $I = (g_1, \dots, g_m, f)$ . Поскольку случаи  $f = 0 \Rightarrow I = I_0$  и  $\exists i g_i \in k \Rightarrow I = P$  не представляют интереса, далее предполагается что  $f \neq 0, \forall i g_i \notin k$ .

Введём обозначения:  $\mathbb{T}_0 = \mathbb{T} \cup \{0\}$  – моноид мономов, расширенный нулём. Порядок  $<$  продолжается с сохранением вполне упорядоченности на  $\mathbb{T}_0$  как  $<_0$  определением  $\forall t \in \mathbb{T} t >_0 0$ . Понятие делимости мономов также расширяется на  $\mathbb{T}_0$ :  $t_1 | t_2 \stackrel{\text{def}}{=} \exists t_3 : t_1 t_3 = t_2$ . Для  $p \in P, p \neq 0$  старшие по  $<$  моном и коэффициент обозначим  $\text{HM}(p) \in \mathbb{T}$  и  $\text{HC}(p) \in k$ . Для нуля –  $\text{HM}(0) \stackrel{\text{def}}{=} 0 \in \mathbb{T}_0$ ,  $\text{HC}(0) \stackrel{\text{def}}{=} 0 \in k$ . За  $\text{LCM}(t_1, t_2) \in \mathbb{T}$  обозначим наименьшее общее кратное  $t_1, t_2 \in \mathbb{T}$ . Далее все определения даются для фиксированных  $I_0$  и  $f$ :

**Определение 1.** *Отмеченным многочленом* называется пара  $h = (\sigma, p) \in \mathbb{T}_0 \times P$ , удовлетворяющая условию корректности:  $\exists u \in P : \text{HM}(u) = \sigma, u f \equiv p \pmod{I_0}$ . На отмеченные многочлены распространяются определения старшего монома  $\text{HM}(h) \stackrel{\text{def}}{=} \text{HM}(p)$  и коэффициента  $\text{HC}(h) \stackrel{\text{def}}{=} \text{HC}(p)$ . Также определяются *сигнатура*  $\mathcal{S}(h) \stackrel{\text{def}}{=} \sigma$  и вводится обозначение многочлена – второго элемента пары:  $\text{poly}(h) \stackrel{\text{def}}{=} p$ . Множество отмеченных многочленов обозначается за  $H \subset \mathbb{T}_0 \times P$ . Тривиальными примерами отмеченных многочленов являются  $(1, f)$  и  $(0, g)$  для  $g \in I_0$ . Другим примером отмеченного многочлена является  $(\text{HM}(g), 0)$  для  $g \in I_0$ . Он корректен, поскольку в качестве  $u$  можно взять  $g$ .

**Лемма 2.** *Умножение для  $h \in H, t \in \mathbb{T}$ , заданное как  $th \stackrel{\text{def}}{=} (t\sigma, tp) \in H$ , корректно.*

**Доказательство.** Корректность определения проверяется явным нахождением  $u$  для  $h$ .

<sup>1</sup> Галкин Василий Витальевич – асп. каф. алгебры мех.-мат. ф-та МГУ, e-mail: galkin-vv@yandex.ru.

**Определение 3.** Если для некоторых  $h'_1, h_2 \in H, t \in \mathbb{T}$  выполняется  $\mathcal{S}(h'_1) \succ_0 \mathcal{S}(th_2)$ ,  $\text{HM}(h'_1) = \text{HM}(th_2) \neq 0$ , то возможна *редукция*  $h'_1$  по  $h_2$  с *сохранением сигнатуры*, дающая в результате многочлен  $h_1 \in H$ , равный:

$$h_1 = (\mathcal{S}(h'_1), \text{poly}(h'_1) + Kt \text{poly}(h_2)),$$

где коэффициент  $K \in k$  взят так, чтоб при сложении сократились старшие мономы и выполнилось  $\text{HM}(h_1) \prec_0 \text{HM}(h'_1)$ . По сути такая редукция представляет из себя обычную редукцию многочлена с сокращением старшего монома, дополненную требованием того, что сигнатура редуктора меньше сигнатуры редуцируемого. Корректность проверяется как и выше. Введём частичный порядок  $<_{\text{gvw}}$  на  $H$ :

$$h_1 = (\sigma_1, p_1) <_{\text{gvw}} h_2 = (\sigma_2, p_2) \stackrel{\text{def}}{\iff} \text{HM}(p_1)\sigma_2 \prec_0 \text{HM}(p_2)\sigma_1.$$

