系 统 科 学 与 数 学
J. Sys. Sci. & Math. Scis.
29(9) (2009, 9), 1266–1277

布尔环上的分支 Gröbner 基算法

孙 瑶 王定康

(中国科学院数学与系统科学研究院数学机械化重点实验室，北京 100190)

摘要 众所周知 Gröbner 基在很多领域都有着十分重要的应用. 近些年来 Gröbner 基算法有了很大的改进，其中最著名的是 Faugère 提出的 F4 和 F5 算法. 这两个算法具有很高的效率但通常需要消耗大量的内存. 鉴于此，将给出一个布尔环上基于 zdd 数据结构的分支 Gröbner 基算法，该算法不仅可以大大降低对内存的消耗，还能有效的控制矩阵规模，从而提高算法的整体效率. 详细阐述并证明了算法的基本理论，介绍该分支算法的数据结构及分支策略. 最后通过实验数据可以发现，在很多例子中此算法都要优于 Magma 中的 F4 算法.

关键词 分支 Gröbner 基，布尔环， zdd 数据结构.

MR(2000) 主题分类号 12Y05

1 引 言

多项式系统求解是计算机代数中的一个基本问题，通过求解多项式系统我们可以解决很多领域中的实际问题，比如：力学，机械学，密码学等. 在所有求解多项式系统的方法之中，Gröbner 基和特征列方法都是我们熟知的最有效的方法. 在特征列方面，高小山的布尔环上的特征列算法在解决流密码等方面取得了很好的效果 [1−3]，同时也给我们的分支 Gröbner 基算法以很大启发. 相对于特征列算法，近些年来 Gröbner 基算法有了很大的改进.

Gröbner 基算法的改进可以分为 3 个方面，具体来说就是：数据结构，选择策略和减少冗余计算. 最早的 Gröbner 基算法是由 Buchberger 于 1965 年给出 [4]，之后他证明了该算法的终止性 [5] 并在 1979 年的文章中提出了三条改进标准 [6]，其中两条是减少冗余计算，另外一条是选择策略. 1983 年， Lazard 第一次提出用线性方法来计算 Gröbner 基的想法 [7]，随后 Mora 等人在此想法上给出借助阶梯形线性基来计算 Gröbner 基的算法 [8]. 与此同时，利用合冲去除冗余计算的算法由 Mora 等人于 1993 年提出 [9]，从而大大减少了不必要的计算，但该算法的效率不高. 最具重要意义的改进来自 1999 年 Faugère 提出的 F4 算法 [10]，该算法继承了 Larzard 的线性化想法，通过成熟的矩阵处理技术大大降低了约化多项式所消耗的时间. 3 年后， Faugère 又在 F5 算法中给出了两条新标准用于去除冗余计算 [11]，该标准是在 Mora 等人提出的合冲标准上的改进，可操作性更强，效率更高.

在算法的实现方面，目前世界上效率最高的是 Steel 在 Magma 上实现的 F4 算法 [12]. 由于 F5 算法对于多项式的约化顺序有特殊要求，某种程度上限制了线性代数技术的使用，使

*“973” 项目 (2004CB318000) 和国家自然科学基金 (NSFC60821002/F02) 资助.
收稿日期：2009-07-30.

矩阵的处理效率受到一些影响，因而除了 Faugère 的实现外，尚无法获得高效的 F5 算法. 然而从算法角度讲， F4 算法固然拥有很高的效率，但其对内存的消耗却是惊人的. 例如，用 F4 算法求解 HFE80 系统需要消耗近 16G 内存. 这是由于目前的 F4 算法几乎都需要保存计算过程中所生成的多项式，因而将消耗掉大量的存储空间以及数据读取时间. 此外，随着次数上升， F4 算法生成的矩阵规模将会迅速增大，从而导致算法整体的复杂度大幅度增加，特别的，当基域的特征为 0 时，多项式的系数会疾速膨胀，这也使得矩阵变得更难处理.

鉴于此，我们的算法主要是在节省内存空间以及控制矩阵规模两方面进行改进. 首先，在算法的理论方面，我们修改并证明了 F5 算法所提出的两条准则，降低了算法对输入多项式顺序的依赖；其次，针对布尔环的特性，我们采用了 zdd 数据结构，使得多项式可以以一种更节省的方式进行存储，从而大大降低了算法对内存的消耗；最后，通过引入分支，我们可以有效的控制矩阵规模，提高算法的整体效率.

算法所采用的 zdd 数据结构 (zero-suppressed binary decision diagrams) 是由 bdd 改进而来，bdd 用于表示布尔函数，而 zdd 则主要用于表示布尔多项式 [13,14]. 2007 年，Brickenstein 最先使用 zdd 数据结构来计算布尔环中的 Gröbner 基，并在纯字典序下取得了较好的结果 [15]，然而，该程序对于计算全次逆字典序的 Gröbner 基效果不甚理想. 考虑到对于同一理想，计算全次逆字典序的 Gröbner 基复杂度更低，因而，我们在实现算法时，对多项式的 zdd 表达形式进行了适当修改，从而使得我们的算法更适于计算全次逆字典序的 Gröbner 基. 此外，前面提到的高小山提出的布尔环上的特征列算法也利用了 zdd 数据结构.

本文的主要内容如下：第 2 节介绍我们 Gröbner 基算法的基本理论；第 3 节介绍算法中多项式的 zdd 表达形式；第 4 节介绍分支算法的实现以及分支算法的分支策略；第 5 节是有关算法实现的一些细节问题；第 6 节给出一些实验数据；第 7 节总结全文.

2 Gröbner 基算法理论

设 F_2 是只包含元素 0,1 的有限域， $X = x_1, x_2, \cdots, x_n$ 是变量集合，多项式集合 $H = \{x_1^2 + x_1, x_2^2 + x_2, \cdots, x_n^2 + x_n\}$ 为域多项式集合，简称为域多项式，则环 $\mathrm{R}_2 = \mathrm{F}_2[X]/(H)$ 是一个布尔环，而环 R_2 中的元素称为布尔多项式.

布尔环中的 Gröbner 基有不同的定义方式，从算法角度出发，我们采用如下定义. 考虑正则函数 $\sigma : \mathrm{F}_2[X] \longrightarrow \mathrm{R}_2$, 函数 σ 是满射，因而 R_2 中任何元素在 $\mathrm{F}_2[X]$ 中都有原像. 一组布尔多项式集合 $F \subset \mathrm{R}_2$, $\sigma^{-1}(F)$ 是 F 在多项式环 $\mathrm{F}_2[X]$ 中的原像，设多项式集合 G 为 $\sigma^{-1}(F) \cup H$ 在 $\mathrm{F}_2[X]$ 中的 Gröbner 基，则集合 $\sigma(G)$ 称为 F 在布尔环 R_2 中的 Grönber 基. 因而，计算 F 在布尔环 R_2 中的 Grönber 基的问题，可以转化成计算 $\sigma^{-1}(F) \cup H$ 在 $\mathrm{F}_2[X]$ 中的 Gröbner 基的问题. 下面我们给出的算法就是在多项式环 $\mathrm{F}_2[X]$ 中计算 $\sigma^{-1}(F) \cup H$ 的 Gröbner 基的算法. 由于环 R_2 与 $\mathrm{F}_2[X]$ 中的元素可以容易的实现相互转换，因而为了叙述的方便，在不引起歧义的情况下，我们将不再区分这两个环中的元素，并通称其为多项式.

考虑到 F5 算法提出的两条标准可以去除绝大多数冗余计算，大大提高算法效率，因此我们的 Gröbner 基算法采用了 F5 算法的基本结构. 不过，通过大量的实验数据我们发现，F5 算法对于初始多项式的输入顺序依赖十分明显，相同多项式组的不同输入顺序会使计算时间大相径庭. 为了减轻这种排列顺序对算法的影响，我们在实现算法时修改了 F5 算法中签名 (signature) 的排列规则，同时相应修改并证明了 F5 算法中的两条标准，该证明过程同

样可以用于证明 F5 算法中的两条原有准则. 在介绍算法前, 需要引入一些必要的符号.

