"One sugar cube, please"
OR
Selection strategies in Buchberger algorithm

Alessandro Giovini, Teo Mora, Gianfranco Niesi
Lorenzo Robbiano, Carlo Traverso

Abstract

In this paper we describe some experimental findings on selection strategies for Gröbner basis computation with the Buchberger algorithm.

In particular, the results suggest that the "sugar flavor" of the "normal selection", implemented first in CoCoA, then in AlP*I*, and now in SCRATCHPAD-II, is the best choice for a selection strategy. It has to be combined with the "straightforward" simplification strategy and with a special form of the Gebauer-Möller criteria to obtain the best results.

The idea of the "sugar flavor" is the following: the Buchberger algorithm for homogeneous ideals, with degree-compatible term ordering and normal selection strategy, usually works fine. Homogenizing the basis of the ideal is good for the strategy, but bad for the basis to be computed. The sugar flavor computes, for every polynomial in the course of the algorithm, "the degree that it would have if computed with the homogeneous algorithm", and uses this phantom degree (the *sugar*) only for the selection strategy.

We have tested several examples with different selection strategies, and the sugar flavor has proved to be always the best choice or very near to it. The comparison between the different variants of the sugar flavor has been made, but the results are up to now inconclusive.

We include a complete deterministic description of the Buchberger algorithm as it was used in our experiments.

1 The problem

In the Buchberger algorithm, one has a certain freedom of choice in several points. The exercise of this freedom has big influence on the algorithm, not on the final output, but on the complexity of the algorithm: number of pairs to process, growth of the coefficients, in general time and space complexity[1]

One of the main points is the *selection strategy*, i. e. the procedure to choose one of the critical pairs for the process of reduction.

Buchberger, in [7], has defined the *normal selection strategy*, that chooses a pair (g_i, g_j) if the least common multiple τ_{ij} of the leading terms is minimal in the current term-ordering.

This does not completely define the strategy, since ties may occur. In this case, our tie-breaking algorithm is the following: assuming $i < j$, we give priority to the pair with least j, no further tie being possible if you apply the criteria of [12]. (There is a motivation to choose in this way, but things are not yet sufficiently clear to explain it. The practice shows that this is a good choice.)

This selection strategy works fine if the term-ordering is degree-compatible, but is quite bad for the lexicographic term-ordering. We try to explain why.

Consider a ring $A = k[x_1, \dots, x_n]$, and an ideal $I = (f_1, \dots, f_m)$. Perform the Buchberger algorithm with pure lexicographic term-ordering and normal selection strategy; this runs as follows.

As soon as the algorithm produces two polynomials without x_1, the algorithm runs on these elements, disregarding the others, up to a complete determination of the Gröbner basis of the ideal generated by these two polynomials. This is not bad, since this ideal is a complete intersection, whose complexity is intrinsically only simply exponential. But as soon as another polynomial comes in, the others remain still, and we compute a Gröbner basis of an ideal of codimension 2 defined by 3 polynomials, that can be horrible, and in the worst case intrinsically doubly exponential in the dimension of the ideal (i. e. $n-2$). Hence this subproblem is much harder than the full problem (intrinsically doubly exponential in the dimension of the original ideal). The normal selection strategy is hence here a *be divided and perish* strategy.

One of the standard remedia has been the following: always homogenize the generators of the ideal, and perform only Buchberger algorithms for homogeneous ideals, and in increasing degrees. It is experimentally known that in this setting the Buchberger algorithm is less sensible to strategies. At the end, dehomogenize and completely interreduct to obtain a reduced Gröbner basis of the original ideal. Some people have argued that this is the best thing to do, so for example *Macaulay* [2] considers only homogeneous ideals.

This strategy can be bad, since the Gröbner basis of the ideal generated by the homogenized polynomials (that is *not* a Gröbner basis of the homogenized ideal) can be much larger than the Gröbner basis of the original ideal, and can have components at infinity of large dimension (consider the extreme case $1 \in I$).

