TERMINATION OF ORIGINAL F5

VASILY GALKIN

Moscow State University

ABSTRACT. The original F_5 algorithm described in [8] 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. The scheme contains two steps: first it is shown that if the algorithm does not terminate it eventually generates two polynomials where first is a reductor for the second. But first step doesn't show that this reduction is permitted by criteria introduced in F_5. The second step shows that if such pair exists then there exists another pair for which the reduction is permitted by all criteria. Existence of such pair leads to contradiction.

1. INTRODUCTION

The Faugère's F_5 algorithm is known to be efficient method for Gröbner basis computation but one of the main problems with it's practical usage is lack of termination proof for all cases. The original paper [8] and detailed investigations in [10] states the termination for the case of reductions to zero absence, which practically means termination proof for the case of input being regular polynomial sequence. But for most input sequences the regularity is not known, so this is not enough for practical implementations termination proof. One of the approaches to solve this issue is adding of additional checks and criteria for ensuring algorithm termination. Examples are [7, 9, 4].

This paper introduces another approach for termination proof of original algorithm without any modifications. The first step of proof is based on the idea of S-pair chains which are introduced in this paper. The second step of proof is based on the method described in Theorem 21 of [3] for the proof of F5C algorithm correctness: the representation of an S-polynomial as the sum of multiplied polynomials from set computed by F5C can be iteratively rewritten using replacements for S-pairs and rejected S-pair parts until a representation with certain good properties is achieved after finitely many steps.

This article shows that the hypothesis of this method can be weakened to apply it for the set at any middle stage of F5 computations and the conclusions can be strengthened to use them for termination proof. The paper is designed as alternative termination proof for exact algorithm described in [8], so the reader is assumed to be familiar with it and all terminology with names for algorithm steps are borrowed from there.

2. POSSIBILITIES FOR INFINITE CYCLES IN F5

2.1. **Inside AlgorithmF5: d growth.** This section aims to prove the following

Claim 1. If the number of **while** cycle iteration inside AlgorithmF5 is infinite then the d value infinitely grow.

Let's suppose that there is an input $\{f_1, \ldots, f_m\}$ over $\mathcal{K}[x_1, \ldots, x_n]$ for which original F_5 does not terminate, and that it is shortest input of such kind – the algorithm do terminate on shorter input $\{f_2, \ldots, f_m\}$. This means that last iteration of outer cycle in incrementalF5 does not terminate, so some call to AlgorithmF5 does not terminate. To investigate this we need to study how the total degree d can change during execution of the cycle inside AlgorithmF5. Let's call d_i the value of d on i-th cycle iteration and extend it to $d_0 = -1$. The simple property of d_i is it's non-strict growth: $d_i \geqslant d_{i-1}$. It holds because on the $i - 1$-th iteration all polynomials in R_d have degree d_{i-1} and therefore all new generated critical pairs have degree at least d_{i-1}. Now suppose that j is number of some iteration. At the iteration j all critical pairs with degree d_j are extracted from P. After call to Reduction some new critical pairs are added to P. There exist a possibility that some of them has degree d_j. We're going to show that all such critical pairs do not generate S-polynomials in the algorithm because they are discarded.

For each new critical pair $[t, u_1, r_1, u_2, r_2]$ generated during iteration j at least one of the generating polynomials belong to R_d and no more than one belong to G_i at the beginning of the iteration. All polynomials in R_d are generated by Reduction function by appending single polynomials to $Done$. So we can select from one or two R_d-belonging generators of critical pair a polynomial r_k that was added to $Done$ later. Then we can state that the other S-pair part r_{3-k} was already present in $G \cup Done$ at the moment of r_k was added to $Done$. So the TopReduction tries to reduce r_k by r_{3-k} but failed to do this because one of IsReducible checks (a) - (d) forbids this.

From the other hand for critical pairs with degree equal to d_j we have $u_k = 1$ because total degree of critical pair is equal to total degree of it's generator r_k. This means that value u_{3-k} is equal to $\frac{\mathrm{HM}(r_k)}{\mathrm{HM}(r_{3-k})}$ so the IsReducible's rule (a) allow reduction r_k by r_{3-k}. It follows that only checks (b) - (d) are left as possibilities.

Suppose that reduction was forbidden by (b). This means that there is a polynomial in G_{i+1} that reduces $u_{3-k}\mathcal{S}(r_{3-k})$. For our case it means that in the CritPair function the same check $\varphi(u_{3-k}\mathcal{S}(r_{3-k})) = u_{3-k}\mathcal{S}(r_{3-k})$ fails and such critical pair wouldn't be created at all. So the rule (b) can't forbid reduction too.

Suppose that reduction was forbidden by (c). This means that there is a rewriting for the multiplied reductor. So for our case it means that Rewritten?(u_{3-k}, r_{3-k}) returns true at the moment of TopReduction execution, so it still returns true for all algorithm execution after this moment because rewritings do not disappear.

