УДК 512

ОСТАНОВКА АЛГОРИТМА F5

В. В. Галкин[1]

Алгоритм F5, предложенный Фожером, принимает в качестве входных данных произвольное множество однородных многочленов и корректность результата доказана для всех случаев, когда алгоритм останавливается. Однако остановка алгоритма за конечное число шагов доказана лишь для случая, когда на вход алгоритма подаётся регулярная последовательность многочленов. В этой работе показано, что алгоритм останавливается на любых входных данных без какого-либо использования регулярности

Ключевые слова: базис Грёбнера, алгоритм F5, доказательство остановки

The original F5 algorithm introduced by Faugère is formulated for any homogeneous polynomial set input. The correctness of output is shown for any input that terminates the algorithm, but the termination itself is proved only for the case of input being regular polynomial sequence. This article shows that algorithm correctly terminates for any homogeneous input without any reference to regularity

Key words: Groebner basis, F5 algorithm, termination proof

Алгоритм Фожера F5 является эффективным алгоритмом вычисления базисов Грёбнера, но обладает рядом проблем, одна из которых – отсутствие доказательства остановки для всех входных данных. Первоисточник [1] и детальные исследования в [2] показывают остановку алгоритма только для случая отсутствия редукций к нулю, что означает доказательство для тех случаев, когда входное множество многочленов представлено регулярной последовательностью. Но для большинства входных последовательностей их регулярность неизвестна. Один из подходов к решению проблемы – добавление в алгоритм дополнительных проверок и критериев, гарантирующих остановку алгоритма. Этот подход даёт строгое доказательство остановки, однако получаемый результат есть доказательство остановки модифицированной версии F5, содержащей дополнительные проверки, которая в силу этого может быть более сложна в реализации или иметь большее время работы на некоторых входных данных. Этот подход применяется в работах [3,4,5,6,7].

Другой подход состоит в доказательстве остановки семейства алгоритмов, основанных на идеях F5, с последующей попыткой переформулировать F5 таким образом, чтоб он являлся представителем этого семейства. Основная проблема этого подхода появляется в процессе переформулировки: описание F5 в других терминах может привести к неявному внесению различий в поведение алгоритма. К примеру, [8] доказывает остановку алгоритма F5GEN, который отличается от исходного F5 отсутствием проверки критериев при выборе редуктора. Работа [9] даёт доказательство остановки алгоритма TRB-F5, который имеет два отличия от F5. Первое – другая схема построения правил, приводящая к тому, что в TRB-F5 правила в массивах *Rule* оказываются отсортированными по возрастанию сигнатуры. Второе – отсутствие применения оператора нормальной формы φ перед редукцией, поэтому TRB-F5 при выборе редуктора проверяет критерии для элементов, которые в F5 используются неявно в φ. Предположительно, эти алгоритмы могут быть изменены таким образом, чтоб в точности повторять поведение алгоритма F5, а доказательство остановки может быть перенесено на изменённые версии. Однако подход с алгоритмами, эквивалентными F5 имеет недостаток: он усложняет понимание того, как теоремы, используемые для доказательства остановки отражают поведение исходного алгоритма F5.

Подход к доказательству остановки, предлагаемый в данной работе, применяется к F5 без каких-либо модификаций. Первый шаг доказательства основан на предлагаемой ниже идеи цепей S-пар. Второй шаг основывается на методе, использованном в теореме 21 работы [10] для доказательства корректности алгоритма F5C: представление S-многочлена в виде суммы домноженных многочленов, вычисленных ранее, может быть модифицировано последовательностью замен S-пар, и за конечное число таких шагов приведено к состоянию, когда выполняются определённые «хорошие» свойства.