Other possibilities that were studied are the following:

1. First compute the Gröbner basis with respect to another term-ordering, (usually the DegRevLex), then

[1] an assertion of the same type is true also for political systems

0

deduce the Lex Gröbner basis, either by linear algebra ([11], in the 0-dimensional case), or with Buchberger algorithm; since the components at infinity have been already suppressed, if the problem is geometrically good then the algorithm can be smoother.

2. Compare the τ_{ij} with respect to a term-ordering different than the current term-ordering: usually, compare the degrees, then compare with the current term-ordering

3. Simulate the homogeneous algorithm in what concerns the selection strategy. This is the "Sugar Flavor" of CoCoA, and will be explained with all the details in a following section, together with some variants.

The simple form of the sugar flavor is experimentally the winning strategy. This explains the first line of the title.

2 The Sugar Flavor

We define a "phantom" homogenization of all the polynomials in the Buchberger algorithm, defining for each polynomial f its *Sugar* S_f, in the following way:

1. for the initial f_i, $S_{f_i} = \deg f_i$, its degree (not the degree of the leading term, but the true total degree)
2. if f is a polynomial and $t = X^\alpha$ a term, then $S_{tf} = \deg t + S_f$
3. if $f = g + h$, then $S_f = \max(S_g, S_h)$.

To every polynomial "with sugar" we can associate an homogeneous polynomial of degree equal to the sugar, homogenizing with an additional variable and multiplying by a suitable power of the same variable, and every operation in the Buchberger algorithm can be performed on the homogeneous polynomials multiplying whenever needed by the powers needed to mantain homogenity. Hence the sugar can be seen as a phantom degree.

We define a selection strategy "with sugar" comparing critical pairs giving priority to lower sugar (the sugar of a pair being the sugar of the S-polynomial), and breaking ties with another selection strategy that involves the τ_{ij}, typically the normal selection strategy. Usually, we refer to the "normal with sugar" strategy simply as "sugar" strategy

The *Double sugar* algorithm is the Buchberger algorithm with sugar selection strategy but with another feature (that substantially concerns the simplification strategy). To explain it, we have to spend a few words on the simplification strategies.

In the Buchberger algorithm, one considers a critical pair, computes the S-polynomial, then rewrites it using the elements of the basis found up to now. Different rewritings are usually possible, and one has to choose one of them.

Some points are not questioned: as soon as the leading term can be rewritten, one has to preliminarly rewrite it (any other rewriting can substantally be at most a loss of time).

Other points have been more or less experimentally settled and universally recognized: it is always better to rewrite also the other terms in the polynomial, the alternative (only rewrite leading terms, and add to the basis polynomials not completely reduced, is worse for combinatorial complexity and coefficient growth).

Other points have now a large consensus between experts: it is better not to interreduce old polynomials with newly added polynomials; it is better, if a term can be rewritten with two or more different elements of the basis, to use the "older" one, i. e. appending new element to the existing basis, always use the first suitable element found. The usefulness of this point was not recognized at once since this is the easiest way to proceed, and the full importance of this criterion can often be masked by an improper selection strategy.

The "double sugar" strategy tries to loosen one of these criteria: it consists on simplifying the intermediate monomials only if the simplification does not increase the sugar of the polynomial: the rationale is, whenever a polynomial of a given leading term is found, try to keep its sugar as low as possible. This strategy (really it should be called "saccharine") was found to be often equivalent to the original strategy, and sometimes worse.

In practice, during the Buchberger algorithm, it is easy to give a tag to every polynomial, and to every critical pair, containing its sugar. In a long computation, the time and space lost in the sugar computations are clearly irrelevant. Hence the comparations of selection strategies with and without sugar can be made with the same implementation if it allows the choice of different types of selection strategies.

The sugar strategy may interphere with the useless pair elimination, if only already considered pairs are used to eliminate other pairs. If this is considered, we have the "fussy" sugar strategy, otherwise we have the "sloppy" sugar strategy. The details are explained in the appendix.