2.1 符 号

令 N 为所有非负整数的集合, T 为所有关于变量 X 的幂积集合, 即 $T=\{x_1^{\alpha_1}x_2^{\alpha_1}\cdots x_n^{\alpha_n}|\ \alpha_i\in N,\ i=1,2,\cdots,n\}$. 设 $\prec$ 是定义在 T 上的一个单项式序, 在我们的算法中, 取 $\prec$ 为全次逆字典序, 且有 $x_1\prec x_2\prec\cdots\prec x_n$. 对于 $t=x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}\in T$, 定义 t 的次数为: $\deg(t)=\sum\limits_{i=1}^{n}\alpha_i$. 对于多项式 $0\neq f\in \mathrm{F}_2[X]$, 有 $f=\sum x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$, 定义 f 的次数为: $\deg(f)=\max\{\alpha_1+\alpha_2+\cdots+\alpha_n\}$, 定义 f 的首项 (leading monomial) 为: $\mathrm{lm}(f)=\max_{\prec}\{x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}\}$.

2.2 标记多项式

为了建立理想中多项式之间的联系, 以减少不必要的计算, 需要对普通多项式附加一些信息, 为此我们引入标记多项式 (labeled polynomial) 的概念.

定义 2.1 令 $L=T\times\mathrm{F}_2[X]\times N\times\mathrm{F}_2[X]\times N$, 标记多项式是一个五元组 $\mathcal{G}=(x^\alpha,f,i,g,k)\in L$. 标记多项式 $\mathcal{G}$ 的签名 (signature) 定义为: $S(\mathcal{G})=x^\alpha$; $\mathcal{G}$ 的初式 (initial) 定义为: $\mathrm{init}(\mathcal{G})=f$; $\mathcal{G}$ 的扩展签名 (extended signature) 定义为: $ES(\mathcal{G})=x^\alpha\mathrm{lm}(f)$; $\mathcal{G}$ 的序 (index) 定义为: $\mathrm{index}(\mathcal{G})=i$; $\mathcal{G}$ 的多项式 (polynomial) 定义为: $\mathrm{poly}(\mathcal{G})=g$; $\mathcal{G}$ 的序号 (number) 定义为: $\mathrm{num}(\mathcal{G})=k$.

标记多项式与普通多项式不同的地方就在于, 它除了包含原来的多项式外, 还包含了很多其它相关信息. 为了区别多项式与标记多项式, 在本文中, 我们用小写字母表示多项式, 如 f,g,h; 用花体字母表示标记多项式, 如 $\mathcal{F},\mathcal{G},\mathcal{H}$. 下面定义标记多项式的排序和运算.

定义 2.2 对于标记多项式 $\mathcal{F},\mathcal{G}\in L$, 定义: $\mathcal{F}\prec_{es}\mathcal{G}$(或者 $\mathcal{G}\succ_{es}\mathcal{F}$), 如果满足下面三个条件之一: (1) $ES(\mathcal{F})\prec ES(\mathcal{G})$. (2) $ES(\mathcal{F})=ES(\mathcal{G})$ 且 $\mathrm{index}(\mathcal{F})>\mathrm{index}(\mathcal{G})$. (3) $ES(\mathcal{F})=ES(\mathcal{G})$, $\mathrm{index}(\mathcal{F})=\mathrm{index}(\mathcal{G})$ 且 $\mathrm{num}(\mathcal{F})>\mathrm{num}(\mathcal{G})$.

定义 2.3 若 $u\in T$ 是一个单项式, $\mathcal{F}=(x^\alpha,f_i,i,f,k)\in L$ 是一个标记多项式, 则单项式与标记多项式的积定义为 $u\mathcal{F}=(ux^\alpha,f_i,i,uf,k)$.

定义 2.4 已知标记多项式 $\mathcal{F}=(x^\alpha,f_i,i,f,k)$ 和 $\mathcal{G}=(x^\beta,f_j,j,g,l)$. 令 $\mathcal{H}=\max_{\prec_{es}}\{\mathcal{F},\mathcal{G}\}$, 则标记多项式 $\mathcal{F}$ 与 $\mathcal{G}$ 的和定义为 $\mathcal{F}+\mathcal{G}=(\mathrm{S}(\mathcal{H}),\mathrm{init}(\mathcal{H}),\ \mathrm{index}(\mathcal{H}),f+g,\mathrm{num}(\mathcal{H}))$.

2.3 算法标准

定义 2.5 对于标记多项式 $\mathcal{F}\in L$, 及标记多项式集合 $B\subset L$. 若存在 $\mathcal{G}\in B$ 和单项式 $u\in T$, 满足: $\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=u\mathrm{lm}(\mathrm{poly}(\mathcal{G}))$ 且 $\mathcal{F}\succ_{es}u\mathcal{G}$, 则称 $\mathcal{F}$ 关于标记多项式集合 B 是可约的; 否则, 称 $\mathcal{F}$ 关于 B 不可约.

定义 2.6 若 $\mathcal{F}\in L$ 关于标记多项式集合 $B\subset L$ 是可约的, 则存在 $\mathcal{G}\in B$ 和单项式 $u\in T$, 满足: $\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=u\mathrm{lm}(\mathrm{poly}(\mathcal{G}))$ 且 $\mathcal{F}\succ_{es}u\mathcal{G}$. 记 $\mathcal{F}'=\mathcal{F}-u\mathcal{G}$, 我们称 $\mathcal{F}\longrightarrow_B\mathcal{F}'$ 为一次约化过程. 若 $\mathcal{F}'$ 仍然关于 B 可约, 则重复上面的约化过程. 若有 $\mathcal{F}\longrightarrow_B\mathcal{F}'\longrightarrow_B\cdots\longrightarrow_B\mathcal{F}^*$ 且 $\mathcal{F}^*$ 关于 B 不可约, 我们称 $\mathcal{F}$ 可以被 B 约化为 $\mathcal{F}^*$, 并记为: $\mathcal{F}\longrightarrow_B^*\mathcal{F}^*$.

容易证明, 任何约化过程都可以在有限步内终止. 若有 $\mathcal{F}\longrightarrow_B^*\mathcal{F}^*$, 根据约化定义可知, 存在 $p_1,p_2,\cdots,p_t\in\mathrm{F}_2[X]$, $\mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_t\in B$, 使得: $\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t+\mathcal{F}^*$, $\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i, i=1,2,\cdots,t$.

定义 2.7 令 $P=T\times L\times T\times L$. 定义标记多项式 $\mathcal{F},\mathcal{G}\in L$ 的 s 对和 s 多项式分别为

$\mathrm{s}(\mathcal{F},\mathcal{G})=\mathrm{s}(\mathcal{G},\mathcal{F})=(u,\mathcal{F},v,\mathcal{G})\in P$ 和 $\mathrm{spoly}(\mathcal{F},\mathcal{G})=\mathrm{spoly}(\mathcal{G},\mathcal{F})=u\mathcal{F}-v\mathcal{G}$. 其中，$u,v$ 满足：$\mathrm{lcm}(\mathrm{lm}(\mathrm{poly}(\mathcal{F})),\mathrm{lm}(\mathrm{poly}(\mathcal{G})))=\mathrm{lm}(u\mathrm{poly}(\mathcal{F}))=\mathrm{lm}(v\mathrm{poly}(\mathcal{G}))$ 且 $u\mathcal{F}\succ_{es}v\mathcal{G}$.