Работа оформлена как альтернативное доказательство остановки алгоритма, описанного в статье [1], доступной на сайте её автора, поэтому читатель предполагается хорошо знакомым с ней. Большинство используемой терминологии, включая названия процедур, взято оттуда. Термин “signature-safe” переводится как «сигнатурный»:

Определение 1. Редукция отмеченного многочлена r_k по отмеченному многочлену r_m называется *сигнатурной редукцией,* если $\mathcal{S}(r_k) \succ t \cdot \mathcal{S}(r_m)$, где $t = \frac{\mathrm{HM}(r_k)}{\mathrm{HM}(r_m)}$ – моном, на который умножается редуктор. Редуктор, соответствующий такой редукции называется *сигнатурным редуктором.*

[1] *Галкин Василий Витальевич* — асп. каф. алгебры мех.-мат. ф-та МГУ, e-mail: galkin-vv@yandex.ru.

Цепи S-пар

При отсутствии остановки алгоритм даёт бесконечную последовательность многочленов $\{r_1, r_2, \ldots, r_m, \ldots, r_l, \ldots\}$, в которой $r_1, \ldots, r_m$ соответствуют m исходным многочленам, а остальные были получены в процедурах `Spol` и `TopReduction`, как S-многочлены двух уже добавленных. В этой главе мы будем искать бесконечную подпоследовательность $\{r_{k_1}, r_{k_2}, \ldots, r_{k_n}, \ldots\}$ этой последовательности, обладающую тем свойством, что r_{k_n} является S-многочленом $r_{k_{n-1}}$ и некоторой другой части с меньшей сигнатурой: $\mathcal{S}(r_{k_n}) = u_{k_n}\mathcal{S}(r_{k_{n-1}})$ и $\mathcal{S}(r_{k_{n-1}})|\mathcal{S}(r_{k_n})$.

Определение 2. Конечную или бесконечную последовательность отмеченных многочленов, соседние элементы которой удовлетворяют последнему свойству, будем называть *цепью S-пар*.

Каждый порождаемый многочлен r_l имеет конечную цепь S-пар, оканчивающуюся этим многочленом. Эта цепь может быть последовательно построена, начиная с последнего элемента r_l, если на каждом шаге переходить от текущего многочлена к части породившей его S-пары с большей сигнатурой.

Теорема 3. *Любой отмеченный многочлен может являться начальным элементом лишь конечного числа различных цепей S-пар длины 2.*

Доказательство. Рассмотрим произвольный отмеченный многочлен r_L с сигнатурой $\mathcal{S}(r_L) = s$ и упорядоченное по порядку добавления подмножество $\{r_{l_1}, \ldots, r_{l_i}, \ldots\}$ отмеченных многочленов с сигнатурами удовлетворяющими условию $\mathcal{S}(r_{l_i}) = v_i\mathcal{S}(r_L)$. Идеал (v_i) в T является конечно порождённым по лемме Диксона, поэтому после некоторого шага i_0 будет выполняться $\forall i > i_0\, \exists j \leqslant i_0$ такое что $v_j|v_i$. С другой стороны, при построения S-многочлена с сигнатурой s старшая часть S-пары соответствует последнему из правил с сигнатурой, делящей s. Поэтому при $\forall i > i_0$ последовательность $\{r_L, r_{l_i}\}$ не может являться цепью S-пар, поскольку $\mathcal{S}(r_L)\cdot v_i$ перезаписывается $\mathcal{S}(r_{l_j})\cdot\frac{v_i}{v_j}$ и существует не более чем i_0 цепей S-пар длины 2, начинающихся с многочлена r_L. □

Определение 4. Конечное множество концов цепей S-пар длины 2, начинающихся с r_L будет называться *множеством S-порождённых* r_L.

Теорема 5. *Если алгоритм не останавливается на некоторых входных данных, то он порождает бесконечную цепь S-пар* $\{h_i\}$.

Доказательство. Если алгоритм не останавливается, то многочлен входного множества $r_1 = (f_1, 1F_1)$ является началом бесконечного числа различных конечных цепей S-пар – каждому многочлену r_l соответствует цепь $\{r_1, \ldots, r_{l^{**}}, r_{l^*}, r_l\}$. Среди конечного числа его S-порождённых также найдётся элемент, являющийся началом бесконечного числа S-пар. В его S-порождённых можно выбрать элемент с тем же свойством. Продолжая выбирать элементы таким образом, мы получим искомую бесконечную последовательность, состоящую из элементов, являющихся началами бесконечного числа S-пар. □

Для следующей теоремы необходимо ввести порядок на частных, образованных мономами, путём транзитивного расширения порядка на мономах: $\frac{m_1}{m_2} >_q \frac{m_3}{m_4} \Leftrightarrow m_1m_4 > m_3m_2$.