Suppose that reduction was forbidden by (d). The pseudo code in [8] is a bit unclear at this point, but the source code of procedure FindReductor attached to [10] is more clear and states that the reductor is discarded if both monomial of signature and index of signature are equal to those of polynomial we're reducing (it's signature monomial is r[k0][1] and index is r[k0][2] in the code):

```
if (ut eq r[k0][1]) and (r[j][2] eq r[k0][2]) then
        // discard reductor by criterion (d)
        continue;
end if;
```

For our case it means that signatures of r_k and $u_{3-k}r_{3-k}$ are equal. This leads to fact that Rewritten?(u_{3-k}, r_{3-k}) returns true after adding rule corresponding to r_k because $u_{3-k} \cdot r_{3-k}$ is rewritable by $1 \cdot r_k$. So like in case (c) Rewritten?(u_{3-k}, r_{3-k}) returns true at the moment of TopReduction execution.

Now consider Spol function execution for some S-pair with total degree d_j generated during iteration j. It executes in $j + 1$ iteration of AlgorithmF5 cycle which is far after TopReduction execution for r_k in algorithm flow so for both cases (c) and (d) call to Rewritten?(u_{3-k}, r_{3-k})

inside Spol returns true. It means that at the $j + 1$ step no S-pair with total degree d_j can add polynomial to F.

In the conclusion we have:

- the first possibility of d_{j+1} and d_j comparison is $d_{j+1} = d_j$. In this case F is empty on $j + 1$ iteration and therefore P does not contain any pairs with degree d_j after $j + 1$ iteration's finish. So $d_{j+2} > d_{j+1}$.
- the other possibility is $d_{j+1} > d_j$.

In conjunction with non-strict growth this gives $\forall j \; d_{j+2} > d_j$ which proves the claim 1.

2.2. Inside Reduction: $\mathcal{S}(h)$ growth.

This section aims to prove the following

Claim 2. Every cycle iteration inside AlgorithmF5 does terminate, in particular all calls to Reduction terminate.

We need to get a simple fact about signatures of polynomials in $ToDo$ set inside Reduction algorithm. All the critical pairs created with CritPair inside the AlgorithmF5 called for new input sequence item f_i have greater S-pair part with signature index i. So, all polynomial initially forming $ToDo$ are generated by Spol and have signature index i. The polynomials added in $ToDo$ inside Reduction are generated by TopReduction. Each such polynomial has signature $u\mathcal{S}(r_{k_1})$ that is greater than signature $\mathcal{S}(r_{k_0})$ of some other polynomial that already is $ToDo$ element because the case $u\mathcal{S}(r_{k_1}) = \mathcal{S}(r_{k_0})$ is discarded inside IsReducible (d) and the case $u\mathcal{S}(r_{k_1}) \prec \mathcal{S}(r_{k_0})$ is an execution path inside TopReduction that does not add new polynomials in $ToDo$. So signature index of all added elements are equal to i too. From the other hand all polynomials inside $ToDo$ has the same total signature d_j. This property together with index equality shows that the total degree of signature monomials is equal to the $d_j - \deg(f_i)$ for all $ToDo$ elements.

Definition 3. The reduction of labeled polynomial r_k with a labeled polynomial r_m is called *signature-safe* if $\mathcal{S}(r_k) \succ t \cdot \mathcal{S}(r_m)$, where $t = \frac{\mathrm{HM}(r_k)}{\mathrm{HM}(r_m)}$ is monomial multiplier of reductor. The reductor corresponding to signature-safe reduction is called *signature-safe reductor*.

We want to show that only possible algorithm non-termination situation correspond to the case of infinite d_j growth. We showed that non-termination leads to AlgorithmF5 does not return, and that it can't stuck in iterations with same d value. So, the only possibilities left are infinite d growth and sticking inside some iteration. We are going to show that such sticking is not possible. The AlgorithmF5 contains 3 cycles:

- **for** cycle inside Spol does terminate because it's number of iterations is limited by a count of critical pairs which is fixed at cycle beginning
- **for** cycle inside AlgorithmF5 iterating over R_d elements also does terminate because count of R_d elements is fixed at cycle beginning
- the most complex case is the **while** cycle inside Reduction which iterates until $ToDo$ becomes empty. The $ToDo$ set is extended by new elements during TopReduction execution, but all new elements have signatures greater than $\mathcal{S}(h)$ – signature of smallest $ToDo$ element at the beginning of cycle iteration. This leads to non-strict growth of $\mathcal{S}(h)$ during cycle iterations. Now the Reduction termination is got by showing two facts:
    - There is finite number of iterations with fixed smallest $ToDo$ signature $\mathcal{S}(h) = S_0$. Every step of a cycle inside Reduction performs an operation with a polynomial of signature S_0 – let's call them S_0-signature polynomials. The number of S_0-signature polynomials in $ToDo$ does not increase after first S_0-signature polynomial was taken as h because all polynomials TopReduction adds in $ToDo$ has signatures greater than smallest. The real presence of example input that produces more than

one polynomial in $ToDo$ with same signature S_0 is left as open question here. It is not needed to termination proof - only the count finiteness is crucial. The performed operation can be one of the following three operations and we are going to show that neither of them can be performed infinite number of times:

      * Some polynomial with a signature S_0 is transferred from $ToDo$ to $Done$. This can be performed only finite number of times because the count of polynomials with signature S_0 inside $ToDo$ is finite.