3 Experiments

The sugar strategy has been implemented originally in CoCoA, and has proved to be a good strategy. It has then been implemented in AlP*I*. The experiments that we report have been performed with AlP*I*, since CoCoA is less suitable to experiments, not having "true" integer arithmetic or complete diagnostic. To make results verifiable, in the appendix we give the complete description of all the choices of the algorithm that were used.

The examples used were some of the examples already tested in [21] (of which some were taken from [4]), and a few more. Some of the examples were less relevant (especially with the Deg-Rev-Lex ordering), but the overall superiority of the sugar options was clear. In no case the sugar strategy was substantially worse than the best choice, and this is of course very important.

An important remark is that the strategy of homogenizing the basis is never very good. This is an indication that the rest of our settings is really good: homogeneous ideals are very little sensible to strategies, provided that one works in increasing degrees, and experimentally the homogenization strategy is better than not very well-tuned strategies.

In the following tables we summarize some of the the results, discarding the less significant ones; the data reported in the columns are:

the different options tested (S sugar, 2 double-sugar, H homogenized polynomials, D degree + normal selection strategy, N normal selection) and the term-ordering (R Deg-Rev-Lex, L Lex term-ordering)

TI execution times (on a 16MHz 80386 DOS computer, running the MuLISP version 1.5a of AlPi). Execution times are slightly variable in different runs of the algorithm, we did not attempt to run the algorithms different times to find the optimal, or average, time.

CL the length of the maximum integer computed in the algorithm, the length being the number of 16-bit words needed to store its absolute value

CP the number of critical pairs considered

UP the number of "useful" critical pairs (added to the length of the original basis gives the length of the redundant Gröbner basis computed)

BL the "length" of the redundant Gröbner basis, i. e. the number of conses needed to represent it

AD the number of monomial additions needed to perform the algorithm

3.1 Gerdt first example

The following example comes from [13], and was already considered in [4] and[21]:

$$
\begin{array}{l}
yw - 1/2zw + tw \\
-2/7uw^2 + 10/7vw^2 - 20/7w^3 + tu - 5tv + 10tw \\
2/7yw^2 - 2/7zw^2 + 6/7tw^2 - yt + zt - 3t^2 \\
-2v^3 + 4uvw + 5v^2w - 6uw^2 - 7vw^2 + 15w^3 + 42yv \\
-14zv - 63yw + 21zw - 42tw + 147x \\
-9/7uw^3 + 45/7vw^3 - 135/7w^4 + 2zv^2 - 2tv^2 - 4zuw \\
\quad +10tuw - 2zvw - 28tvw + 4zw^2 + 86tw^2 - 42yz \\
\quad +14z^2 + 42yt - 14zt - 21xu + 105xv - 315xw \\
6/7yw^3 - 9/7zw^3 + 36/7tw^3 - 2xv^2 - 4ytw + 6ztw - 24t^2w \\
\quad +4xuw + 2xvw - 4xw^2 + 56xy - 35xz + 84xt \\
2uvw - 6v^2w - uw^2 + 13vw^2 - 5w^3 + 14yw - 28tw \\
u^2w - 3uvw + 5uw^2 + 14yw - 28tw \\
-2zuw - 2tuw + 4yvw + 6zvw - 2tvw - 16yw^2 - 10zw^2 \\
\quad +22tw^2 + 42xw \\
28/3yuw + 8/3zuw - 20/3tuw - 88/3yvw - 8zvw \\
\quad +68/3tvw + 52yw^2 + 40/3zw^2 - 44tw^2 - 84xw \\
-4yzw + 10ytw + 8ztw - 20t^2w + 12xuw - 30xvw + 15xw^2 \\
-y^2w + 1/2yzw + ytw - ztw + 2t^2w - 3xuw + 6xvw - 3xw^2 \\
8xyw - 4xzw + 8xtw
\end{array}
$$

