Простой итеративный алгоритм вычисления базисов Грёбнера, основанный на сигнатурах.

11 апреля 2012 г.

1 Обозначения

Будем придерживаться следующих обозначений, соответствующих большинству статей по алгоритмам, использующим отмеченные полиномы.

k - поле, над которым рассматривается кольцо многочленов $P = k[x_1, \ldots, x_n]$

$\mathbb{T}$ - множество мономов

$\prec$ - произвольный мономиальный порядок на $\mathbb{T}$, используемый как для определения старшего монома многочлена, так и для упорядочивания сигнатур.

$\{g_1, \ldots, g_m\}$ - базис Грёбнера идеала $I_0 = \langle g_1, \ldots, g_m \rangle$, полученный на предыдущем шаге алгоритма. Дополнительно предполагается, что $g_i \notin k$, т.е. $I_0 \neq P$.

f - многочлен, добавляемый на новой итерации, $I = \langle f, g_1, \ldots, g_m \rangle$ - идеал, для которого требуется найти базис.

$\mathrm{HM}(p) \in \mathbb{T}$ - старший моном многочлена p. Ненулевой для всех многочленов кроме нулевого. $\mathrm{HM}(0) = 0$.

$\mathrm{HC}(p) \in P$ - старший коэффициент многочлена p. Ненулевой для всех многочленов кроме нулевого. $\mathrm{HC}(0) = 0$.

$\mathrm{LCM}(t_1, t_2)$ - наименьшее общее кратное мономов t_i.

$h = (\sigma, p)$ - отмеченный многочлен. Сигнатура это просто моном $\sigma \in \mathbb{T}$ или 0. Она не имеет индекса, поскольку рассматривается итеративный алгоритм.

На отмеченном многочлене заданы $\mathrm{HM}(h) = \mathrm{HM}(p)$, $\mathrm{HC}(h) = \mathrm{HC}(p)$, $\mathcal{S}(h) = \sigma$.

Умножение отмеченного многочлена h на моном t даёт отмеченный многочлен $th = (t\sigma, tp)$.

Будем говорить, что h_1 есть результат редукции h'_1 по h_2 с сохранением сигнатуры, если:

$$h_1 = K_1(h'_1 + K_2 t h_2), K_i \in k, t \in \mathbb{T}$$

$$\mathcal{S}(th_2) \prec \mathcal{S}(h'_1), \mathrm{HM}(h_1) \prec \mathrm{HM}(h'_1) = \mathrm{HM}(th_2)$$

$$\mathrm{HC}(h_1) = 1 \vee h_1 = 0$$

$<_{gwv}$ - частичный порядок на отмеченных многочленах, определяемый как $h_1 = (\sigma_1, p_1) <_{gwv} h_2 = (\sigma_2, p_2)$, если и только если $\mathrm{HM}(p_1)\sigma_2 \prec \mathrm{HM}(p_2)\sigma_1$. Условно (хотя и не строго) это можно интерпретировать как $h_1 <_{gvw} h_2 \iff \frac{\mathrm{HM}(h_1)}{\mathcal{S}(h_1)} \prec \frac{\mathrm{HM}(h_2)}{\mathcal{S}(h_2)}$.

2 Алгоритм

Описание типов переменных:

- Входные данные: базис Грёбнера, представленный многочленами $\{g_1, \ldots, g_m\}$, многочлен f.
- Внутренние переменные:
  - R и B - множества отмеченных многочленов
  - (σ, p') - отмеченный многочлен, редуцируемый на текущем шаге, (σ, p) – он же после редукции
  - r - отмеченный многочлен при переборе множества R
  - b - отмеченный многочлен при переборе множества B
- Результат: по завершении алгоритма R содержит базис Грёбнера.