定义　2.8　已知两个 s 对：$\mathrm{s}(\mathcal{F},\mathcal{G})=(u,\mathcal{F},v,\mathcal{G})$ 和 $\mathrm{s}(\mathcal{P},\mathcal{Q})=(r,\mathcal{P},t,\mathcal{Q})$. 定义：$\mathrm{s}(\mathcal{F},\mathcal{G})\succ_{es}\mathrm{s}(\mathcal{P},\mathcal{Q})$ (或者 $\mathrm{s}(\mathcal{P},\mathcal{Q})\prec_{es}\mathrm{s}(\mathcal{F},\mathcal{G})$), 如果有 $u\mathcal{F}\succ_{es}r\mathcal{P}$, 或者 $ES(u\mathcal{F})=ES(r\mathcal{P})$, $\mathrm{index}(u\mathcal{F})=\mathrm{index}(r\mathcal{P})$, $\mathrm{num}(u\mathcal{F})=\mathrm{num}(r\mathcal{P})$ 且 $v\mathcal{G}\succ_{es}t\mathcal{Q}$.

有了以上定义，我们便可以给出算法的基本流程

输入　$F=(f_1,f_2,\cdots,f_m)\subset \mathrm{F}_2[X]$ 是一组有序多项式集合.

输出　$GB\subset \mathrm{F}_2[X]$ 是理想 $I=\langle F\rangle$ 的 Gröbner 基.

1)　初始多项式 f_i 对应的标记多项式记为 $\mathcal{F}_i:=(1,f_i,i,f_i,i)$, $i=1,2,\cdots,m$;

2)　$B:=\{\mathcal{F}_i|i=1,2,\cdots,m\}$, 序号 $k:=m$, $CP:=\{\mathrm{s}(\mathcal{F}_i,\mathcal{F}_j)|i,j=1,2,\cdots,k\}$;

3)　while $CP\neq\phi$

　　$c:=\min_{\prec_{es}}\{CP\}$.

　　计算 c 所对应的 s 多项式，并用 B 约化成 $\mathcal{F}$.

　　$k:=k+1$, $\mathrm{Num}(\mathcal{F}):=k$,

　　$B:=B\cup\{\mathcal{F}\}$, $CP:=CP\cup\{\mathrm{s}(\mathcal{F}_i,\mathcal{F})|i=1,2,\cdots,k-1\}$.

4)　输出 B.

注　在算法的输入 F 中包含域多项式 H, 为了提高算法效率，假设域多项式的序 (index) 大于 F 中非域多项式的序.

以上算法与 Buchberger 算法没有本质区别，所不同的是这个算法将 Buchberger 算法中的多项式换成了标记多项式. 标记多项式可以提供更多有关多项式的相关信息，利用这些信息便可以去除掉几乎所有可以被约化成 0 的 s 对. 从算法中可以看到，标记多项式的序号表示该标记多项式加入集合 B 的顺序，序号越大，标记多项式加入的越晚，同时也表明其越新. 此外，算法还规定了 s 对的处理顺序，即按照关于 $\prec_{es}$ 的从小到大的顺序进行处理，这个顺序对于算法标准的证明十分重要.

接下来我们给出两条算法的标准用于除去冗余的 s 对，该标准是由 F5 算法中的两条标准修改而来. 由于只是修改了标签 (signature) 的排序，因而这两条标准的证明过程完全可以用于证明 F5 算法的两条标准. 以下标准与证明中的符号将沿用前面算法流程中的符号.

1)　合冲 (syzygy) 标准　已知 s 对：$\mathrm{s}(\mathcal{G}_1,\mathcal{G}_2)=(u_1,\mathcal{G}_1,u_2,\mathcal{G}_2)$, $u_1,u_2\in T$, $\mathcal{G}_1,\mathcal{G}_2\in B$. 对于 $u_k\mathcal{G}_k$, 其中 $k=1$ 或 2, 若存在初始多项式 f_j, 满足 $\mathrm{index}(u_k\mathcal{G}_k)\leq j\leq m$ 且 $\mathrm{lm}(f_j)|S(u_k\mathcal{G}_k)$, 则 s 多项式 $\mathrm{spoly}(\mathcal{G}_1,\mathcal{G}_2)=u_1\mathcal{G}_1-u_2\mathcal{G}_2$ 可以被 B 约化成 0.

2)　重写 (rewritten) 标准　已知 s 对：$\mathrm{s}(\mathcal{G}_1,\mathcal{G}_2)=(u_1,\mathcal{G}_1,u_2,\mathcal{G}_2)$, $u_1,u_2\in T$, $\mathcal{G}_1,\mathcal{G}_2\in B$. 对于 $u_k\mathcal{G}_k$, 其中 $k=1$ 或 2, 若存在标记多项式 $\mathcal{F}\in B$ 以及单项式 $v\in T$, 满足 $S(u_k\mathcal{G}_k)=S(v\mathcal{F})$, $\mathrm{index}(\mathcal{G}_k)=\mathrm{index}(\mathcal{F})$ 且 $\mathrm{num}(\mathcal{G}_k)<\mathrm{num}(\mathcal{F})$, 则 s 多项式 $\mathrm{spoly}(\mathcal{G}_1,\mathcal{G}_2)=u_1\mathcal{G}_1-u_2\mathcal{G}_2$ 可以被 B 约化成 0.

为了证明这两条标准，我们首先需要下面的核心引理.

引理　2.9　已知当前的标记多项式集合 B, 以及当前需要处理的 s 对为 $\mathrm{s}(\mathcal{F},\mathcal{F}')=(u,\mathcal{F},u',\mathcal{F}')$, 其中 $u,u'\in T$, $\mathcal{F},\mathcal{F}'\in B$. 若存在 $p_1,p_2,\cdots,p_t\in \mathrm{F}_2[X]$, $\mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_t\in B$, 使得：$u\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$ 且 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i$, $i=1,2,\cdots,t$. 则存在标记多项式 $\mathcal{H}\in B$, 以及单项式 $v\in T$, 满足：$u\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=v\mathrm{lm}(\mathrm{poly}(\mathcal{H}))$ 且 $u\mathcal{F}\succ_{es}v\mathcal{H}$. 更进一步，s 多项式 $\mathrm{spoly}(\mathcal{F},\mathcal{F}')$ 可以被 B 约化成 0.

证 首先，已知 $\mathrm{s}(\mathcal{F},\mathcal{F}')$ 为当前需要处理的 s 对，结合算法对于 s 对的处理顺序可知，对于任意 $\mathcal{G},\mathcal{G}'\in B$, 若 $\mathrm{s}(\mathcal{G},\mathcal{G}')\prec_{es}\mathrm{s}(\mathcal{F},\mathcal{F}')$, 则 s 对 $\mathrm{s}(\mathcal{G},\mathcal{G}')$ 已经被处理过，无论当时的处理结果如何， s 多项式 $\mathrm{spoly}(\mathcal{G},\mathcal{G}')$ 均可被当前的标记多项式集合 B 约化成 0.

接下来我们证明，当条件 $u\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$ 且 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t$ 满足时，存在 $v\in T$ 以及 $\mathcal{H}\in B$, 使得： $u\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=v\mathrm{lm}(\mathrm{poly}(\mathcal{H}))$ 且 $u\mathcal{F}\succ_{es}v\mathcal{H}$.

一方面，若不存在 $j\in\{1,2,\cdots,t\}$ 使得 $\mathrm{lm}(p_j\mathrm{poly}(\mathcal{G}_j))\succ\mathrm{lm}(u\mathrm{poly}(\mathcal{F}))$. 则根据等式的性质可知，必存在 $\mathcal{G}_k,\ k\in\{1,2,\cdots,t\}$, 满足 $u\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=\mathrm{lm}(p_k)\mathrm{lm}(\mathrm{poly}(\mathcal{G}_k))$ 且 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_k)\mathcal{G}_k$. 于是有： $v=\mathrm{lm}(p_k),\ \mathcal{H}=\mathcal{G}_k$.