      * Some polynomial is signature-safe top-reduced. This can be done only finite number of steps for each of polynomials because each top-reduction $\prec$-reduces the head monomial and the set of monomials is well-ordered with $\prec$. Together with finiteness of S_0-signature polynomials count this shows that the number of such steps is finite too.

      * Set of S_0-signature polynomials is not touched, but a possibility of signature-unsafe top-reduction adds a polynomial in $ToDo$. The consecutive number of operations of this type without interleaving second type operations which change the S_0-signature polynomials is limited. This holds because the number of all signature-safe and signature-unsafe possible top-reductions for S_0-signature polynomials is limited to number of such polynomials multiplied by number of possible reducers. Therefore the fixed count of S_0-signature polynomials gives the number of polynomials to reduce and the fixed $|G \cup Done|$ gives the count of reducers. Every (polynomial-to-reduce; reducer) pair can produce no more than one new polynomial in $ToDo$ because after the first addition the multiplied reducer will be rewritten by added polynomial and IsReducible discards this pair. The limit on consecutive number of operations of this type gives a limit on overall number of iterations with $\mathcal{S}(h) = S_0$ because the finiteness of other operation types was proved.

     – All different signatures $\mathcal{S}(h)$ appeared during Reduction execution have the same index and total degree. So their count is finite because number of monomials with a fixed total degree is finite.

We got that all of the cycles inside AlgorithmF5 do finish and the claim 2 is proved.

Claims 1 and 2 together give the following result:

Claim 4. If the algorithm does not terminate for some input then the value of d infinitely grow during iterations.

3. S-PAIR-CHAINS

The claim 4 shows that algorithm non termination leads to existence of infinite sequence of nonzero labeled polynomials being added to G_i and the total degrees of polynomials in the sequence infinitely grow. So, in this case there is an infinite sequence of labeled polynomials $\{r_1, r_2, \ldots, r_m, \ldots, r_n, \ldots\}$ where $r_1, \ldots, r_m$ correspond to m input polynomials and other elements are generated either in Spol or in TopReduction. In both cases new element r_l is formed as S-polynomial of two already existing polynomials r_{i_l} and r_{j_l} as $poly(r_l) = u_l poly(r_{i_l}) - v_l poly(r_{j_l})$ and $\mathcal{S}(r_l) = u_l \mathcal{S}(r_{i_l}) \succ v_l \mathcal{S}(r_{j_l})$. The $poly(r_l)$ value can further change inside TopReduction to the polynomial with a smaller HM, but the $\mathcal{S}(r_l)$ does never change after creation. Now, we want to select an infinite sub-sequence $\{r_{k_1}, r_{k_2}, \ldots, r_{k_n}, \ldots\}$ in that sequence with the property that $r_{k_{n+1}}$ is an S-polynomial generated by $r_{k_n} = r_{i_{k_{n+1}}}$ and some other polynomial $r_{j_{k_{n+1}}}$, where r_{k_n} correspond to greater by signature S-pair part, so $\mathcal{S}(r_{k_{n+1}}) = u_{k_{n+1}} \mathcal{S}(r_{k_n})$ and

$$(3.1) \qquad\qquad \mathcal{S}(r_{k_n}) | \mathcal{S}(r_{k_{n+1}}).$$

Finite or infinite labeled polynomial sequence which successive elements satisfy property 3.1 will be called *S-pair-chain*. Every generated labeled polynomial r_l has an finite S-pair-chain ending with that polynomial. This chain can be constructed in reverse direction going from it's last element r_l by selecting every step from a given polynomial r_n a polynomial r_{i_n} which was used to generate r_n as $Spoly(r_{i_n}, r_{j_n})$. The resulting S-pair-chain has the form $\{r_q, \ldots, r_{i_{i_l}}, r_{i_l}, r_l\}$ where all polynomials has the same signature index $q = index(r_l)$ and the first element is the input polynomial of that index.