Элементы с нулевой сигнатурой и старшим мономом оказываются экстремумами:  $\forall \sigma_1, \sigma_2, p_1, p_2 \ (0, p_1) \not<_{\text{gvw}} (\sigma_2, p_2), (\sigma_1, 0) \not>_{\text{gvw}} (\sigma_2, p_2)$ .

**Лемма 4.** Пусть  $h_1, h_2 \in H, t \in \mathbb{T}$ . Тогда  $h_1 >_{\text{gvw}} h_2 \iff h_1 >_{\text{gvw}} th_2$ .

**Доказательство.** Выводится из того, что умножение на  $t$  одного из сравниваемых отмеченных многочленов приводит к умножению на  $t$  обеих частей в определении  $>_{\text{gvw}}$ .

**Лемма 5.** Пусть  $h_1, h_2 \in H, \text{HM}(h_1) | \text{HM}(h_2), \text{HM}(h_2) \neq 0$ . Тогда редукция  $h_2$  по  $h_1$  с сохранением сигнатуры возможна если и только если  $h_1 >_{\text{gvw}} h_2$ .

**Доказательство.** Следует из того что оба утверждения равносильны  $\mathcal{S}(h_2) \succ_0 \mathcal{S}(h_1) \frac{\text{HM}(h_2)}{\text{HM}(h_1)}$ .

**Лемма 6.** Пусть  $h_1 \in H$  – результат редукции  $h'_1$  с сохранением сигнатур по некоторому многочлену. Тогда  $h_1 <_{\text{gvw}} h'_1$ .

**Доказательство.** Следует из  $\mathcal{S}(h_1) = \mathcal{S}(h'_1)$  и уменьшения  $\text{HM}$  при редукции:  $\text{HM}(h_1) \prec_0 \text{HM}(h'_1)$ .

**Лемма 7.** Пусть  $h_1 <_{\text{gvw}} h_2$  отмеченные многочлены. Тогда  $\forall h_3 \in H, h_3 \neq (0, 0)$  выполняется хотя бы одно из двух неравенств:  $h_1 <_{\text{gvw}} h_3$  или  $h_3 <_{\text{gvw}} h_2$ .

**Доказательство.** Из условия леммы известно, что

$$\text{HM}(h_1) \mathcal{S}(h_2) \prec_0 \text{HM}(h_2) \mathcal{S}(h_1) \quad (1)$$

откуда  $\text{HM}(h_2) \neq 0, \mathcal{S}(h_1) \neq 0$ . Поэтому, если  $\text{HM}(h_3) = 0$ , имеем  $h_3 <_{\text{gvw}} h_2$ , а если  $\mathcal{S}(h_3) = 0$  – то  $h_1 <_{\text{gvw}} h_3$ . Иначе можно домножить неравенство (1) на ненулевой элемент  $\text{HM}(h_3) \mathcal{S}(h_3)$ :

$$\text{HM}(h_3) \mathcal{S}(h_3) \text{HM}(h_1) \mathcal{S}(h_2) \prec_0 \text{HM}(h_3) \mathcal{S}(h_3) \text{HM}(h_2) \mathcal{S}(h_1). \quad (2)$$

Поэтому  $\text{HM}(h_3)^2 \mathcal{S}(h_2) \mathcal{S}(h_1) \in \mathbb{T}_0$  будет или  $\succ_0$  левой или  $\prec_0$  правой части неравенства (2), и после сокращения даст эквивалентное утверждению леммы неравенство.

## Алгоритм

Вход: многочлены  $\{g_1, \dots, g_m\}$ , образующие базис Грёбнера; многочлен  $f$ .