Алгоритм SimpleSignatureGroebner:

1. $R \leftarrow \{(0, g_1), (0, g_2), ..., (0, g_n)\}$
2. $B \leftarrow \{\}$
3. $(\sigma, p') \leftarrow (1, f)$
4. $p \leftarrow$Редуцировать_с_учётом_сигнатур(p', σ, R)
5. $R \leftarrow R \cup \{(\sigma, p)\}$
6. **if** $p \neq 0$
   (a) **for** $\{r \in R | r <_{gwv} (\sigma, p), \mathrm{HM}(r) \neq 0\}$
      i. $B \leftarrow B \cup \{\frac{\mathrm{LCM}(\mathrm{HM}(r), \mathrm{HM}(p))}{\mathrm{HM}(r)} r\}$
   (b) **for** $\{r \in R | r >_{gwv} (\sigma, p)\}$
      i. $B \leftarrow B \cup \{\frac{\mathrm{LCM}(\mathrm{HM}(r), \mathrm{HM}(p))}{\mathrm{HM}(p)} (\sigma, p)\}$
7. $B \leftarrow B \setminus \{b \in B | \exists r \in R,\, r <_{gwv} b \wedge \mathcal{S}(r) | \mathcal{S}(b)\}$
8. **if** $B = \emptyset$
   (a) **return** R
9. $(\sigma, p') \leftarrow$ элемент B с минимальной сигнатурой
10. **goto** 4

Function Редуцировать_с_учётом_сигнатур(p, σ, R)

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

3 Теория

Лемма 1. *Пусть h_1 и h_2 – отмеченные многочлены,$t \in \mathbb{T}$. Тогда $h_1 >_{gvw} h_2 \iff h_1 >_{gvw} th_2$.*

Выводится из того, что домножение на t одного из сравниваемых отмеченных многочленов приводит к домножению на t обоих частей в определении $>_{gvw}$.

Лемма 2. *Пусть h_1 и h_2 – отмеченные многочлены,* $\mathrm{HM}(h_1) | \mathrm{HM}(h_2)$. *Тогда редукция второго по первому сохраняет сигнатуру если и только если $h_1 >_{gvw} h_2$.*

Следует из того что оба утверждения равносильны $\mathcal{S}(h_2) \succ \mathcal{S}(h_1)\frac{\mathrm{HM}(h_2)}{\mathrm{HM}(h_1)}$.

Лемма 3. *Пусть h_2 – отмеченный многочлен, полученный из h_1 редукцией с сохранением сигнатур. Тогда $h_1 >_{gvw} h_2$.*

Следует из того что их сигнатуры равны, а HM уменьшается при редукции: $\mathrm{HM}(h_2) \prec \mathrm{HM}(h_1)$.

Определение 4. При фиксированных f и $\{g_1, \ldots, g_m\}$ отмеченный многочлен $h = (\sigma, p)$ называется *допустимым*, если он удовлетворяют соотношению $p = uf \mod I_0$ для некоторого многочлена u, такого что $\mathrm{HM}(u) = \sigma$.

Далее многочлены f и $\{g_1, \ldots, g_m\}$ предполагаются фиксированными, поэтому понятие допустимости используется без их явного указания.

Лемма 5. *Все отмеченные многочлены, порождаемые алгоритмом допустимы.*

Если $p_0 \in I_0$, то, поскольку $p_0 = 0f \mod I_0$, отмеченный многочлен $(0, p_0)$ допустим. Поэтому допустимы многочлены, изначально добавляемые в R. Поскольку $f = 1f \mod I_0$ допустимым является отмеченный многочлен $(1, f)$. Все остальные многочлены, присутствующие в алгоритме порождаются или домножением на моном допустимого многочлена, присутствующего в R или редукцией с сохранением сигнатуры. Обе эти операции сохраняют свойство допустимости.

Лемма 6. *Пусть $h_1 <_{gvw} h_2$ отмеченные многочлены. Тогда для любого отмеченного многочлена h_3 с ненулевой сигнатурой выполняется по крайней мере одно из двух неравенств $h_1 <_{gvw} h_3$ и $h_3 <_{gvw} h_2$.*

Из условия леммы известно, что

$$\mathrm{HM}(h_1)\,\mathcal{S}(h_2) \prec \mathrm{HM}(h_2)\,\mathcal{S}(h_1) \tag{3.1}$$

откуда $\mathrm{HM}(h_2) \neq 0$. Поэтому, если $\mathrm{HM}(h_3) = 0$, имеем $h_3 <_{gvw} h_2$. Иначе $\mathrm{HM}(h_3) \neq 0$ и можно домножить неравенство 3.1 на ненулевой элемент $\mathrm{HM}(h_3)\,\mathcal{S}(h_3)$:

$$\mathrm{HM}(h_3)\,\mathcal{S}(h_3)\,\mathrm{HM}(h_1)\,\mathcal{S}(h_2) \prec \mathrm{HM}(h_3)\,\mathcal{S}(h_3)\,\mathrm{HM}(h_2)\,\mathcal{S}(h_1).$$

Поэтому моном $\mathrm{HM}(h_3)^2\,\mathcal{S}(h_2)\,\mathcal{S}(h_1)$ окажется или $\succ$ левого или $\prec$ правого монома последнего неравенства, что после сокращения даст неравенство, эквивалентное утверждению леммы.