The results are the following:

		TI	CL	CP	UP	BL	AD
S	L	1'52"	6	74	18	2616	45999
2		2'10"	7	74	18	2804	53192
H		11'02"	6	336	61	7207	203126
D		2'03"	10	74	18	2662	48376
N		2'37"	6	148	40	4272	52085
S	R	9'20"	4	294	65	7137	156429
2		9'50"	4	294	65	7181	156784
H		22'37"	4	418	81	11033	224794
N		11'01"	4	307	87	7903	159910

3.2 Gerdt second example

The example comes from [14].

$$
\begin{array}{l}
35y^4 - 30xy^2 - 210y^2z + 3x^2 + 30xz - 105z^2 \\
\quad +140yt - 21u \\
5xy^3 - 140y^3z - 3x^2y + 45xyz - 420yz^2 + 210y^2t \\
\quad -25xt + 70zt + 126yu
\end{array}
$$

The results are the following:

		TI	CL	CP	UP	BL	AD
S	L	30"	2	18	8	1703	15206
2		30"	2	18	8	1703	15206
H		38"	2	20	10	2094	15018
D		40"	2	22	10	2508	16686
N		54"	2	50	24	5549	17397
S	R	6"	2	10	6	1079	2428
2		6"	2	10	6	1079	2428
H		7"	2	10	6	1380	2428
N		6"	2	10	6	1079	2428

3.3 Gerdt third example

The example comes from [15].

$$
\begin{array}{l}
6xy^2t - x^2zt - 6xyzt + 3xz^2t - 2z^3t - 6xy^2 + 6xyz - 2xz^2 \\
-63xy^2t^2 + 9x^2zt^2 + 63xyzt^2 + 18y^2zt^2 - 27xz^2t^2 \\
\quad -18yz^2t^2 + 18z^3t^2 + 78xy^2t - 78xyzt - 18y^2zt \\
\quad +24xz^2t + 18yz^2t - 9z^3t - 15xy^2 + 15xyz - 5xz^2 \\
18x^2y^2t - 3x^3zt - 18x^2yzt + 12xy^2zt + 5x^2z^2t - 12xyz^2t \\
\quad +6xz^3t - 8z^4t - 18x^2y^2 + 18x^2yz \\
\quad -12xy^2z - 4x^2z^2 + 12xyz^2 - 6xz^3 \\
-x^2yt + 3xy^2t + 10y^3t - 15y^2zt + 3yz^2t - 3xy^2 - 10y^3 \\
\quad +xyz + 15y^2z - 5yz^2
\end{array}
$$

		TI	CL	CP	UP	BL	AD
S	L	34"	2	53	26	2280	13658
2		37"	3	53	26	2380	14226
H		1'15"	2	98	35	3318	18291
D		53"	3	74	37	3357	15076
N		Memory full after 40'					
S	R	20"	3	49	24	1573	5557
2		20"	3	49	24	1573	5557
H		47"	3	78	30	2284	9435
N		19"	2	44	20	1330	5506

3.4 Gerdt fourth example

$$
\begin{array}{l}
-2y^3z + 6x^2zt - 6xyzt + 3y^2zt - yzt^2 - 6x^2t + 6xyt - 2y^2t \\
18x^2yz^2 - 18xy^2z^2 + 18y^3z^2 - 63x^2z^2t + 63xyz^2t \\
\quad -27y^2z^2t + 9yz^2t^2 - 18x^2yz + 18xy^2z - 9y^3z \\
\quad +78x^2zt - 78xyzt + 24y^2zt - 15x^2t + 15xyt - 5y^2t \\
-8y^4z + 12x^2yzt - 12xy^2zt + 6y^3zt + 18x^2zt^2 - 18xyzt^2 \\
\quad +5y^2zt^2 - 3yzt^3 - 12x^2yt + 12xy^2t \\
\quad -6y^3t - 18x^2t^2 + 18xyt^2 - 4y^2t^2 \\
10x^3z - 15x^2yz + 3xy^2z + 3x^2zt - xzt^2 - 10x^3 + 15x^2y \\
\quad -5xy^2 - 3x^2t + xyt
\end{array}
$$