Переменные:  $R$  и  $B$  – подмножества  $H$ ;  $(\sigma, p') \in H$  – отмеченный многочлен текущего шага до редукции;  $(\sigma, p)$  – он же после редукции;  $r, b$  – элементы  $R$  и  $B$

Результат: базис Грёбнера идеала  $I = (g_1, \dots, g_m, f)$

**SimpleSignatureGroebner**( $\{g_1, \dots, g_m\}, f$ )

1.  $R \leftarrow \{(\text{HM}(g_1), 0), (\text{HM}(g_2), 0), \dots, (\text{HM}(g_m), 0), (0, g_1), (0, g_2), \dots, (0, g_m)\}$
2.  $B \leftarrow \{\}$
3.  $(\sigma, p') \leftarrow (1, f)$
4. **do forever:**
  - (a)  $p \leftarrow \text{ReduceCheckingSignatures}(\sigma, p', R)$
  - (b)  $R \leftarrow R \cup \{(\sigma, p)\}$

- (c) if  $p \neq 0$ :
  - i. for  $\{r \in R \mid r <_{\text{gvw}} (\sigma, p), \text{HM}(r) \neq 0\}$ :
    - A.  $B \leftarrow B \cup \{\frac{\text{LCM}(\text{HM}(r), \text{HM}(p))}{\text{HM}(r)} r\}$
  - ii. for  $\{r \in R \mid r >_{\text{gvw}} (\sigma, p)\}$ :
    - A.  $B \leftarrow B \cup \{\frac{\text{LCM}(\text{HM}(r), \text{HM}(p))}{\text{HM}(p)} (\sigma, p)\}$
- (d)  $B \leftarrow B \setminus \{b \in B \mid \exists r \in R, r <_{\text{gvw}} b \wedge \mathcal{S}(r) \mid \mathcal{S}(b)\}$
- (e) if  $B \neq \emptyset$ :  $(\sigma, p') \leftarrow$  элемент  $B$  с  $\prec$ -минимальной сигнатурой
- (f) else: break

5. return  $\{\text{poly}(r) \mid r \in R\}$

### ReduceCheckingSignatures( $\sigma, p, R$ )

1. do while  $\exists r \in R \mid r >_{\text{gvw}} (\sigma, p) \wedge \text{HM}(r) \nmid \text{HM}(p)$ :
  - (a)  $p \leftarrow$  редуцировать  $p$  с сохранением сигнатуры по  $>_{\text{gvw}}$ -максимальному элементу  $r$  среди указанных в условии цикла
2. return  $p$

**Лемма 8.** Все пары из  $\mathbb{T}_0 \times P$  в алгоритме – элементы  $H \setminus \{(0, 0)\}$ .

**Доказательство.** Элементы, формируемые до начала главного цикла, являются рассмотренными выше примерами отмеченных многочленов. Все остальные отмеченные многочлены в алгоритме формируются или умножением на  $t \in \mathbb{T}$  или редукцией с сохранением сигнатуры, поэтому они корректны и лежат в  $H$ .

Условия циклов, расширяющих  $B$ , таковы, что в  $B$  нет ни нулевых сигнатур, ни нулевых старших мономов. Поэтому  $\sigma$  никогда не обращается в 0 и нулевые сигнатуры в  $R$  лишь у элементов  $(0, g_1), \dots, (0, g_m)$ . Нулевой старший моном может быть у любого многочлена, добавляемого в  $R$ , а нулевых многочленов с одновременно нулевой сигнатурой в  $R$  нет.

## Остановка алгоритма

**Лемма 9.** В любой момент работы алгоритма любой отмеченный многочлен из  $B$  может быть редуцирован с сохранением сигнатуры по некоторому элементу  $R$ .

**Доказательство.** Отмеченные многочлены добавляются в  $B$  таким образом, чтоб иметь хотя бы один подходящий редуктор.  $(\sigma, p) \in R$  является таким редуктором при добавлении в первом цикле **for**,  $r \in R$  – во втором.

**Лемма 10.** До редукции многочлена  $p'$ , то есть на шаге 4a любой итерации алгоритма, сигнатуры элементов  $\{r \in R \mid r <_{\text{gvw}} (\sigma, p')\}$  не делят  $\sigma$ .