3.1 Остановка алгоритма

Лемма 7. *В любой момент работы алгоритма любой отмеченный многочлен из B может быть редуцирован с сохранением сигнатуры по некоторому элементу R.*

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

В последующих утверждениях понятие делимости мономов расширяется на случай нулевых – будем полагать что нулевой моном делит нулевой моном и только его. Отметим также, что нулевыми сигнатурами в R обладают лишь элементы $(0, g_1), (0, g_2), ..., (0, g_n)$, нулевым старшим мономом могут обладать многочлены добавляемые в R после редукции. При этом нулевые многочлены с одновременно нулевой сигнатурой не встречаются. В множестве B нет многочленов ни с нулевой сигнатурой, ни с нулевым старшим мономом.

Лемма 8. *До редукции многочлена p', т.е. на этапе 4 любой итерации алгоритма, сигнатуры элементов $\{r \in R | r <_{gvw} (\sigma, p')\}$ не делят σ.*

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 (σ, p') был бы убран из B в предыдущей итерации на шаге 7.

Лемма 9. *После редукции многочлена p' до p, т.е. на этапе 5 любой итерации алгоритма, старшие мономы элементов $\{r \in R | r >_{gvw} (\sigma, p)\}$ не делят* $\mathrm{HM}(p)$.

Утверждение леммы вытекает из того, что цикл редукций в функции Редуцировать_с_учётом_сигнатур останавливается только тогда, когда достигается p, для которого такие элементы в R не существуют.

Лемма 10. *После редукции многочлена p' до p, т.е. на этапе 5 любой итерации алгоритма, элементы R не могут одновременно иметь старшие мономы, делящие* $\mathrm{HM}(p)$ *и сигнатуры, делящие σ.*

В силу леммы 7 будет произведена хотя бы одна редукция p', поэтому $(\sigma, p') >_{gvw} (\sigma, p)$. Отсюда, для элементов $\{r \in R | \mathcal{S}(r) \neq 0\}$ имеем $r >_{gvw} (\sigma, p)$ или $r <_{gvw} (\sigma, p')$. Для $\{r \in R | \mathcal{S}(r) = 0\}$ имеем $r >_{gvw} (\sigma, p)$, поскольку $\sigma \neq 0$ и $\mathrm{HM}(r) \neq 0$. Выполнение одного из неравенств позволяет применить одну из лемм 8 и 9.

Теорема 11. *Описанный алгоритм останавливается на любых входных данных*

Для доказательства остановки нужно показать что оба цикла, присутствующие в алгоритме могут выполняться лишь конечное число раз. Цикл, находящийся в процедуре редукции выполняется конечное число раз, поскольку на каждой итерации уменьшается старший моном многочлена p, а бесконечной убывающей последовательности мономов не существует.

Для основного цикла алгоритма рассмотрим два множества: $L = \{\sigma | (\sigma, 0) \in R\} \subset \mathbb{T}$ и $N = \{(\sigma, t) | \exists (\sigma, p) \in R, t = \mathrm{HM}(p)\} \subset \mathbb{T} \times \mathbb{T}$. Основной цикл алгоритма на шаге 5 добавляет элемент в R, чему соответствует добавление элемента в одно из этих множеств. В силу леммы 10 добавляемые в эти множества элементы расширяют на каждом шаге конусы-идеалы $\langle L \rangle$ и $\langle N \rangle$, что по лемме Диксона может происходить лишь конечное число раз.

3.2 Корректность

Определение 12. *S-представлением* отмеченного многочлена h относительно множества отмеченных многочленов $R = \{r_i\}$ будем называть выражение $h = \sum_j K_j t_j r_{i_j}$, $K_j \in k, t_j \in \mathbb{T}, i_j \in \mathbb{N}$, такое что $\forall j\, \mathrm{HM}(h) \succcurlyeq \mathrm{HM}(t_j r_j), \mathcal{S}(h) \succcurlyeq \mathcal{S}(t_j r_j)$.