		TI	CL	CP	UP	BL	AD
S	L	38"	7	60	26	1631	11419
2		38"	6	60	26	1683	11682
H		50"	5	86	31	2209	13428
D		1'09"	13	90	42	2651	17292
N		2'33"	10	195	65	3664	33079
S	R	25"	7	50	23	1715	10161
2		25"	7	50	23	1715	9848
H		50"	8	82	29	2350	16451
N		35"	5	67	31	2063	11422

3.5 Cyclic roots

The example, due to Bjork and Fröberg, was popularized by Arnborg and Davenport, and has now become, with the corresponding examples in more variables, a classical benchmark.

$$\begin{array}{l} x+y+z+t+u \\ xy+yz+zt+tu+ux \\ xyz+yzt+ztu+tux+uxy \\ xyzt+yztu+ztux+tuxy+uxyz \\ xyztu-1 \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	3'52"	9	110	42	4383	89390
2		3'52"	9	110	42	4383	89390
H		5'20"	6	138	42	4672	118194
D		9'59"	194	123	52	7328	124509
N		Memory full after 90'					
S	R	2'30"	2	115	37	4661	57985
2		2'30"	2	115	37	4661	57985
H		2'57"	2	115	37	5143	58797
N		2'23"	2	110	41	5122	48430

3.6 Arnborg-Lazard system

This system arises in the study of Bjork-Fröberg system of degree 7, [11].

$$\begin{array}{l} x^2yz+xy^2z+xyz^2+xyz+xy+xz+yz \\ x^2y^2z+x^2yz+xy^2z^2+xyz+x+yz+z \\ x^2y^2z^2+x^2y^2z+xy^2z+xyz+xz+z+1 \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	6'03"	134	95	57	9473	98292
2		6'03"	134	95	57	9473	98292
H		10'40"	51	150	57	9900	197550
D		Memory full after 2'					
N		Memory full after 2'					
S	R	55"	3	50	26	2899	25989
2		55"	3	50	26	2899	25989
H		1'26"	3	59	26	3773	33598
N		1'24"	19	55	29	3960	38049

3.7 A parametric curve

The following is characteristic in a series of examples of the same type.

$$\begin{array}{l} x^{31}-x^6-x-y \\ x^8-z \\ x^{10}-t \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	1'38"	1	82	35	2329	44023
2		1'19"	1	85	36	2357	32388
H		8'43"	1	341	90	8857	180224
D		1'58"	1	93	36	2590	59636
N		6'39"	2	123	36	3849	256377
S	R	7"	1	53	22	684	984
2		7"	1	53	22	684	984
H		11"	1	56	22	861	1046
N		10"	1	61	26	825	1108

3.8 The Katsura-4 example

$$\begin{array}{l} 2x^2+2y^2+2z^2+2t^2+u^2-u \\ xy+2yz+2zt+2tu-t \\ 2xz+2yt+t^2+2zu-z \\ 2xt+2zt+2yu-y \\ 2x+2y+2z+2t+u-1 \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	7'48"	287	66	39	8037	63457
2		7'48"	287	66	39	8037	63457
H		10'10"	176	121	36	6979	19595
D		Memory full after 10'					
N		Memory full after 10'					
S	R	40"	12	30	12	1556	19595
2		40"	12	30	12	1556	19595
H		47"	12	30	12	1810	19595
N		40"	12	30	12	1556	19595

3.9 An example from integer programming

This example was studied in connection with integer programming, [18], [9], .