**Доказательство.** На первой итерации алгоритма это выполняется, поскольку  $\sigma = 1$  и  $R$  не содержит элементы с сигнатурами, делящими 1. На последующих итерациях это выполнено, поскольку если бы в  $R$  существовали такие элементы, то  $(\sigma, p')$  был бы убран из  $B$  в предыдущей итерации на шаге 4d.

**Лемма 11.** После редукции многочлена  $p'$  до  $p$ , на шаге 4b любой итерации алгоритма, старшие мономы элементов  $\{r \in R \mid r >_{\text{gvw}} (\sigma, p)\}$  не делят  $\text{HM}(p)$ .

**Доказательство.** Вытекает из того, что цикл в ReduceCheckingSignatures( $\sigma, p, R$ ) останавливается по достижении  $p$ , для которого такие элементы в  $R$  не существуют.

**Лемма 12.** После редукции многочлена  $p'$  до  $p$ , на шаге 4b любой итерации алгоритма, элементы  $R$  не могут одновременно иметь старшие мономы, делящие  $\text{HM}(p)$ , и сигнатуры, делящие  $\sigma$ .

**Доказательство.** В силу леммы 9 будет произведена хотя бы одна редукция  $p'$ , поэтому  $(\sigma, p') >_{\text{gvw}} (\sigma, p)$ . Отсюда по лемме 7 для  $\forall r \in R$  имеем  $r >_{\text{gvw}} (\sigma, p)$  или  $r <_{\text{gvw}} (\sigma, p')$ . Выполнение одного из неравенств позволяет применить одну из лемм 10 и 11.

**Теорема 13.** Алгоритм SimpleSignatureGroebner( $\{g_1, \dots, g_m\}, f$ ) останавливается

**Доказательство.** Для доказательства остановки нужно показать, что все циклы **do** выполняются лишь конечное число раз. В ReduceCheckingSignatures( $\sigma, p, R$ ) при ненулевых  $p$  на каждой итерации  $\text{HM}(p)$

уменьшается по  $<_0$ , что возможно лишь конечное число раз. При обнулении  $p$  он завершится в силу  $<_{\text{gvw}}$ -минимальности  $(\sigma, 0)$ .

На каждом шаге основного цикла пополняется множество  $R \subset T_0 \times P$ . Оно может быть разбито как  $R_{*0} \cup R_{0*} \cup R_{**}$ , где  $R_{*0} \subset T \times \{0\}$ ,  $R_{0*} \subset \{0\} \times P \setminus \{0\}$ ,  $R_{**} \subset T \times P \setminus \{0\}$ .  $R_{0*}$  не пополняется в силу  $\sigma \neq 0$ . Для остальных рассмотрим идеалы моноидов:  $L_{*0} = (\{\sigma | (\sigma, 0) \in R_{*0}\}) \subset T$  и  $L_{**} = (\{(\sigma, t) | \exists (\sigma, p) \in R_{**}, t = \text{HM}(p)\}) \subset T \times T$ . В силу леммы 12 добавляемые элементы расширяют на каждом шаге  $L_{*0}$  или  $L_{**}$ , что по лемме Диксона может происходить лишь конечное число раз.

## Корректность результата

**Определение 14.**  $S$ -представлением  $h \in H$  над множеством  $\{r_i\} \subset H$  будем называть выражение  $\text{poly}(h) = \sum_j K_j t_j \text{poly}(r_{i_j})$ ,  $K_j \in k, t_j \in T, i_j \in \mathbb{N}$ , такое что  $\forall j \text{ HM}(h) \succ_0 \text{HM}(t_j r_{i_j}), \mathcal{S}(h) \succ_0 \mathcal{S}(t_j r_{i_j})$ .

**Лемма 15.** Пусть  $\text{poly}(h) = \sum_j K_j t_j \text{poly}(r_{i_j})$  –  $S$ -представление для  $h$ . Тогда для хотя бы одного  $j$  достигается  $\text{HM}(h) = \text{HM}(t_j r_{i_j})$ .

**Доказательство.** В качестве такого  $j$  можно взять то, на котором достигается  $\succ$ -максимум  $\text{HM}(t_j r_{i_j})$ .