Теорема 6. *Если алгоритм не останавливается на некоторых входных данных, то после некоторого конечного шага множество G содержит пару отмеченных многочленов f', f, причём f сгенерирован после f' и выполняются свойства* $\mathrm{HM}(f')|\mathrm{HM}(f)$ *и* $\frac{\mathrm{HM}(f')}{\mathcal{S}(f')} >_q \frac{\mathrm{HM}(f)}{\mathcal{S}(f)}$.

Доказательство. При работе с цепями S-пар является важным тот факт, что многочлен никогда не редуцируется дальше, после того как он был использован для создания S-пары в качестве старшей по сигнатуре части. Факт выполняется, поскольку все многочлены, которые ещё могут быть подвергнуты редукции находятся в множестве $ToDo$, а все многочлены, используемые как старшая часть S-пары, находятся в G или в $Done$. Поэтому многочлен h_n, предшествующий многочлену h_{n+1} в цепи S-пар, сохраняет одно и то же значение $\mathrm{poly}(h_n)$ после того как был использован для создания какой-либо S-пары. И можно утверждать, что выполняется равенство $\mathrm{poly}(h_{n+1}) = c\frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)}\mathrm{poly}(h_n) + g_n$, где g_n – многочлен, соответствующей младшей части S-пары, использованный при генерации h_{n+1} из h_n, удовлетворяющее следующему:

$$\mathrm{HM}(h_{n+1}) < \mathrm{HM}\left(\frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)}h_n\right) = \mathrm{HM}(g_n),\ \mathcal{S}(h_{n+1}) = \mathcal{S}\left(\frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)}h_n\right) \succ \mathcal{S}(g_n). \tag{1}$$

Из первого неравенства в (1) получаем, что $\frac{\mathrm{HM}(h_n)}{\mathcal{S}(h_n)} >_q \frac{\mathrm{HM}(h_{n+1})}{\mathcal{S}(h_{n+1})}$, поэтому в цепи S-пар частные $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)}$ строго убывают в смысле порядка на частных.

В цепи S-пар $\{h_i\}$ для различных h_i и h_j делимость $\mathrm{HM}(h_i)|\mathrm{HM}(h_j)$ возможна только в случае $i < j$ и $\deg(h_i) < \deg(h_j)$. Для нахождения таких многочленов используем идею из Предложения 14 работы [11]: для бесконечной цепи S-пар $\{h_i\}$, рассмотрим бесконечную последовательность в T, образованную $\{\mathrm{HM}(h_i)\}$. В ней по лемме Диксона существует два элемента, удовлетворяющие $\mathrm{HM}(h_i)|\mathrm{HM}(h_j)$. При этом $i < j$, а при помощи свойств цепи S-пар мы получаем, что $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)} >_q \frac{\mathrm{HM}(h_{i+1})}{\mathcal{S}(h_{i+1})} >_q \cdots >_q \frac{\mathrm{HM}(h_j)}{\mathcal{S}(h_j)}$, поэтому можно взять $f' = h_i$ и $f = h_j$. □

Из неравенства на частных следует, что f – сигнатурный редуктор f'. Последующие главы посвящены доказательству невозможности такой ситуации, приводящей к противоречию предположение об отсутствии остановки.