另一方面，若存在 $j\in\{1,2,\cdots,t\}$ 使得 $\mathrm{lm}(p_j\mathrm{poly}(\mathcal{G}_j))\succ\mathrm{lm}(u\mathrm{poly}(\mathcal{F}))$. 不失一般性，设 $\mathrm{lm}(p_j\mathrm{poly}(\mathcal{G}_j))=\max_{\prec}\{\mathrm{lm}(p_i\mathrm{poly}(\mathcal{G}_i)),\ i=1,2,\cdots,t\}$. 根据等式 $u\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$ 可知，必然还存在 $k\in\{1,2,\cdots,t\}$ 且 $k\neq j$, 使得 $\mathrm{lm}(p_j\mathrm{poly}(\mathcal{G}_j))=\mathrm{lm}(p_k\mathrm{poly}(\mathcal{G}_k))$. 于是存在单项式 $w\in T$, 使 $\mathrm{lm}(p_j)\mathcal{G}_j-\mathrm{lm}(p_k)\mathcal{G}_k=w\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k)$ 成立. 由于 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t$, 因此 $\mathrm{s}(\mathcal{G}_j,\mathcal{G}_k)\prec_{es}\mathrm{s}(\mathcal{F},\mathcal{F}')$ 已被处理，从而 s 多项式 $\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k)$ 可被当前的标记多项式集合 B 约化成 0, 即存在 $q_1,q_2,\cdots,q_s\in\mathrm{F}_2[X],\ \mathcal{R}_1,\mathcal{R}_2,\cdots,\mathcal{R}_{s+1}\in B$, 使得： $\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k)=q_1\mathcal{R}_1+q_2\mathcal{R}_2+\cdots+q_s\mathcal{R}_s+\mathcal{R}_{s+1}$ 且 $ES(\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k))=ES(\mathcal{R}_{s+1})$, $\mathrm{index}(\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k))=\mathrm{index}(\mathcal{R}_{s+1}),\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k)\succ_{es}\mathrm{lm}(q_i)\mathcal{R}_i,\ i=1,2,\cdots,s$. 将 $\mathrm{spoly}(\mathcal{G}_j,\mathcal{G}_k)$ 的表达式带回到原等式中，可以得到一个新的等式： $u\mathcal{F}=p_1'\mathcal{G}_1'+p_2'\mathcal{G}_2'+\cdots+p_{t'}'\mathcal{G}_{t'}'$ 依然满足 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i')\mathcal{G}_i',\ i=1,2,\cdots,t'$. 更重要的是，等式右边出现的 $\mathrm{lm}(p_i'\mathrm{poly}(\mathcal{G}_i'))$ 不会高于原来的等式. 重复这个过程，容易证明，在有限步内必可以得到等式：$u\mathcal{F}=\widetilde{p}_1\widetilde{\mathcal{G}}_1+\widetilde{p}_2\widetilde{\mathcal{G}}_2+\cdots+\widetilde{p}_{\widetilde{t}}\widetilde{\mathcal{G}}_{\widetilde{t}}$, 使得不存在 $j\in\{1,2,\cdots,\widetilde{t}\}$ 满足 $\mathrm{lm}(\widetilde{p}_j\mathrm{poly}(\widetilde{\mathcal{G}}_j))\succ\mathrm{lm}(u\mathrm{poly}(\mathcal{F}))$. 结合前一段的讨论，可知存在 $v=\mathrm{lm}(\widetilde{p}_k),\ \mathcal{H}=\widetilde{\mathcal{G}}_k,\ k\in\{1,2,\cdots,\widetilde{t}\}$, 满足 $u\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=v\mathrm{lm}(\mathrm{poly}(\mathcal{H}))$ 且 $u\mathcal{F}\succ_{es}v\mathcal{H}$.

最后，我们来证明 s 多项式 $\mathrm{spoly}(\mathcal{F},\mathcal{F}')$ 可以被 B 约化成 0. 由 s 多项式的定义 $\mathrm{spoly}(\mathcal{F},\mathcal{F}')=u\mathcal{F}-u'\mathcal{F}'$, 再结合等式 $u\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$, 于是可以得到 $\mathrm{spoly}(\mathcal{F},\mathcal{F}')=u\mathcal{F}-u'\mathcal{F}'=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t-u'\mathcal{F}'$. 由于 $\mathcal{F}'\in B$, 因而可以令 $p_{t+1}=-u',\ \mathcal{G}_{t+1}=\mathcal{F}'$, 为了叙述方便设 $\mathcal{F}_1=\mathrm{spoly}(\mathcal{F},\mathcal{F}')$. 于是得到表达式： $\mathcal{F}_1=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t+p_{t+1}\mathcal{G}_{t+1}$. 因为 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t$, 又由 s 对的定义有 $u\mathcal{F}\succ_{es}u'\mathcal{F}'=-p_{t+1}\mathcal{G}_{t+1}$, 易知 $\mathcal{F}_1=u\mathcal{F}-u'\mathcal{F}'\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t+1$. 结合前面的证明，存在 $\mathcal{H}_1\in B$, 以及单项式 $v_1\in T$, 满足：$\mathrm{lm}(\mathrm{poly}(\mathcal{F}_1))=v_1\mathrm{lm}(\mathrm{poly}(\mathcal{H}_1))$ 并且 $\mathcal{F}_1\succ_{es}v_1\mathcal{H}_1$. 令 $\mathcal{F}_2=\mathcal{F}_1-v_1\mathcal{H}_1,\ p_{t+2}=-v_1,\ \mathcal{G}_{t+2}=\mathcal{H}_1$, 则有等式 $\mathcal{F}_2=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_{t+2}\mathcal{G}_{t+2}$, 且易证 $\mathcal{F}_2\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t+2$. 接下来我们重复利用前面的证明寻找 v_j 与 $\mathcal{H}_j$, 并构造相应的 $\mathcal{F}_{j+1}=\mathcal{F}_j-v_j\mathcal{H}_j$. 因为 $\mathrm{lm}(\mathrm{poly}(\mathcal{F}_j))\succ\mathrm{lm}(\mathrm{poly}(\mathcal{F}_{j+1}))$, 这个过程必会在有限步内终止，不妨设在 $\mathcal{F}_l$ 时终止，则必有 $\mathrm{poly}(\mathcal{F}_l)=0$, 否则将仍会找到 v_l 与 $\mathcal{H}_l$ 使操作继续进行下去.

重新审视整个过程就会发现，由 $\mathcal{F}_j$ 生成 $\mathcal{F}_{j+1}$, 实际上就是用 B 对 $\mathcal{F}_j$ 进行了一次约化，因此，上面所作的所有操作可以简单的写为： $\mathcal{F}_1=\mathrm{spoly}(\mathcal{F},\mathcal{F}')\longrightarrow_B^*\mathcal{F}_l$. 由于 $\mathrm{poly}(\mathcal{F}_l)=0$, 所以 s 多项式 $\mathrm{spoly}(\mathcal{F},\mathcal{F}')$ 可以被 B 约化成 0. 证毕.

推论 2.10 已知当前的标记多项式集合 B, 以及当前需要处理的 s 对为 $\mathrm{s}(\mathcal{F}',\mathcal{F})=(u',\mathcal{F}',u,\mathcal{F})$, 其中 $u',u\in T,\ \mathcal{F}',\mathcal{F}\in B$. 若存在 $p_1,p_2,\cdots,p_t\in\mathrm{F}_2[X],\ \mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_t\in B$, 使得： $u\mathcal{F}=p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$ 且 $u\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i,\ i=1,2,\cdots,t$. 则 $\mathrm{spoly}(\mathcal{F}',\mathcal{F})$ 可以被 B 约化成 0.