Следующее определение расширяет понятие  $S$ -базиса из работы [1]:

**Определение 16.** Назовём  $R \subset H$   $S$ -базисом (соответственно  $S_\sigma$ -базисом), если  $\forall h \in H$  (соответственно  $\forall h \in H, \mathcal{S}(h) <_0 \sigma$ ) имеет  $S$ -представление над  $R$ .

**Лемма 17.** Пусть  $\sigma \succ_0 0, R = \{r_i\}$  –  $S_\sigma$ -базис и выбраны  $h_1, h_2 \in H, \mathcal{S}(h_i) = \sigma$ , которые не редуцируются по  $R$  с сохранением сигнатуры. Тогда  $\text{HM}(h_1) = \text{HM}(h_2)$  и у  $h_1$  есть  $S$ -представление над  $R \cup \{h_2\}$ .

**Доказательство.** Из определения  $H$  имеем  $\exists u_i \in P : \text{HM}(u_i) = \sigma, u_i f \equiv \text{poly}(h_i) \pmod{I_0}, i = 1, 2$ . Значит некоторой линейной комбинации  $\text{poly}(h_i)$  сопоставляется  $<_0 \sigma$  сигнатура:

$$\exists K \in k \exists v \in P : \text{HM}(v) = \sigma' <_0 \sigma, v f \equiv \text{poly}(h_1) - K \text{poly}(h_2) \pmod{I_0},$$

то есть  $(\sigma', p') = (\sigma', \text{poly}(h_1) - K \text{poly}(h_2)) \in H$ . Из определения  $S_\sigma$ -базиса и  $\sigma' <_0 \sigma$  вытекает  $\exists r_j \in R, t \in T : \mathcal{S}(tr_j) \leq_0 \sigma', \text{HM}(tr_j) = \text{HM}(p')$ . Отсюда  $\text{HM}(h_i) \neq \text{HM}(p'), i = 1, 2$ , иначе  $r_j$  редуцировало бы  $h_i$  с сохранением сигнатуры. Значит,  $\text{HM}(h_i)$  сокращаются при вычитании с  $k$ -коэффициентом, что даёт  $\text{HM}(h_1) = \text{HM}(h_2)$ .  $S$ -представление  $h_1$  получается добавлением  $K \text{poly}(h_2)$  к  $S$ -представлению  $(\sigma', p')$ .

**Теорема 18.** На каждой итерации алгоритма после шага 4d выполнен инвариант: для  $\forall \sigma \in T, \sigma < \text{сигнатур элементов } B$ , найдутся  $r_\sigma \in R, t_\sigma \in T : \mathcal{S}(t_\sigma r_\sigma) = \sigma$  и  $t_\sigma r_\sigma$  не редуцируется по  $R$  с сохранением сигнатуры.

**Доказательство.** Множество  $R_\sigma = \{r \in R | \mathcal{S}(r) | \sigma\}$  непусто, так как содержит добавленный на первой итерации элемент  $r_0$  с  $\mathcal{S}(r_0) = 1$ . Обозначим за  $r_\sigma$  его  $<_{\text{gvw}}$ -минимальный элемент; положим  $t_\sigma = \frac{\sigma}{\mathcal{S}(r_\sigma)}$ . Предположим, что  $t_\sigma r_\sigma$  может быть редуцирован с сохранением сигнатуры относительно некоторого  $r_1 \in R$ . Отсюда следует, что  $r_1 >_{\text{gvw}} r_\sigma$ , а также что они не нулевые. Значит на той же итерации, когда в  $R$  был добавлен последний из  $\{r_\sigma, r_1\}$ , в множество  $B$  был добавлен многочлен  $t' r_\sigma$ , где  $t' = \frac{\text{LCM}(\text{HM}(r_1), \text{HM}(r_\sigma))}{\text{HM}(r_\sigma)}$ , причём  $t' | t_\sigma$ . Отсюда  $\mathcal{S}(t' r_\sigma) | \mathcal{S}(t_\sigma r_\sigma) = \sigma \Rightarrow \mathcal{S}(t' r_\sigma) \preceq \sigma < \text{сигнатур элементов } B$ . В силу этого неравенства на сигнатуры получается, что  $t' r_\sigma$  уже не может быть элементом  $B$ , а значит был выкинут на шаге 4d одной из итераций, то есть  $\exists r_2 \in R, r_2 <_{\text{gvw}} t' r_\sigma, \mathcal{S}(r_2) | \mathcal{S}(t' r_\sigma)$ . Это невозможно, поскольку влечёт  $r_2 <_{\text{gvw}} r_\sigma, r_2 \in R_\sigma$ , что противоречит  $<_{\text{gvw}}$ -минимальности  $r_\sigma$ .