S-пары с сигнатурами, меньшими $\mathcal{S}(g)$

В этой и последующих частях g подразумевается некоторым фиксированным многочленом с индексом сигнатуры 1, добавленным на некоторой итерации алгоритма в *Done*. Мы будем анализировать состояние алгоритма в момент непосредственно предшествующий добавлению g в *Done*. Рассмотрим в этот момент конечное множество $G_1 \cup Done$. Оно состоит из чисел, являющихся позициями отмеченных многочленов в R, поэтому его элементы могут быть упорядочены в соответствии с позицией в R и оно окажется записанным в виде упорядоченной последовательности целых чисел $G_g = \{b_1, \ldots, b_N\}$ с $b_j < b_{j+1}$. Необходимо отметить, что этот порядок соответствует порядку отмеченных многочленов в последовательности, получаемой склеиванием массивов правил $Rule[m] : Rule[m-1] : \cdots : Rule[1]$, поскольку добавление нового многочлена в R всегда сопровождается добавлением соответствующего правила. Но этот порядок может отличаться от порядка в котором многочлены добавлялись в множество $G_1 \cup Done$, поскольку многочлены одной полной степени добавляются в *Done* в порядке возрастания сигнатуры, при том что добавление многочленов одной полной степени в R производится в довольно случайном порядке в процедурах `Spol` и `TopReduction`. Далее для простоты мы будем говорить о отмеченных многочленах b_j в G_g, подразумевая что G_g является не упорядоченным списком позиций, а упорядоченным списком отмеченных многочленов, расположенных на этих позициях. В этой терминологии можно сказать, что все входные многочлены $\{f_1, \ldots, f_m\}$ присутствуют в G_g, поскольку они присутствуют в G_1 в момент его создания.

S-пары могут обрабатываться в алгоритме различным путями, но главный факт, описывающий порядок их обработки выражается следующими свойствами, соответствующими свойствам, использованным в Теореме 21 работы [10], но рассматриваются на произвольной итерации алгоритма, а не после его остановки.

Теорема 7. *К моменту добавления g в Done каждая S-пара элементов G_g, сигнатура которой меньше $\mathcal{S}(g)$, удовлетворяет одному из трёх свойств:*

1. S-пара имеет часть, которая была отброшена критерием проверки нормальной формы φ (в `CritPair` или в `IsReducible`). Такие S-пары будут называться *S-парами с частью, удовлетворяющей критерию F5.*
2. S-пара имеет часть, которая была отброшена проверкой `Rewritten?` (в `SPol` или в `IsReducible`). Такие S-пары будут называться *S-парами с частью, удовлетворяющей критерию Перезаписи.*
3. S-пара не была отброшена, её S-многочлен был сигнатурно редуцирован по некоторым элементам G_g и результат был добавлен в G_g. Такие S-пары будут называться *S-парами с известным G_g-представлением.*

Доказательство. Вытекает из порядка обработки S-пар и S-многочленов в алгоритме. □

Понятия *удовлетворять критерию F5* и *удовлетворять критерию Перезаписи* могут быть расширены на произвольные умноженные на моном отмеченные многочлены sh, $h \in G_g$:

Определение 8. Умноженный на моном отмеченный многочлен sr_i, $r_i \in G_g$ называется *удовлетворяющим критерию F5*, если $\varphi_{index(r_i)+1}(s\mathcal{S}(r_i)) \neq s\mathcal{S}(r_i)$, где $\varphi_{index(r_i)+1}$ – нормальная форма относительно $G_{index(r_i)+1}$.

Это определение эквивалентно тому, что sr_i является ненормализованным отмеченным многочленом с точки зрения определения 2 в части 5 работы [1].

Определение 9. Умноженный на моном отмеченный многочлен sr_i, $r_i \in G_g$ называется *удовлетворяющим критерию Перезаписи*, если $\exists j > i$ такое что $\mathcal{S}(r_j) | s\mathcal{S}(r_i)$.

Для обоих критериев выполняется важное свойство, утверждающее, что если sr_i удовлетворяет критерию, то то и дополнительно домноженный многочлен $s_1 sr_i$ также ему удовлетворяет.

Представления

Идея представлений, определённых ниже, приходит из [10], где подобный метод используется в доказательстве Теоремы 21. Представления используются для описания способов, которыми многочлен p может быть записан как элемент идеала (G_g). Одно представление соответствует записи отмеченного многочлена p в виде конечной суммы вида $p = \sum_k m_k \cdot b_{i_k}$, $b_{i_k} \in G_g$ с коэффициентами $m_k = c_k t_k \in \mathcal{K} \times T$.