证　由引理 2.9 可知, 存在 $\mathcal{H}\in B$, 以及单项式 $v\in T$, 满足 $u\mathrm{lm}(\mathrm{poly}(\mathcal{F}))=v\mathrm{lm}(\mathrm{poly}(\mathcal{H}))$ 且 $u\mathcal{F}\succ_{es}v\mathcal{H}$. 则有 $\mathrm{spoly}(\mathcal{F}',\mathcal{F})=u'\mathcal{F}'-u\mathcal{F}=(u'\mathcal{F}'-v\mathcal{H})-(u\mathcal{F}-v\mathcal{H})=w_1\mathrm{spoly}(\mathcal{F}',\mathcal{H})-w_2\mathrm{spoly}(\mathcal{F},\mathcal{H})$, 其中 $w_1=GCD(u',v)$, $w_2=GCD(u,v)$. 因为 $u'\mathcal{F}'\succ_{es}u\mathcal{F}$, $u\mathcal{F}\succ_{es}v\mathcal{H}$, 所以有 $\mathrm{s}(\mathcal{F}',\mathcal{F})\succ_{es}\mathrm{s}(\mathcal{F}',\mathcal{H})$, $\mathrm{s}(\mathcal{F}',\mathcal{F})\succ_{es}\mathrm{s}(\mathcal{F},\mathcal{H})$. 因此 s 对 $\mathrm{s}(\mathcal{F}',\mathcal{H})$ 与 $\mathrm{s}(\mathcal{F},\mathcal{H})$ 都已经被处理过, 其 s 多项式可以被 B 约化成 0, 所以 s 多项式 $\mathrm{spoly}(\mathcal{F}',\mathcal{F})$ 也可以被 B 约化成 0. 证毕.

有了引理 2.9 和推论 2.10, 证明 s 多项式可以被约化成 0 的问题便可以转化为构造标记多项式的 “低阶表示 ” 的问题. 下面我们要做的就是将两个标准的已知条件转化成标记多项式的 “低阶表示”. 为此, 我们还需要另外一个引理.

引理 2.11 若已知标记多项式 $\mathcal{F}=(x^\alpha,f_j,j,f,k)\in B$, 则必存在 $p_1,p_2,\cdots,p_t\in\mathrm{F}_2[X]$, $\mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_t\in B$, 使得: $\mathcal{F}=x^\alpha\mathcal{F}_j+p_1\mathcal{G}_1+p_2\mathcal{G}_2+\cdots+p_t\mathcal{G}_t$ 且 $\mathcal{F}\succ_{es}\mathrm{lm}(p_i)\mathcal{G}_i, i=1,2,\cdots,t$, 其中 $\mathcal{F}_i$ 是初始多项式 f_i 的标记多项式.

证　对 $\mathcal{F}$ 加入集合 B 的顺序进行归纳法. 首先, 初始多项式对应的标记多项式 $\mathcal{F}_i$, $i=1,2,\cdots,m$ 都满足上面的条件. 其次, 假设在 $\mathcal{F}$ 之前加入 B 的标记多项式都满足条件. 由算法可知, 标记多项式 $\mathcal{F}$ 必由某个 s 多项式约化而来, 设这个 s 多项式为 $\mathrm{spoly}(\mathcal{G},\mathcal{G}')=u\mathcal{G}-u'\mathcal{G}'$. 由约化的过程可知, 存在 $q_1,q_2,\cdots,q_s\in\mathrm{F}_2[X]$, $\mathcal{H}_1,\mathcal{H}_2,\cdots,\mathcal{H}_s\in B$, 使得 $\mathcal{F}=u\mathcal{G}-u'\mathcal{G}'-q_1\mathcal{H}_1-q_2\mathcal{H}_2-\cdots-q_s\mathcal{H}_s$, 并且 $\mathcal{F}\succ_{es}u'\mathcal{G}'$, $\mathcal{F}\succ_{es}\mathrm{lm}(q_i)\mathcal{H}_i, i=1,2,\cdots,s$. 设 $\mathcal{G}=(x^\beta,f_j,j,g,k')$, 因为 $\mathcal{G}$ 比 $\mathcal{F}$ 早加入 B, 由归纳假设可知, 则存在 $p'_1,p'_2,\cdots,p'_{t'}\in\mathrm{F}_2[X]$, $\mathcal{G}'_1,\mathcal{G}'_2,\cdots,\mathcal{G}'_{t'}\in B$, 使得 $\mathcal{G}=x^\beta\mathcal{F}_j+p'_1\mathcal{G}'_1+\cdots+p'_{t'}\mathcal{G}'_{t'}$. 将这个等式带入前面的等式, 易证 $\mathcal{F}$ 也满足条件. 证毕.

引理 2.11 告诉我们, 每个标记多项式都有一个 “表示 ”, 但注意到由于 $x^\alpha\mathcal{F}_j$ 的存在, 这个 “表示 ” 并不是一个 “低阶表示 ”. 因此, 为了应用引理 2.9 和推论 2.10, 我们还需要一些额外的信息用以构造严格的 “低阶表示 ”. 下面我们来证明两条标准.

定理 2.12 合冲 (syzygy) 标准是正确的.

证　因为 $\mathrm{lm}(f_j)|S(u_k\mathcal{G}_k)$, 则存在 $u\in T$, 使得 $S(u_k\mathcal{G}_k)=u\mathrm{lm}(f_i)$. 设 $\mathcal{G}_k=(x^\alpha,f_l,l,g,k')$, 由引理 2.11 可知, 存在 $p_1,p_2,\cdots,p_t\in\mathrm{F}_2[X]$, $\mathcal{H}_1,\mathcal{H}_2,\cdots,\mathcal{H}_t\in B$, 使得 $\mathcal{G}_k=x^\alpha\mathcal{F}_l+p_1\mathcal{H}_1+p_2\mathcal{H}_2+\cdots+p_t\mathcal{H}_t$ 且 $\mathcal{G}_k\succ_{es}\mathrm{lm}(p_i)\mathcal{H}_i,\ i=1,2,\cdots,t$, 其中 $\mathcal{F}_l$ 是初始多项式 f_l 的标记多项式.

当 $j=\mathrm{index}(u_k\mathcal{G}_k)=l$ 时, 在多项式环 $\mathrm{F}_2[X]$ 中, 有等式 $f_l^2=f_l+q_1(x_1^2+x_1)+q_2(x_2^2+x_2)+\cdots+q_n(x_n^2+x_n)$ 成立, 等价的可以写为 $\mathrm{lm}(f_l)f_l=(1-(f_l-\mathrm{lm}(f_l)))f_l+q_1(x_1^2+x_1)+q_2(x_2^2+x_2)+\cdots+q_n(x_n^2+x_n)$, 其中 $q_i\in\mathrm{F}_2[X]$, $i=1,2,\cdots,n$. 由于在我们的算法里域多项式 $x_i^2+x_i$ 也是初始多项式, 因而上面的等式关系写成标记多项式的形式为: $\mathrm{lm}(f_l)\mathcal{F}_l=q_0\mathcal{F}_l+q_1\mathcal{R}_1+q_2\mathcal{R}_2+\cdots+q_n\mathcal{R}_n$, 其中 $q_0=(1-(f_l-\mathrm{lm}(f_l)))$, $\mathcal{R}_i$ 为域多项式所对应的标记多项式. 由于我们假设域多项式的序大于非域多项式的序, 因而不难证明: $\mathrm{lm}(f_l)\mathcal{F}_l\succ_{es}\mathrm{lm}(q_i)\mathcal{R}_i$, $i=1,2,\cdots,n$. 由于 $\mathrm{lm}(f_l)\succ\mathrm{lm}(q_0)$, 于是有 $\mathrm{lm}(f_l)\mathcal{F}_l\succ_{es}\mathrm{lm}(p_0)\mathcal{F}_l$. 结合上一段中的等式, 有 $u_k\mathcal{G}_k=u_kx^\alpha\mathcal{F}_l+u_kp_1\mathcal{H}_1+u_kp_2\mathcal{H}_2+\cdots+u_kp_t\mathcal{H}_t=u\mathrm{lm}(f_l)\mathcal{F}_l+u_kp_1\mathcal{H}_1+u_kp_2\mathcal{H}_2+\cdots+u_kp_t\mathcal{H}_t=uq_0\mathcal{F}_l+uq_1\mathcal{R}_1+uq_2\mathcal{R}_2+\cdots+uq_n\mathcal{R}_n+u_kp_1\mathcal{H}_1+u_kp_2\mathcal{H}_2+\cdots+u_kp_t\mathcal{H}_t$. 且有 $u_k\mathcal{G}_k\succ_{es}u\mathrm{lm}(q_0)\mathcal{F}_l$, $u_k\mathcal{G}_k\succ_{es}u\mathrm{lm}(q_i)\mathcal{R}_i$, $i=1,2,\cdots,n$, $u_k\mathcal{G}_k\succ_{es}u_k\mathrm{lm}(p_i)\mathcal{H}_i$, $i=1,2,\cdots,t$. 于是, 由引理 2.9 和推论 2.10 可知, s 多项式 $\mathrm{spoly}(\mathcal{G}_1,\mathcal{G}_2)$ 可以被 B 约化成 0.