Лемма 13. *Если S-представление относительно множества $R = \{r_i\}$ имеет любой допустимый отмеченный многочлен h, то R – базис Грёбнера идеала I.*

Вытекает из того что для $p \in I$ можно взять некоторый допустимый $h = (\sigma, p)$ в S-представлении которого найдётся r_{i_j}, такой что $\mathrm{HM}(p) = t_j\, \mathrm{HM}(r_{i_j})$.

Для доказательства корректности алгоритма будем доказывать следующую теорему, из которой будет следовать корректность ответа в момент остановки алгоритма, поскольку при $B = \emptyset$ её инвариант станет выполняться для всех многочленов.

Теорема 14. *На каждой итерации алгоритма после шага 7 выполнен следующий инвариант: любой допустимый отмеченный многочлен, сигнатура которого $\prec$ чем сигнатуры всех элементов B имеет S-представление относительно R.*

Предположим, что утверждение теоремы неверно на определённой итерации алгоритма. Зафиксируем множества R и B на момент после шага 7 этой итерации, и все S-представления будем рассматривать относительно этого R. Пусть V – множество допустимых многочленов, для которых не выполняется инвариант, а $V_0 = \{v \in V | \mathcal{S}(v) = \sigma_0\}$ – подмножество с минимальной сигнатурой. Все элементы I_0 имеют S-представления относительно $\{(0, g_1), (0, g_2), ..., (0, g_n)\} \subset R$, поэтому $\sigma_0 \neq 0$. Выберем $v_0 = (\sigma_0, q_0) \in V_0$ – многочлен с наименьшим старшим мономом. Он не может быть редуцирован с сохранением сигнатуры относительно R, поскольку иначе результат его редукции v_1 являлся бы элементом V_0 с $\mathrm{HM}(v_1) \prec \mathrm{HM}(v_0)$.

Рассмотрим множество $\{r \in R | \mathcal{S}(r) | \sigma_0\}$. Поскольку $\sigma_0 \neq 0$ оно содержит добавленный на первой итерации элемент r с $\mathcal{S}(r) = 1$ и значит непусто. Рассмотрим его$<_{gvw}$-минимальный элемент r_0 и обозначим $t_0 = \frac{\sigma_0}{\mathcal{S}(r_0)}$. Предположим, что отмеченный многочлен $t_0 r_0 = (\sigma_0, t_0 p_0)$ может быть редуцирован с сохранением сигнатуры относительно некоторого$r_1 \in R$. Отсюда следует, что $r_1 >_{gvw} r_0$, а также что они не нулевые. Значит на той же итерации, когда в R был добавлен последний из $\{r_0, r_1\}$, в множество B был добавлен многочлен $t' r_0$, где $t' = \frac{\mathrm{LCM}(\mathrm{HM}(r_1), \mathrm{HM}(r_0))}{\mathrm{HM}(r_0)}$, причём $t' | t_0$. Отсюда $\mathcal{S}(t' r_0) | \mathcal{S}(t_0 r_0) = \sigma_0 \Rightarrow \mathcal{S}(t' r_0) \preccurlyeq \sigma_0$. Поскольку σ_0 – сигнатура v_0, для которого не выполняется инвариант, она $\prec$ чем сигнатуры элементов B – то есть $t' r_0$ был выкинут на шаге 7 одной из итераций, что влечёт существование $r_2 \in R, r_2 <_{gvw} t' r_0, \mathcal{S}(r_2) | \mathcal{S}(t' r_0)$. Это невозможно, поскольку влечёт $r_2 <_{gvw} r_0, \mathcal{S}(r_2) | \mathcal{S}(t_0 r_0) = \sigma_0$, что противоречит $<_{gvw}$-минимальности r_0.