Определение 10. Сумма такого вида, в которой все пары (t_k, b_{i_k}) различны, называется *G_g-представлением* p. Символические произведения $m_k \cdot b_{i_k}$ называются *элементами* представления. Они же, рассмотренные как умножение многочленов, дают отмеченные многочлены $m_k b_{i_k}$, соответствующие элементам представления.

Большинство представлений также имеют следующее свойство, ограничивающее сигнатуру элементов:

Определение 11. G_g-представление p называется *сигнатурным*, если $\forall k\, \mathcal{S}(m_k b_{i_k}) \preccurlyeq \mathcal{S}(p)$.

Порядок на представлениях

Определение 12. Для введения порядка на G_g-представлениях мы начнём с *порядка $\succcurlyeq_1$ на элементах представления*: будем говорить, что $c_i t_i \cdot b_i \succcurlyeq_1 c_j t_j \cdot b_j$ если $t_i \mathcal{S}(b_i) \succ t_j \mathcal{S}(b_j)$ или $t_i \mathcal{S}(b_i) = t_j \mathcal{S}(b_j)$ и $i < j$.

Используем *упорядоченную форму* представления – $\succcurlyeq_1$-убывающий список его элементов.

Определение 13. Введём *порядок* $\succcurlyeq$ *на* G_g*-представлениях*: представление $\sum_k m'_k \cdot b_{i'_k}$ является $\succcurlyeq$-большим, чем $\sum_k m_k \cdot b_{i_k}$, если их упорядоченные формы удовлетворяют лексикографически расширенному отношению $\succcurlyeq_1$.

Порядок на представлениях совместим с понятием сигнатурности представления:

Теорема 14. *Если для пары представлений отмеченного многочлена* p *выполняется отношение* $\sum_k m'_k \cdot b_{i'_k} \preccurlyeq \sum_k m_k \cdot b_{i_k}$ *и правое представление сигнатурно, то и левое представление сигнатурно.*

Доказательство. Теорема легко следует из того, что сигнатуры элементов $\preccurlyeq$-меньшего представления не могут быть $\succ$-больше, чем максимальная сигнатура $\succcurlyeq$-большего представления. □

Важным фактом, позволяющим брать $\preccurlyeq$-минимальный элемент, является вполне упорядоченность:

Теорема 15. *Представления вполне упорядочены порядком* $\preccurlyeq$.

Доказательство. Вполне упорядоченность сигнатур по $\prec$ даёт вполне упорядоченность элементов представлений по $\preccurlyeq_1$. Теорема 2.5.5 книги [12] утверждает вполне упорядоченность конечных наборов, упорядоченных лексикографическим расширением порядка на их составляющих. Это применимо к представлениям – наборам своих элементов. □

Последовательность представлений

В этой части строится конечная последовательность $\preccurlyeq$-убывающих G_g-представлений для некоторого отмеченного многочлена mh, $m \in \mathcal{K} \times T$, $h \in G_g$ со свойством $\mathcal{S}(mh) \prec \mathcal{S}(g)$. Она начинается с сигнатурного представления $mh = m \cdot h$, и заканчивается представлением $mh = \sum_k m_k \cdot b_{i_k}$ со свойствами, выполняющимися для $\forall k$:

1. $m_k b_{i_k}$ не удовлетворяет критерию F5.
2. $m_k b_{i_k}$ не удовлетворяет критерию Перезаписи.
3. $\mathrm{HM}(m_k b_{i_k}) \leqslant \mathrm{HM}(mh)$

Доказательство существования такой последовательности основано на том, что если некоторое сигнатурное представление mh содержит не удовлетворяющий одному из свойств элемент $m_K \cdot b_{i_K}$, то может быть найдено $\preccurlyeq$-меньшее представление. Схема построения такова: выбирается элемент $m_Q \cdot b_{i_Q}$ в представлении mh, где Q может как совпадать, так и отличаться от K. Далее строится некоторое представление $m_Q b_{i_Q} = \sum_l m_l \cdot b_{i_l}$ для этого элемента и показывается, что оно $\preccurlyeq$-меньше, чем представление $m_Q b_{i_Q} = m_Q \cdot b_{i_Q}$. К нему применяется следующая лемма:

Лемма 16. *Если элемент* $m_Q \cdot b_{i_Q}$ *представления* $mh = \sum_k m_k \cdot b_{i_k}$ *имеет представление* $m_Q b_{i_Q} = \sum_l m_l \cdot b_{i_l}$, $\preccurlyeq$*-меньшее, чем* $m_Q b_{i_Q} = m_Q \cdot b_{i_Q}$, *то* mh *имеет представление* $\preccurlyeq$*-меньшее, чем* $\sum_k m_k \cdot b_{i_k}$.

Доказательство. Заменим элемент $m_Q \cdot b_{i_Q}$ в представлении $mh = \sum_k m_k \cdot b_{i_k}$ на $\sum_l m_l \cdot b_{i_l}$ и скомбинируем коэффициенты при элементах с одновременно одинаковыми мономами и многочленами. □

Осталось показать, что представление элемента по вышеуказанной схеме может быть построено. Выражения, позволяющие построить такое представления описаны в работе [10] и довольно громоздки, поэтому здесь мы лишь приведём точную формулировку лемм, позволяющих применить эти выражения в нашем случае.

Лемма 17. *Если сигнатурное* G_g*-представление* $mh = \sum_k m_k \cdot b_{i_k}$ *содержит элемент* $m_K \cdot b_{i_K}$, *не удовлетворяющий свойству 1, то он обладает* G_g*-представлением* $m_K b_{i_K} = \sum_l m_l \cdot b_{i_l}$ $\preccurlyeq$*-меньшим, чем представление* $m_K b_{i_K} = m_K \cdot b_{i_K}$.

Доказательство. Рассмотрим выражение, аналогичное упоминаемому в Теореме 20 из [10]. □

Лемма 18. *Если сигнатурное* G_g*-представление* $mh = \sum_k m_k \cdot b_{i_k}$ *с* $\mathcal{S}(mh) \prec \mathcal{S}(g)$ *содержит элемент* $m_K \cdot b_{i_K}$, *не удовлетворяющий свойству 2, то он обладает* G_g*-представлением* $m_K b_{i_K} = \sum_l m_l \cdot b_{i_l}$ $\preccurlyeq$*-меньшим, чем представление* $m_K b_{i_K} = m_K \cdot b_{i_K}$.

Доказательство. Рассмотрим выражение, аналогичное упоминаемому в Предложении 17 из [10]. □

Лемма 19. *Если все элементы сигнатурного представления* $mh = \sum_k m_k \cdot b_{i_k}$ *с* $\mathcal{S}(mh) \prec \mathcal{S}(g)$ *удовлетворяют свойствам 1 и 2, но хотя бы один из них не удовлетворяет свойству 3, то найдётся элемент* $m_Q \cdot b_{i_Q}$, *имеющий представление* $m_Q b_{i_Q} = \sum_l m_l \cdot b_{i_l}$ $\preccurlyeq$*-меньшее, чем представление* $m_Q b_{i_Q} = m_Q \cdot b_{i_Q}$.

Доказательство. Применим идею о сокращении старших мономов, аналогично Теореме 21 из [10]. □

Теорема 20. *Или сигнатурное представление* $mh = \sum_k m_k \cdot b_{i_k}$ *с* $\mathcal{S}(mh) \prec \mathcal{S}(g)$ *удовлетворяет свойствам 1-3 или существует другое сигнатурное представление* $mh = \sum_l m_l \cdot b_{i_l}$ $\preccurlyeq$*-меньшее, чем* $\sum_k m_k \cdot b_{i_k}$.