当 $\mathrm{index}(u_k\mathcal{G}_k)<j\le m$ 时, 显然有等式 $f_jf_l=f_lf_j$ 成立, 等价的有 $\mathrm{lm}(f_j)f_l=f_lf_j-(f_j-\mathrm{lm}(f_j))f_l$. 写成标记多项式的形式为: $\mathrm{lm}(f_j)\mathcal{F}_l=q_1\mathcal{F}_j+q_2\mathcal{F}_l$, 其中 $q_1=f_l$, $q_2=$

$-(f_j - \mathrm{lm}(f_j))$, 且有 $\mathrm{lm}(f_j)\mathcal{F}_l \succ_{es} \mathrm{lm}(q_1)\mathcal{F}_j$ 和 $\mathrm{lm}(f_j)\mathcal{F}_l \succ_{es} \mathrm{lm}(q_2)\mathcal{F}_l$ 成立. 结合证明部分第一段中给出的等式, 再利用上一段中所使用的方法推导 $u_k\mathcal{G}_k$ 的表达式, 不难证明 s 多项式 $\mathrm{spoly}(\mathcal{G}_1, \mathcal{G}_2)$ 可以被 B 约化成 0. 证毕.

定理 2.13 重写 (rewritten) 标准是正确的.

证 设 $\mathcal{G}_k = (x^\alpha, f_l, l, g, k')$, 由引理 2.11 可知, 存在 $p_1, p_2, \cdots, p_t \in \mathrm{F}_2[X]$, $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_t \in B$, 使得 $\mathcal{G}_k = x^\alpha\mathcal{F}_l + p_1\mathcal{H}_1 + p_2\mathcal{H}_2 + \cdots + p_t\mathcal{H}_t$ 且 $\mathcal{G}_k \succ_{es} \mathrm{lm}(p_i)\mathcal{H}_i$, $i = 1, 2, \cdots, t$, 其中 $\mathcal{F}_l$ 是初始多项式 f_l 的标记多项式. 同样, 设 $\mathcal{F} = (x^\beta, f_l, l, f, k'')$, 存在 $q_1, q_2, \cdots, q_s \in \mathrm{F}_2[X]$, $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_s \in B$, 使得 $\mathcal{F} = x^\beta\mathcal{F}_l + q_1\mathcal{R}_1 + q_2\mathcal{R}_2 + \cdots + q_s\mathcal{R}_s$ 且 $\mathcal{F} \succ_{es} \mathrm{lm}(q_i)\mathcal{R}_i$, $i = 1, 2, \cdots, s$. 由已知可得 $u_kx^\alpha = vx^\beta$, 于是有等式: $u_k\mathcal{G}_k = u_kx^\alpha\mathcal{F}_l + u_kp_1\mathcal{H}_1 + u_kp_2\mathcal{H}_2 + \cdots + u_kp_t\mathcal{H}_t = vx^\beta\mathcal{F}_l + u_kp_1\mathcal{H}_1 + u_kp_2\mathcal{H}_2 + \cdots + u_kp_t\mathcal{H}_t = v\mathcal{F} - (vq_1\mathcal{R}_1 + vq_2\mathcal{R}_2 + \cdots + vq_s\mathcal{R}_s) + u_kp_1\mathcal{H}_1 + u_kp_2\mathcal{H}_2 + \cdots + u_kp_t\mathcal{H}_t$. 由于 $\mathrm{num}(\mathcal{G}_k) < \mathrm{num}(\mathcal{F})$, 故而 $u_k\mathcal{G}_k \succ_{es} v\mathcal{F}$, 加上 $v\mathcal{F} \succ_{es} vq_i\mathcal{R}_i$, $i = 1, 2, \cdots, s$ 和 $u_k\mathcal{G}_k \succ_{es} u_kp_i\mathcal{H}_i$, $i = 1, 2, \cdots, t$. 再结合引理 2.9 和推论 2.10 可知, s 多项式 $\mathrm{spoly}(\mathcal{G}_1, \mathcal{G}_2)$ 可以被 B 约化成 0. 证毕.

3 多项式的 zdd 表达形式

zdd 是一种特殊的二叉树, 其前身是用来表示布尔函数 bdd. 近些年来, 随着研究的深入, 学者们发现用 zdd 来表示多项式效果很好, 同时还产生了一些基于 zdd 数据结构的算法, 其中最著名的是 Brickenstein 给出的 PolyBoRi 程序包, 以及高小山的布尔环上特征列算法. zdd 数据结构的最大特点在于, 可以将布尔环中的多项式以一种共享的形式进行存储, 从而大大减少算法对于内存空间的使用.

用 zdd 表示多项式非常直观易懂. 我们称, 从树的根节点出发, 到叶节点 1 的路线为非平凡的路线; 类似的, 从根节点出发, 到叶节点 0 的路线称为平凡的路线. 非平凡路线上经过的所有从右支出发的节点可以表示一个单项式 (从左支出发的节点表示该节点在单项式中不出现), 而整棵树所表示的多项式就是所有非平凡路线表示的单项式之和. 比如, 图 1 中的 (1) 表示的多项式为 $f = x_3x_1 + x_2$.

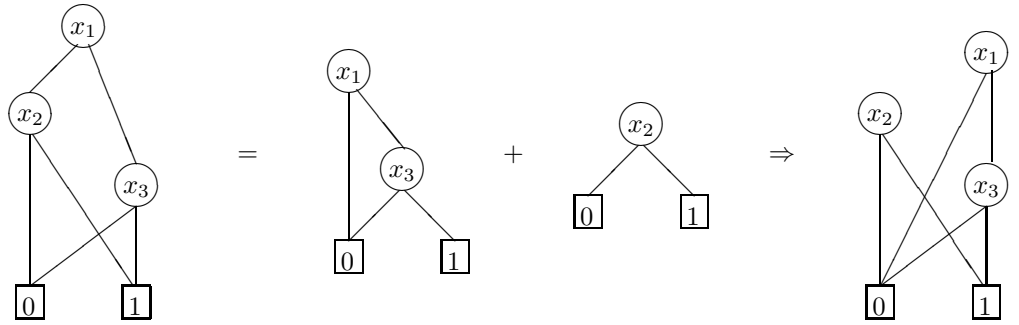

图 1 zdd 数据结构

此前, Brickenstein 给出的 PolyBoRi 程序包也包含利用 zdd 数据结构计算布尔环上 Gröbner 基的算法. 但从其文章来看, PolyBoRi 中的算法更适合计算字典序下的 Gröbner

基，这是因为 zdd 所表示的多项式在字典序下的首项 (或逆字典序) 可以通过遍历一条路线获得，而用 Brickenstein 给出的算法计算全次字典序 (或全次逆字典序) 的首项则需要至少遍历半棵树. 同时， Brickenstein 在文中给出的实验数据也全部都是字典序下的计算结果.

众所周知，相比字典序和其它多项式序，计算全次逆字典序下的 Gröbner 基理论复杂度最低. 由于多项式首项在算法运行中需要频繁计算，因而若想采用 zdd 数据结构计算全次逆字典序下的 Gröbner 基，就必须先解决首项计算的问题. 为此，我们在算法实现中采用了多项式分次表示的形式，即多项式的每个齐次分量用一棵树表示. 例如，前面提到的多项式 $f = x_3x_1 + x_2$ 有两个齐次分量，因而可以用两棵树表示，如图 1 中的 (2).