**Теорема 19.** На каждой итерации алгоритма после шага 4d выполнен инвариант:  $\forall h \in H, \mathcal{S}(h) < \text{сигнатур элементов } B$ , имеет  $S$ -представление над  $R$ .

**Доказательство.** Предположим нарушение инварианта на какой-то итерации и рассмотрим  $<_0$ -минимальную  $\sigma$ , для которой  $V_\sigma \stackrel{\text{def}}{=} \{h \in H \text{ нарушает инвариант} | \mathcal{S}(h) = \sigma\} \neq \emptyset$ . Тогда  $R$  –  $S_\sigma$ -базис.  $\forall g \in I_0$   $(0, g)$  имеют  $S$ -представления над  $\{(0, g_1), \dots, (0, g_m)\} \subset R$ , поэтому  $\sigma \succ_0 0$ . Выберем  $v_\sigma$  – один из элементов  $V_\sigma$  с  $<_0$ -наименьшим  $\text{HM}$ . Он не может быть редуцирован с сохранением сигнатуры по  $R$ , поскольку результат редукции  $v_1$  был бы элементом  $V_\sigma$  с  $\text{HM}(v_1) <_0 \text{HM}(v_\sigma)$ . Возьмём  $w_\sigma \stackrel{\text{def}}{=} t_\sigma r_\sigma$  из инварианта теоремы 18 и применив лемму 17 к  $v_\sigma, w_\sigma$  и  $R$  получим что  $v_\sigma$  имеет  $S$ -представление над  $R \cup \{w_\sigma\}$ . Вхождения  $w_\sigma$  в нём можно заменить на  $t_\sigma r_\sigma$ , получив представление  $v_\sigma$  над  $R$ , что приводит к противоречию.

**Лемма 20.** Если  $R$  –  $S$ -базис, то  $R$  является базисом Грёбнера идеала  $I$ .

**Доказательство.** Для  $\forall p \in I$  можно взять некоторый  $h = (\sigma, p) \in H$  и применить лемму 15.

**Теорема 21.** SimpleSignatureGroebner( $\{g_1, \dots, g_m\}, f$ ) возвращает базис Грёбнера

**Доказательство.** К моменту остановки  $B = \emptyset$ , значит по теореме 19  $R$  –  $S$ -базис.

## Сравнение с аналогами

Представленный алгоритм связан с двумя уже известными алгоритмами, основанных на сигнатурах – алгоритмом G2V из работы [2], версией F5, опубликованную в работе [1] и схеме алгоритма TRB-MJ, предложенной Lei Huang в статье "A new conception for computing Gröbner basis and its applications", депонированной на <http://arxiv.org/abs/1012.5425v2>. Все они в определённом смысле являются модификациями простого F5, впервые представленного в работе [3], причём модификации направлены на сокращение числа редукций многочленов, занимающих большую часть времени в процессе вычисления базисов Грёбнера. Первый из них отличается от немодифицированной версии тем, что не использует явно вычисления S-полиномов до проведения редукции и вводит специальный критерий "super-topreducible" для отбрасывания некоторых многочленов после их редукции. Второй применяет расширенный критерий, отбрасывая до редукции те из посчитанных S-полиномов, которые не смогут удовлетворять определённому в работе критерию "primitive S-irreducible" после редукции. Эти методики позволяют ускорить работу алгоритма, однако в обоих случаях в определённый момент отбрасывается многочлен, полученный в процессе предыдущих вычислений путём редукции или создания S-полинома.