Доказательство. Теорема немедленно следует из комбинации четырёх предыдущих лемм. □

Теорема 21. *Для любого многочлена* mh, $m \in \mathcal{K} \times T$, $h \in G_g$ *с* $\mathcal{S}(mh) \prec \mathcal{S}(g)$ *существует удовлетворяющее свойствам 1-3 сигнатурное* G_g*-представление* $mh = \sum_k m_k \cdot b_{i_k}$.

Доказательство. Начнём с представления $mh = m \cdot h$ и будем заменять текущее представление на $\preccurlyeq$-меньшее из теоремы 20 до тех пор, пока текущее представление не будет удовлетворять свойствам 1-3. Конечность процесса гарантируется вполне упорядоченностью представлений по $\preccurlyeq$. □

Следствие 22. *Рассмотрим произвольный многочлен f без ограничений на его сигнатуру. Если существует сигнатурный редуктор $f' \in G_g$ для f с $\mathcal{S}(f')\frac{\mathrm{HM}(f)}{\mathrm{HM}(f')} \prec \mathcal{S}(g)$, то G_g содержит сигнатурный редуктор для f, который не отбрасывается критериями F5 и перезаписи.*

Доказательство. Пусть mf', $m = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f')} \in \mathcal{K} \times T$, $f' \in G_g$ умноженный редуктор с $\mathcal{S}(mf') \prec \mathcal{S}(g)$. Из предыдущей теоремы мы можем найти удовлетворяющее свойствам 1-3 представление $mf' = \sum_k m_k \cdot b_{i_k}$. Свойство 3 означает отсутствие элементов с HM, большим чем у mf', значит существует K, на котором достигается равенство с $\mathrm{HM}(mf')$: $\mathrm{HM}(m_K \cdot b_{i_K}) = \mathrm{HM}(mf') = \mathrm{HM}(f)$. Из сигнатурности имеем $\mathcal{S}(m_K \cdot b_{i_K}) \preccurlyeq \mathcal{S}(mf') \prec \mathcal{S}(f)$, а значит $m_K b_{i_K}$ – сигнатурный редуктор для f и свойства 1-2 гарантируют прохождение критериев. □

Обнаружение противоречия

Теорема 6 при отсутствии остановки утверждает существование многочленов $f', f \in G$, таких что $\mathrm{HM}(f')|\mathrm{HM}(f)$, $\frac{\mathrm{HM}(f')}{\mathcal{S}(f')} >_q \frac{\mathrm{HM}(f)}{\mathcal{S}(f)}$. Применяя к ним последнее следствие построим приводящие к противоречию многочлены.

Теорема 23. *Если алгоритм не останавливается на некоторых входных данных, то найдётся шаг, после которого конечное множество $G \cup Done$ содержит пару отмеченных многочленов f_1', f, для которых выполняется: f_1' было добавлено в $G \cup Done$ до f; f_1' – сигнатурный редуктор для f; $t_1 f_1'$ не удовлетворяет критериям F5 и Перезаписи, где $t_1 = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f_1')}$.*

Доказательство. Пусть f', f – многочлены из теоремы 6 и $t = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f')}$. Построенная выше теория о представлениях может быть рассмотрена применительно к случаю g равного f. Поскольку tf' – сигнатурный редуктор для f – мы имеем $\mathcal{S}(f')t \prec \mathcal{S}(f)$ и следствие 22 может быть применено для нахождения сигнатурного редуктора $t_1 f_1'$ для f, который не удовлетворяет критериям. □

Теорема 24. *Алгоритм F5, описанный в [1], останавливается на любых входных данных.*

Доказательство. Предположим отсутствие остановки и рассмотрим многочлены f_1', f из теоремы 23. Вызов `TopReduction` после которого f был добавлен в *Done* вернул многочлен f как первую половину возвращаемого значения, то есть предшествующий вызов `IsReducible` вернул пустое множество. Значит, для всех многочленов в $G \cup Done$ хотя бы одно из условий (a) - (d) не выполнилось. Это невозможно, поскольку для f_1' их выполнение следует из утверждения теоремы 23. □