$$\begin{array}{l} x^2yz^4-t \\ x^5y^7-z^2u \\ -x^3zv+y^2 \\ -z^5w+xy^3 \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	45"	1	235	60	1206	1454
2		49"	1	242	62	1244	992
H		5'18"	1	473	116	2729	2810
D		1'41"	1	340	87	1761	1992
N		$>$ 2h	1	$>$ 2200	$>$ 600	Interrupted	
S	R	5"	1	53	17	375	246
2		5"	1	53	17	375	246
H		24"	1	107	29	745	474
N		12"	1	88	20	587	308

3.10 Another example from integer programming

The following sistem was computed only with lexicographic term-ordering, (DegRevLex is much longer) T being the highest variable, and with the following feature: whenever a polynomial is divisible by a monomial, we perform the division, and take the quotient instead of the polynomial. This is justified by the last equation, that says that in the quotient x, y, z are invertible, hence (from the other equations) also a, b, c, d, e are invertible.

$$\begin{array}{l} -y^{82}a+x^{32}z^{23} \\ x^{45}-y^{13}z^{21}b \\ y^{33}z^{12}-x^{41}c \\ -y^{33}z^{12}d+x^{22} \\ x^5y^{17}z^{22}e-1 \\ xyzT-1 \end{array}$$

		TI	CL	CP	UP	BL	AD
S	L	11'03"	1	501	169	3954	6518
2		11'32"	1	501	169	3954	10154
H		Memory full after 1h					
D		10'35"	1	354	143	2337	16616
N		6h49'	1	1664	467	3373	59544

4 Conclusions

The computations performed show that usually the sugar option is the best one, and that the double sugar option is usually almost as good. This is especially evident with the Lex term-ordering. The homogenization technique is sometimes better than the non-sugar options, but invariably worse than the sugar options. Of course, further examples can show a different behaviour; we are presently testing more examples, with longer running times (of course, long algorithms often exibit more pronounced differences).

We have not tested, when the ideal is zero-dimensional, the linear-algebra approach of [11] (this can also be generalized in arbitrary dimension). This is due to the fact that we do not have a system where linear algebra can be reasonably performed and the sugar strategy is implemented. This is planned in a short time in SCRATCHPAD-II.

This was the state of the conclusions when the paper was submitted. A referee made the following remark:

> It appears that the option H is always the best or nearly the best for the coefficient length. Reviewer's experience is that integer management becomes the most expensive part of the algorithm, for big systems. Thus the conclusions may possibly change for more difficult computations (hours on RISC stations).

This remark led us to look more closely at the two examples where this phenomenon is evident: the Katsura-4 and the Arnborg-Lazard.

As for Katsura, one sees that homogenizing the basis, one obtains a basis of the homogenized ideal: this is remarkable, since usually the original system has a big component at infinity. In this case the sugar coincides with the degree, and one should expect approximately the same developement of the algorithm for the homogenized computation and the sugar computation.

We have tested the "fussy" variant, described in the appendix, and in this case it is considerably better:

		TI	CL	CP	UP	BL	AD
F	L	5'02"	225	69	35	6351	67001

The growth of the coefficients is partly due to the fact that we do not make polynomials primitive at every step (this is costly) but only at the end of the reduction procedure: the homogenized basis has shorter reductions. However, even making polynomials primitive at every step the difference remains.

Arnborg-Lazard was the big surprise:

		TI	CL	CP	UP	BL	AD
S	L	2'50"	32	93	57	9473	73005
F		2'27"	21	99	57	8526	61946
H		8'21"	51	150	57	9900	197550

(where F stands for the fussy version). What happened, is that a new and more precise version of the useless pair elimination (written after the paper, and following more closely the appendix) has discovered two useless pairs that concentrated all the mess of coefficient growth! Remark that the useful pairs are the same for the three versions, and the homogenized version has remained the same (with a slight timings improvement that might be not significant). This pattern (a few useless pairs with very high intermediate coefficient growth) seems to be quite common. This is extremely incouraging for further research on the strategies and on the useless pair elimination criteria.