多项式的分次表示形式所带来的最大好处，就是使多项式在全次逆字典序 (或全次字典序) 下的首项变得非常容易计算，即只需要遍历最高次齐次分量中的一条路线便可得到. 分次表示所带来的另一个好处是在构造矩阵时，我们可以根据次数构造分次矩阵，使得原来的大矩阵可以拆分成有限个小矩阵来处理，一定程度上降低了矩阵处理的复杂度. 但是由于采用分次表示的形式，一个多项式经常需要几棵树来表示，这样便会使总的树的个数增加. 为了解决这一问题，我们采用了共享节点的技术，即共享不同树中的相同节点，如图 1 中的 (3) 就是共享了叶节点 0 和 1. 采用共享的技术，树个数的增加并不会占用太多内存，相反还能提高树间运算的效率. 这种共享节点的技术，已在我们所调用的程序包 CUDD[15] 中实现.

注 在算法实现中， zdd 只用来存储多项式，而多项式的约化过程仍然需要转化为矩阵进行处理，这是因为相比 zdd 间的运算，特别是加法运算，矩阵的处理效率要高出很多.

4 分支 Gröbner 基算法

我们的分支 Gröbner 基想法主要是从高小山的布尔环上特征列算法中得到的启发. 因为在多项式环 $\mathrm{F}_2[X]$ 中，由于变量 x_i 只能从 0 和 1 中取值，因而任意多项式 $f \in \mathrm{F}_2[X]$ 其取值也只有 0 和 1 两种. 在算法中，分别讨论多项式的两种取值就构成了分支算法. 相比无分支算法，分支算法中每个分支会有更多的多项式信息，因而有理由相信在分支中可以更容易的计算出结果，这也正是我们提出分支 Gröbner 基算法的动机. 同时，分支算法也更适于运用并行处理的技术，可以更有效的利用计算机资源. 另一方面，由于分支算法中需要保存各个不同分支的多项式及相关信息，将占用更多的内存资源，因而分支算法的内存使用是一个很关键的问题. 幸运的是，我们采用了 zdd 数据结构，可以利用很少的内存空间保存大量的多项式信息，这也使分支算法的实现变成可能.

分支算法的目标是希望所有分支系统的总复杂度低于原系统的复杂度，因而分支算法中最核心的一点，就是确保每个分支系统的复杂度都要低于原系统的复杂度. 对于 F4 和 F5 算法，其复杂度基本上可以由参数 D 完全确定，其中 D 是算法中所构造的最大矩阵的次数. 由于我们的算法采用了 F5 算法的基本结构，因而在构造分支时就要努力控制分支中的矩阵次数，使其不能高于原系统，这也是我们选择分支策略的一个基本原则. 选择分支策略的另一个原则是要保证可能出现的分支个数在一个合理的范围内. 如果分支个数过多，则所有分支的总复杂度将会超过原系统，这样反而得不偿失. 有关分支复杂度与分支个数关系的讨论将会在另一篇文章中进行详细阐述.

通过大量的实验发现，可行的分支策略虽然有很多，但很难找到一种在所有例子中效果都很好的分支策略. 因而应针对不同系统的内部结构选择其适合的分支策略. 这里需要提及

的是, 在我们的算法实现中, 构造分支的过程是通过加入新的初始多项式来实现的. 所以, 一种分支策略可以简单的理解为是一种选择多项式的策略. 下面介绍 3 种实验效果较好的分支策略, 其中 D' 是事先设定好的一个次数

1) 若当前需要处理的多项式次数高于 D' 时, 取该多项式首项中的一个变量 x_i, 利用 $x_i = 0$ 与 $x_i - 1 = 0$ 构造分支;

2) 若当前需要处理的多项式次数高于 D' 时, 设该多项式最高次的齐次分量为 h_i, 利用 $h_i = 0$ 与 $h_i - 1 = 0$ 构造分支;

3) 若当前需要处理的多项式次数高于 D' 时, 检索现有的多项式集合, 设 g 是 zdd 数据结构中引用次数最高的子多项式, 则利用 $g = 0$ 与 $g - 1 = 0$ 构造分支. 其中, 引用次数是指在 zdd 中节点的共享次数, 引用次数越高, 说明包含该子多项式的多项式越多.

这 3 种分支策略都可以很容易的在算法中实现, 并且获得了不错的效果. 注意到这 3 种策略都非常注重对分支次数的控制, 可以确保每个分支的复杂度不会过高. 因而当分支的个数控制在一个合理范围内时, 分支算法的效果非常好, 在第 6 节中我们会看到分支算法在求解随机生成的多项式系统以及流密码系统方面的应用.

5 算法实现

算法的基本流程如第 2 节中所述, 但实现过程中仍有很多细节问题需要处理.

首先, 鉴于矩阵技术在 F4, F5 算法中的成功, 我们算法的约化过程也采用了矩阵的技术. 由于合冲与重写准则对于约化顺序有很强的要求, 同时这也是 F5 算法的一个特点, 因而在矩阵处理中, 我们采用了结构高斯消去法 [16], 以保证约化有序进行.

然而, 完全按照扩展签名从低到高的顺序处理 s 对的效率并不理想, 这是因为这样做实际上等价于先将原多项式系统齐次化, 再计算其 Gröbner 基, 从另一个角度看, 这种 s 对的处理顺序也等价于 Mora 等人提出的一种按照拟齐次次数进行处理的顺序 [17]. 究其根本原因, 是由标记多项式的标签定义与其对应多项式之间的不协调所致. 标记多项式的标签记录了该多项式与初始多项式之间的联系, 这种联系会随着算法运行而逐渐变得复杂, 而另一方面, 标记多项式所对应的多项式本身却往往会随着算法运行而被约化的更为简单. 也就是说, 按照扩展标签的顺序处理 s 对经常会先处理较为复杂的多项式, 而后才去处理简单的多项式. 在算法实现过程中我们发现, 有两种方法可以解决这一问题.

1) 在处理 s 对时, 先选择多项式次数最低的 s 对, 然后再按照扩展签名的顺序进行处理.

2) 当标记多项式的签名较为复杂, 但其多项式本身十分简单时, 将该标记多项式改为初始标记多项式. 即将其签名清 0, 其序定义为大于现有非域多项式的序, 但小于域多项式的序. 可以理解为在算法中加入一个新的多项式.

在实现时方法 1 的效果很好, 但由于先按照次数选取 s 对, 本文中所给的证明将不再适用. 虽然在实验中, 该方法输出的结果完全正确, 但目前尚无该算法的完整理论证明. 相比于方法 1, 由于方法 2 并没有改变 s 对的处理顺序, 因而其正确性完全可以被理论证明, 并且实现效果也非常不错. 更重要的是方法 2 与方法 1 并不矛盾, 可以同时使用. 实际上, 方法 2 是 F4 与 F5 算法在某种意义上的中和. 将签名清 0 意味着切断该多项式与其他多项式之间的联系, 若将所有生成多项式的签名都清 0, 则算法几乎就变成了 F4 算法.

分支 Gröbner 基算法也主要借助方法 2 来实现，即在算法运行过程中适时的加入一些初始多项式以构造分支.

6 实验数据

在这一节中，我们主要比较分支 Grönber 基算法与 Magma 中 Steel 实现的 F4 算法在流密码以及随机生成的多项式系统求解方面的应用. 在实验中，我们采用分支策略 1. 实验所用电脑为： Linux 操作系统， 3.00GHz CPU, 16G 内存. 实验数据的单位为：秒.

流密码是密码学中的一类十分重要的密码系统，这里我们主要考虑基于线性反馈移位寄存器 (linear feedback shift register, LFSR) 的流密码. 下面的 filter 函数取自 [18].