Выводы

Данная работа показывает, что исходный алгоритм F5 завершается на любых однородных входных данных, не ссылаясь на остановку других алгоритмов. При этом не даётся никакого ограничения на количество операций. Простейшее доказательство остановки алгоритма Бухбергера основано на свойстве Нётеровости и тоже не даёт такого ограничения. Однако доказательство остановки, приведённое здесь, значительно отличается по структуре от доказательства остановки алгоритма Бухбергера, поэтому не может быть использовано для сравнения эффективности алгоритма F5 с алгоритмом Бухбергера. В отличие от этого, остановка изменённых вариаций алгоритма F5 в работах [3,4,5] показывается методом, близким к доказательству остановки алгоритма Бухбергера, за счёт чего остаётся возможность сравнения их эффективности с алгоритмом Бухбергера.

С точки зрения практической компьютерной алгебры существует вопрос эффективности изменённых вариаций по сравнению с исходным F5. Изменённые вариации могут проводить больше времени в дополнительных проверках, инициирующих остановку. Но в некоторых случаях возможна ситуация, когда за счёт этих проверок остановка в изменённых версиях инициируется раньше, чем в исходной, за счёт чего изменённые версии проводят меньше редукций. Поэтому становится возможным, что на некоторых входных данных быстрее оказывается исходный алгоритм, а на других – модифицированный. Экспериментально измеренные времена работы в Таблице 1 работы [3] показывают, что оба случая действительно встречаются на практике, но разница во временах работы незначительна. Поэтому, хотя данная работа и показывает, что останавливаются не только модифицированные версии, но и исходный алгоритм, вопрос сравнения эффективности версий остаётся открытым.

В доказательстве используются три свойства исходного алгоритма F5, которые отсутствуют или опциональны в других F5-подобных алгоритмах: однородность входных многочленов, наличие критерия перезаписи и совпадения порядка на мономах $<$ с порядком на сигнатурах $\prec$. Возможность расширения доказательства на F5-подобные алгоритмы, не обладающие этими свойствами, остаётся открытым. Автор предполагает, что доказательство может быть модифицировано таким образом, что зависимость от первых двух свойств исчезнет, однако от третьего избавиться не получится, поскольку оно является ключевым моментом при получении противоречия на основе результата теоремы 6.

Автор благодарит Christian Eder, Jean-Charles Faugère, Amir Hashemi, John Perry, Till Stagers и Алексея Зобнина за их работы и комментарии, воодушевившие автора на исследования в этой области. Спасибо!

СПИСОК ЛИТЕРАТУРЫ

1. *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. 2002. 75–83.
2. *Stegers T.* Faugere's F5 Algorithm Revisited. // IACR Cryptology ePrint Archive. 2006. **404**.
3. *Eder C., Gash J., Perry J.* Modifying Faugère's F5 Algorithm to ensure termination // ACM Commun. Comput. Algebra. 2011. **45**, №1/2. 70–89.
4. *Ars G.* Applications des bases de Gröbner à la cryptographie. Rennes, France: Université de Rennes 1, 2005.
5. *Gash J. M.* On efficient computation of Gröbner bases. Indianapolis, IN, USA: Indiana University, 2008.
6. *Hashemi A., Ars G.* Extended F5 criteria // J. Symb. Comput. 2010. **45**, № 12. 1330–1340.
7. *Зобнин А. И.* Обобщение алгоритма F5 вычисления базиса Грёбнера полиномиальных идеалов // Программирование. 2010. № 2. 21–30.
8. *Pan S., Hu Y., Wang B.* The Termination of Algorithms for Computing Gröbner Bases // ArXiv e-prints. 2012. **1202**, №3524.
9. *Huang L.* A new conception for computing Gröbner basis and its applications // ArXiv e-prints. 2010. **1012**, №5425.
10. *Eder C., Perry J.* F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases // J. Symb. Comput. 2010. **45**, № 12. 1442–1458.
11. *Arri A., Perry J.* The F5 Criterion revised // J. Symb. Comput. 2011. **46**, № 9. 1017–1029.
12. *Baader F., Nipkow T.* Term Rewriting and All That. United Kingdom: Cambridge University Press, 1998.