Recently Barry Trager has implemented the sugar flavor in SCRATCHPAD-II, and he says that now the direct computation of a Lex basis is almost always more convenient than the linear algebra conversion algorithm.

A selection strategy based on the degree of the S-polynomial (or rather an a priori estimate of the degree) has been suggested; this is easily implemented redefining the sugar of a polynomial of the basis as its degree. We have tested this possibility, that in many cases coincides with the sugar (maybe inverting a few computations), and sometimes has been, in out tests, considerably worse.

Appendix: the Buchberger algorithm (as implemented for our experiments)

We describe here the precise form of all the points of the Buchberger algorithm that need to be specified exactly to obtain a deterministic algorithm, in the form that we used in our experiments. This form is experimentally superior to all the other forms that were experimented.

We assume that the reader knows the algorithm, so we are very sketchy. The footnotes are for wizards only.

- Use varables x, y, z, t, u, v, w, a, b, c, d (in this order). We used two term-orderings: the pure lexicographic ordering (Lex) and the degree reverse lexicographic ordering (DegRevLex) defined as follows: monomials μ_1, μ_2 are compared in Lex ordering stating that μ_1 is larger than μ_2 if μ_1 has larger degree in the first variable for which they differ, and in DegRevLex ordering μ_1 is larger than μ_2 if μ_1 has larger total degree or they have the same total degree and μ_1 has smaller degree in the last variable for which they differ.
- The algorithm starts with a basis (a list of poloynomials), and new elements are appended to the basis. Elements, once added to the basis, are neither deleted nor modified, except in the last phase. Elements of the basis are denoted g_i, with leading term τ_i and sugar s_i
- The algorithm starts from the empty set, adding the elements of the original basis one by one, in the original order, and without modifying them.[2]
- Whenever we add an element to the basis, the set of critical pairs is adjusted. We describe here our interpretation of the Gebauer-Möller criteria, [12] that has proved to be quite efficient with respect to other interpretations.

[2] This might not be the best choice, but it was our choice in this case: interreductions and correct order of introduction might be left to a pre-processor

A critical pair is stored as $((i,j), s_{ij}, \tau_{ij})$, with (i,j) integers, (the true pair), τ_{ij} the leading term of the pair (the least common multiple of the leading terms (τ_i, τ_j)) s_{ij} the sugar of the pair, $s_{ij} = \max(s_i - \deg\tau_i, s_j - \deg\tau_j) + \deg\tau_{ij}$. We assume that we are adding the element g_k.

- Discard from the existing critical pair set all the pairs (i,j) such that τ_{ij} is strictly multiple of τ_{ik} an τ_{jk}. This is a "sloppy" variant. In the "fussy" variant one should discard these pairs only if s_{ik} and s_{jk} are not larger than s_{ij}.
- Build a new-pair set $N = \{((i,k), s_{ik}, \tau_{ik}) \mid 1 \leq i < k\}$
- If τ_i and τ_k are coprime, discard from N all pairs (j,k) such that $\tau_{jk} = \tau_{ik}$.[3]
- The following point has two variations;
  * ("Fussy"). Sort N with respect to the current strategy order; this is different in the different options, and always consists in comparing in sequence different quantities; the first quantity is the sugar (in the sugar option), the degree of τ (in the degree option) or nothing; the second is τ (in the current term-ordering), and the third is the second index of the pair; then discard from N all pairs (j,k) such that exists a preceding (i,k) such that τ_{ik} divides τjk (this is the "fussy" variant).
  * ("Sloppy"). Sort N with respect to the non-sugar version of the current strategy order; then discard from N all pairs (j,k) such that exists a preceding (i,k) such that τ_{ik} divides τjk; then sort with respect to the strategy order.