- CanFil 2, $x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_4 + x_2x_5 + x_3 + x_4 + x_5$
- CanFil 3, $x_2x_3x_4x_5 + x_1x_2x_3 + x_2x_4 + x_3x_5 + x_4 + x_5$
- CanFil 8, $x_1x_2x_3 + x_2x_3x_6 + x_1x_2 + x_3x_4 + x_5x_6 + x_4 + x_5$
- CanFil 9, $x_2x_4x_5x_7 + x_2x_5x_6x_7 + x_3x_4x_6x_7 + x_1x_2x_4x_7 + x_1x_3x_4x_7 + x_1x_3x_6x_7 + x_1x_4x_5x_7 + x_1x_2x_5x_7 + x_1x_2x_6x_7 + x_1x_4x_6x_7 + x_3x_4x_5x_7 + x_2x_4x_6x_7 + x_3x_5x_6x_7 + x_1x_3x_5x_7 + x_1x_2x_3x_7 + x_3x_4x_5 + x_3x_4x_7 + x_3x_6x_7 + x_5x_6x_7 + x_2x_6x_7 + x_1x_4x_6 + x_1x_5x_7 + x_2x_4x_5 + x_2x_3x_7 + x_1x_2x_7 + x_1x_4x_5 + x_6x_7 + x_4x_6 + x_4x_7 + x_5x_7 + x_2x_5 + x_3x_4 + x_3x_5 + x_1x_4 + x_2x_7 + x_6 + x_5 + x_2 + x_1$
- CanFil 10, $x_1x_2x_3 + x_2x_3x_4 + x_2x_3x_5 + x_6x_7 + x_3 + x_2 + x_1$

表 1 中给出了相关数据比较，其中 n 表示变量个数， MGB 是 Magma 中的 F4 算法，BGB 是我们的分支函数. Deg 是算法 (分支算法) 所达到的最高次数， Num 为分支的总个数，表中 – 表示 Magma 因内存溢出而终止.

表 1

Filters	n	81			100			128		
		Time	Deg	Num	Time	Deg	Num	Time	Deg	Num
	MGB	18.730	7		32.930	7		–	–	
CanFil2	BGB	0.027	3	9	0.172	3	66	0.357	3	40
	MGB	–	–		1.360	7		–	–	
CanFil3	BGB	0.085	3	11	0.150	3	19	1.210	3	17
	MGB	49.460	7		12.590	7		–	–	
CanFil8	BGB	0.046	3	27	0.170	3	190	0.371	3	140
	MGB	–	–		–	–		–	–	
CanFil9	BGB	0.418	4	40	1.230	4	217	20.901	4	77
	MGB	331.880	7		–	–		–	–	
CanFil10	BGB	0.131	3	99	0.612	3	501	1.296	3	340

表 2 中给出了两个算法求解随机生成的二次多项式系统的比较，其中 n 表示变量个数，并取多项式个数 $m = n$, 表中的数据是计算三组随机生成数据的平均值.

表 2

n	18		20		22		24		26	
	Time	Deg	Time	Deg	Time	Deg	Time	Deg	Time	Deg
MGB	3.890	6	14.220	6	82.790	6	–	–	–	–
BGB	0.791	3	2.992	3	13.235	3	47.862	3	149.121	3
Num	2048		8160		29046		65400		262018	

通过两个表的比较我们可以发现，在这两种例子中，分支 Gröbner 基算法的表现要明显好于 Magma 中的 F4 算法．主要原因在于，这些例子中无分支的 F4 算法所要达到的最高次数往往较高，因而复杂度较大；而分支算法一直将分支的次数控制在较低的水平，并且在这些例子中分支个数不多，因而总体复杂度大大低于无分支的 F4 算法，这也使得我们的分支算法表现的更好．

7 结束语

在本文中，我们主要介绍了分支 Gröbner 基算法的基本理论，数据结构，以及分支策略．实验数据表明，分支 Gröbner 基算法在很多例子中的表现要明显好于 Magma 中的 F4 算法，因而具有一定的实用价值．但我们应该注意到，分支算法的总效率很大程度上取决于分支策略的选取，优秀的分支策略可以在降低分支复杂度的同时合理的控制分支个数，因而合理的分支策略将是我们日后研究的一个重要内容．另一方面，分支复杂度与分支个数的关系也是我们将要研究的主要课题之一．这些内容都会在将来的文章中进行详细讨论．

参 考 文 献

[1] Wu W T. Basic principles of mechanical theorem-proving in elementary geometries. *J. Automated Reasoning*, 1986, **2**: 221–252.

[2] Wu W T. On zeros of algebraic equations - an application of ritt principle. *Chinese Science Bulletin*, 1986, **31**: 1–5.

[3] Gao X S, Chai F J, Yuan C M. A characteristic set method for equation solving in F2 and applications in cryptanalysis of stream ciphers. *Journal of Systems Science and Complexity*, 2008, **21**(2): 191–208.

[4] Buchberger B. Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal. Ph.D thesis, University of Innsbruck, Innsbruck, 1965.

[5] Buchberger B. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. Reidel Publishing Co., New York, 1985.

[6] Buchberger B. A criterion for detecting unnecessary reductions in the construction of Gröbner basis. European Conference on Computer Algebra, Austria, 1979.

[7] Lazard D. Gaussian elimination and resolution of systems of algebraic equations. European Conference on Computer Algebra, London, 1983.

[8] Gebauer R and Moller H M. Buchberger's algorithm and staggered linear bases. Proceedings of the Symposium on Symbolic and Algebraic Computation, Canada, 1986.

[9] Mora T, Möller H M and Traverso C. Gröbner bases computation using syzygies. Proceedings of the Internation Symposium on Symbolic and Algebraic Computing, USA, 1992.

[10] Faugère J C. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 1999, **139**(1): 61–88.

[11] Faugère J C. A new efficient algorithm for computing Grönber bases without reduction to zero (F5). Proceedings of the Internation Symposium on Symbolic and Algebraic Computing, New York, 2002.

[12] Steel A. Allan steel's Gröbner basis timings page. http://magma.maths.usyd.edu.au/users/allan/gb/.

[13] Minto S. Zero-sppressed BDDs for set manipulation in combinatorial problems. Proc. ACM/IEEE Design Automation, United State, 1993.
[14] Brickenstein M and Dreyer A. PolyBoRi: A framework for Gröbner basis computations with boolean polynomials. Proceedings of Molecular Evolutionary Genetics Analysis, Austria, 2007.
[15] Somenzi F. CUDD: CU decision diagram package. University of Colorado at Boulder, http://vlsi.colorado.edu/ fabio/CUDD/, 2005.
[16] Fatinma K A S and Beirut. A new sparse gaussian elimination algorithm and the niederreiter linear system for trinomials over F_2. *Computing*, 2006, **77** : 179–203.
[17] Giovini A, Mora T, Niesi G, Robbiano L and Traverso C. One sugar cube, please, or selection strategies in the buchberger algorithm. Proceedings of the International Symposium on Symbolic and Algebraic Computation, Germany, 1991.
[18] Faugère J C and Ars G. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Institut National de Recherche en Informatique et en Automatique, Lorraine, 2003.

BRANCH GRÖBNER BASES ALGORITHM OVER BOOLEAN RING

SUN Yao WANG Dingkang

(*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190)

Abstract It is well known that Gröbner bases have extensive applications in many fields. In the recent years, many improvements have been made for the Gröbner algorithm, the most famous of which are the F4 and F5 algorithm introduced by Faugère. Although both of the two algorithms have excellent efficiency, they need enormous memories during the computation. So we will present a new branch Gröbner bases algorithm based on the zdd data structure over the boolean ring. This new algorithm not only lowers the usage of memories but also constrains the matrix generated in the computation within a reasonable size. In this paper, we will detail the theory and the proof of this basic algorithm and introduce the zdd data strucure and the branch strategy as well. For many cases, its implementation in Linux is superior to the F4 algorithm implemented by Steel in Magma.

Key words Branch Gröbner bases, boolean ring, zdd data structure.