Таким образом показано, что $t_0 r_0 = (\sigma_0, t_0 p_0)$ и $v_0 = (\sigma_0, q_0)$ не могут быть редуцированы относительно R с сохранением сигнатуры, при этом первый из них имеет тривиальное S-представление $t_0 r_0$, а второй не имеет S-представления. Из допустимости следует, что их разность $w = (\sigma_1, q_0 - K t_0 p_0)$ с некоторым коэффициентом $K \in k$ есть допустимый многочлен сигнатуры $\sigma_1 \prec \sigma_0$. $w \notin V$ в силу малости сигнатуры и имеет S-представление $w = \sum_j K_j t_j r_{i_j}$. Значит $\exists j | \mathrm{HM}(t_j r_{i_j}) = \mathrm{HM}(w), \mathcal{S}(t_j r_{i_j}) \preccurlyeq \mathcal{S}(w) = \sigma_1 \prec \sigma_0$. Если бы $\mathrm{HM}(w) \in \{\mathrm{HM}(v_0), \mathrm{HM}(t_0 r_0)\}$, то r_{i_j} был бы сохраняющим сигнатуру редуктором для v_0 или $t_0 r_0$, что невозможно. Значит старшие мономы сокращаются при вычитании, то есть $\mathrm{HM}(v_0) = \mathrm{HM}(t_0 r_0)$. Отсюда выводим S-представление: $v_0 = K t_0 r_0 + \sum_j K_j K t_j r_{i_j}$. Противоречие.

3.3 Связь с аналогами

Представленный алгоритм связан с двумя уже известными алгоритмами, основанных на сигнатурах – алгоритм G2V из работы **?**]и версию F5, опубликованную в работе **?**]. Оба они в определённом смысле являются модификациями простого F5, впервые представленного в работе **?**], 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 явного 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 критерию «primitve S-irreducble» 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.

Практически во всех алгоритмах, основанных на сигнатурах, в том числе в исходном алгоритме F5 применяется и более эффективный тип критериев: критерии отбрасывания S-пар, не требующие вычислений с многочленами для своей проверки – их отличия в различных алгоритмах подробно разобраны в работе **?**]. Однако вопрос об их сравнительной эффективности остаётся неясным как с теоретической, так и с эмпирической точек зрения – на различных примерах большее или меньшее преимущество могут иметь различные подходы.

В алгоритме SimpleSignatureGroebner используется подход, который можно интерпретировать как объединение всего вышеуказанного: S-полиномы не вычисляются явно и их отбрасывание осуществляется на основе критерия, записываемого в точности как критерий второго алгоритма. При этом для проверки на шаге 7 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 домножения всего многочлена на моном.Таким образом, для отброшенных элементов B не производится никаких операций, сложность которых пропорциональна длине многочлена. Заметим, что применяемое во втором алгоритме вычисление старшего монома S-пары в общем случае напротив имеет именно такую сложность. Таким образом, возможна реализация алгоритма SimpleSignatureGroebner, в которой все операции над многолченами производятся лишь для многочленов, попадающих в результирующее множество. Результирующее множество не является минимальным базисом Грёбнера, но при этом в определённом смысле минимально. Это минимальное множество, удовлетворяющее следующему определению:

Определение 15. Множество отмеченных многочленов называется *базисом S-представлений*, если относительно него любой допустимый отмеченный многочлен имеет S-представление.

Отсюда можно сделать вывод о некоторой полноте критериев отбрасывания: 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 базисом S-представлений. Из доказательств корректности многих алгоритмов, основанных на сигнатурах следует что их результат содержит такое

множество, поэтому представленный алгоритм является в указанном смысле оптимальным алгоритмом, находящим минимальный базис S-представлений для последовательности идеалов, расширяемых одним многочленом на каждом шаге.

Эта оптимальность не глобальна – могут существовать более эффективные модификации алгоритма, которые не просто убирают вычисления, а заменяют одни вычисления над полиномами другими. К примеру, вопрос наиболее эффективного способа выбора редуктора в процедуре Редуцировать_с_учётом_сигнатур является открытым. Представленный способ выбора, основанный на $<_{gvw}$-сравнении наиболее близок к способу выбора, применявшегося в оригинальном алгоритме F5.

Алгоритм был реализован на C++ с использованием функций ядра программного комплекса Singular 3-1-4 и открытых наработок Christian Eder 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 с другими алгоритмами вычисления базисов Грёбнера, реализованных Christian Eder подтвердили следующие теоретические соображения:

- Алгоритм SimpleSignatureGroebner корректно вычисляет базис Грёбнера
- Возвращаемое множество содержит не большее число многочленов, чем множество возвращаемое другими инкрементальными алгоритмами, про которые известно, что они возвращают S-представление
- Время работы алгоритма оказывается несколько меньше, чем у других инкрементальных алгоритмов, основанных на сигнатурах.

Список литературы