В алгоритме SimpleSignatureGroebner S-полиномы не вычисляются явно и их отбрасывание осуществляется на основе критерия, записываемого в точности как критерий алгоритма из [1]. При этом для проверки на шаге 4d элемента  $B$  достаточно знать лишь его сигнатуру и старший моном. Поскольку каждый элемент  $B$  получается как домноженный на моном  $m$  элемент  $r \in R$  их можно хранить в виде пары  $(m, r)$ , не выполняя без необходимости операции умножения всего многочлена на моном. Таким образом, для отброшенных элементов  $B$  не производится никаких операций, сложность которых пропорциональна длине многочлена. Заметим, что применяемое в алгоритме из [1] вычисление старшего монома S-пары в общем случае напротив имеет именно такую сложность. Это делает возможной реализацию алгоритма SimpleSignatureGroebner, в которой все длительные операции над многочленами производятся лишь для многочленов, попадающих в результирующее множество. Хотя результирующее множество не является минимальным базисом Грёбнера, оно является минимальным S-базисом идеала, дополненным сигнатурами некоторых нулевых многочленов. Отсюда можно сделать вывод о некоторой оптимальности критериев отбрасывания в случае регулярной входной последовательности: не может существовать критерия, который бы позволил убрать какие-либо операции редукции многочленов из алгоритма не добавив новых редукций и не лишив результат свойства быть S-базисом.

Другим отличием алгоритма SimpleSignatureGroebner от двух рассмотренных выше и от TRB-MJ является отсутствие явной проверки критериев, связанных с *сизигиями* – нулевыми многочленами ненулевой сигнатуры. Это вызвано тем, что сизигии являются  $<_{\text{gvw}}$ -меньшими элементами  $R$  и участвуют в отбрасывании элементов  $B$  наравне с другими многочленами, реализуя критерии отбрасывания эквивалентные алгоритму G2V, являющиеся оптимальными в смысле содержания минимального базиса идеала сигнатур сизигий в моноиде  $T$ .

Вопрос наиболее эффективного способа выбора редуктора в  $\text{ReduceCheckingSignatures}(\sigma, p, R)$  является открытым. Представленный в этой работе способ выбора, основанный на  $<_{\text{gvw}}$ -сравнении совпадает со способом выбора, применявшегося в оригинальном алгоритме F5 для случая однородных многочленов позволяет осуществить этот выбор на основе всё того же упорядочения множества  $R$ .

Таким образом алгоритм SimpleSignatureGroebner реализует все критерии отбрасывания и выбора редуктора на основе одного и того же простого порядка  $<_{\text{gvw}}$  на  $R$ , что и позволяет сформулировать и создать его простую реализацию без дополнительных списков многочленов (как в G2V и TRB-MJ) или правил (как в F5), которая оказывается не менее эффективной чем указанные алгоритмы.

Алгоритм был реализован на C++ с использованием функций ядра программного комплекса Singular 3-1-4 и открытых наработок Кристиана Эдера (одного из авторов [4]) по реализации F5-подобных алгоритмов на этом ядре. Исходный код реализации содержится в функции ssg файла, доступного по адресу <https://github.com/galkinvv/Singular-f5-like/blob/ssg/kernel/kstd2.cc>

Сравнение реализации SimpleSignatureGroebner с другими алгоритмами вычисления базисов Грёбнера, реализованных Christian Eder подтвердили следующие соображения:

- Алгоритм SimpleSignatureGroebner корректно вычисляет базис Грёбнера
- Результат содержит не большее число многочленов, чем результат других инкрементальных алгоритмов, возвращающих S-базис
- Время работы алгоритма оказывается не больше, чем у других инкрементальных алгоритмов, основанных на сигнатурах

## СПИСОК ЛИТЕРАТУРЫ

1. *Arri A., Perry J.* The F5 criterion revised // Journal of Symbolic Computation. 2011. **46**, № 9. 1017–1029.
  2. *Gao S., Guan Y., Volny F.* A new incremental algorithm for computing Groebner bases // Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation. ACM. New York. 2010. 13–19.
  3. *Faugère J.-C.* A new efficient algorithm for computing Gröbner bases without reduction to zero (F5) // Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation. ACM. New York. 2002. 75–83.
  4. *Eder C., Perry J.* Signature-based algorithms to compute Gröbner bases // Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation. ACM. New York. 2011. 99–106.
-