  The sloppy variant allows to discard as useless more pairs. Both variants have been implemented and tested; in most cases there was no difference; in a few cases, the differences were tiny (with the second not systematically better) and in one case the second variant was considerably better (execution time reduced 50%). The example was a very special one (a binomial ideal), so further experiments are needed to suggest the sloppy variant, that might be dangerous, since it might destroy the scheduling effects of the strategy: the result of a later pair might be used to eliminate an earlier one. See also the discussion in the conclusion section: the fussy variant seems to reduce the coefficient growth, and this might be an excellent reason to suggest it. The sloppy variant is the one that was used to build the tables.
- Merge the resulting N with the existing pair list.

(this part of the procedure is combinatorially quite complicate; one can find an alternative description that has lower complexity and gives the same result, but this description is even more complicate. This version was implemented in AlP*I* after the experiments reported in section 3, and gives considerable time saving in the examples from integer programming.)

- Choose the first element from the pair list, compute the S-polynomial (and its sugar), and totally reduce it using the existing elements of the basis as rewrite rules. Always reduce the first monomial that is simplifiable, using for that the first element of the basis that is usable.

[3] It has to be done this way: it saves a lot of work, and does not seem to destroy the strategy: the earlier pairs might need the result coming from the last one to be simplified to 0, and the last one indeed simplifies to 0. A "fussy" variant seems useless, and was not implemented.

References

[1] Attardi, G., Traverso, C., *A network implementation of Buchberger algorithm*, submitted to ISSAC-91.

[2] Bayer, D., Stillman, M., Stillman, M.:, "Macaulay User Manual," 1989.

[3] Bayer, D., Stillman, M.:, *Macaulay: A system for computation in algebraic geometry and commutative algebra*, Source and object code available for Unix and Macintosh computers. Contact the authors, or `ftp 128.103.1.107`, `Name: ftp`, `Password: any`, `cd Macaulay`, `binary`, `get M3.tar`, `quit`, `tar xf M3.tar`.

[4] W. Boege, R. Gebauer, H. Kredel, *Some examples for solving systems of algebraic equations by calculating Gröbner bases*, J. of Symbolic Computation **2** (1986), 83–98.

[5] Bjørk-Frøberg

[6] Buchberger, B., *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal*, Aequationes Mathematicae **4** (1970), 374–383. (Ph.D. Thesis, Math. Inst. Univ. Innsbruck, 1965).

[7] Buchberger, B., *A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases*, in "EUROSAM 1979," Lecture Notes in Computer Science 72, Springer Verlag, Berlin–Heidelberg–New York, 1979, pp. 3–21.

[8] Buchberger, equations...

[9] Conti, P., *Basi di Gröbner e sistemi lineari diofantei.*

[10] Conti, P., Traverso, C., *Buchberger algorithm and integer programming*, (submitted to AAECC-7).

[11] Faugère, J.C., Gianni, P., Lazard, D., Mora, T., *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering*, (submitted), J. Symb. Comp. (1989). Technical Report LITP 89–52

[12] R. Gebauer, H. M. Möller, *An installation of Buchberger's algorithm*, J. of Symbolic Computation.

[13] Gerdt, 1

[14] Gerdt, 2

[15] Gerdt, 3

[16] Giovini, A., Niesi, C., *CoCoA: a user-friendly system for commutative algebra*, in "DISCO-90," Lecture Notes in Computer Science **???**, Springer Verlag, Berlin–Heidelberg–New York, 1990.

[17] Giovini, A., Niesi, C., "CoCoA User Manual," Manual and code obtainable by anonymous `ftp` on `gauss.dm.unipi.it`, 1990.

[18] Pottier, L., *Minimal solutions of linear diophantine systems: bounds and algorithms*. 1991, to appear

[19] Robbiano, L., *Introduction to the theory of Gröbner bases*, Queen's papers in pure and applied math. (1989).

[20] Traverso, C., Donati, L., AlP*I source code for MuLISP and COMMON-LISP*, obtainable by anonymous `ftp` on `gauss.dm.unipi.it`.

[21] Traverso, C., Donati, L., *Experimenting the Gröbner basis algorithm with the* AlP*I system*, in "ISSAC 89," A. C. M., 1989.