The first fact about S-pair-chains is based on the rewritten criteria and consists in the following

Theorem 5. *Every labeled S-polynomial can participate as the first element only in finite number of S-pair-chains of length 2.*

The Algorithm F5 computes S-polynomials in 2 places: in procedure SPol and in the procedure TopReduction. It's important that in both places the Rewritten? check for the part of S-Poly with greater signature is performed just before the S-polynomial is constructed. In the first case the SPol itself is checking that $(u_l; r_{i_l})$ is not rewritten, in the TopReduction the check is in the IsReducible procedure. And in both cases the computed S-polynomial is immediately added to the list of rules, as the newest element. So, at the moment of the construction of S-polynomial with signature s we can assert that the higher part of S-pair correspond to the newest rule with signature dividing s – this part even may be determined by list of rules and s without knowing anything other about computation.

Consider arbitrary labeled polynomial r_{l_0} with signature $\mathcal{S}(r_{l_0}) = s$ and an ordered by generating time subset $\{r_{l_1}, \ldots, r_{l_n}, \ldots\}$ of labeled polynomials with signatures of form $\mathcal{S}(r_{l_n}) = \mathcal{S}(r_{l_0}) \cdot u_n$. From the signature divisibility point of view all of the possibly infinite number of pairs $\{r_{l_0}, r_{l_n}\}$ can be S-pair-chains of length 2. But the ideal (u_i) in T is finitely generated by Dickson's lemma, so after some step i_0 we have $\forall i > i_0 \, \exists j_i \leqslant i_0 < i$ such that $u_{j_i} | u_i$. So $\forall i > i_0$ the sequence $\{r_{l_0}, r_{l_i}\}$ is not S-pair-chain because $\mathcal{S}(r_{l_0}) \cdot u_i$ is rewritten by $\mathcal{S}(r_{l_{j_i}}) \cdot \frac{u_i}{u_{j_i}}$ and no more than i_0 S-pair-chains of length 2 with first element r_{l_0} exist. This finishes the theorem proof. The set of ends of those 2-length S-pair-chains will be called S-pair-descendants of r_{l_0}.

Proof of existence of infinite S-pair-chain inside infinite labeled polynomial sequence require some caution while dealing with infinities, so the following definition is needed.

Definition 6. The labeled polynomial r_l is called *chain generator* if there exist infinite number of different finite S-pair-chains starting with r_l.

If the algorithm doesn't terminate the input labeled polynomial $r_1 = (f_1, 1F_1)$ is chain generator because every labeled polynomial r_l generated in the last non-terminating call to Algorithm F5 has signature index 1 so there is an S-pair-chain $\{r_1, \ldots, r_{i_{i_l}}, r_{i_l}, r_l\}$.

Now assume that some labeled polynomial r_n is known to be a chain generator. Then one of the finite number of S-pair-descendants of r_n need to be a chain generator too, because in the other case the number of different chains of length greater than 2 coming from r_n was limited by a finite sum of the finite counts of chains coming from every S-pair-descendant, and the finite number of length 2 chains coming from r_n. So, if labeled polynomial r_n is chain-generator, we can select another chain generator from it's S-pair-descendants.

Using theses result we can find infinite S-pair-chain starting with r_1 and consisting of chain generators which proves the following

Theorem 7. *If the algorithm does not terminate for some input then there exists infinite S-pair-chain $\{h_i\}$.*

For the next S-pair-chain property it is important that the polynomial can never reduce after it was used for S-pair generation as higher S-pair part. That's true because all polynomials that

potentially can reduce are stored in set $ToDo$, but all polynomials that are used as higher S-pair part are stored in G or in $Done$. So we may state that the polynomial h_n preceding polynomial h_{n+1} in the S-pair-chain retains the same $poly(h_n)$ value that was used for S-pair generation and we can state that

$$poly(h_{n+1}) = c \frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)} poly(h_n) + g_n,$$

where g_n is the polynomial corresponding to smaller part of S-pair used to generate h_{n+1} from h_n and satisfy:

$$(3.2) \qquad \mathrm{HM}(h_{n+1}) < \mathrm{HM}\left(\frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)} h_n\right) = \mathrm{HM}(g_n),\ \mathcal{S}(h_{n+1}) = \mathcal{S}\left(\frac{\mathcal{S}(h_{n+1})}{\mathcal{S}(h_n)} h_n\right) \succ \mathcal{S}(g_n).$$

The monomial ordering can be transitively extended to monomial quotients: $\frac{m_1}{m_2} > \frac{m_3}{m_4} \Leftrightarrow m_1 m_4 > m_3 m_2$. From the first inequality in 3.2 we can get $\frac{\mathrm{HM}(h_n)}{\mathcal{S}(h_n)} > \frac{\mathrm{HM}(h_{n+1})}{\mathcal{S}(h_{n+1})}$, so in the S-pair-chain the quotients $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)}$ are strictly descending according to term ordering. This fact can't be used directly to show chains finiteness because unlike the ordering of monomials the ordering of monomial quotients is not well ordering – for example the sequence $\frac{x}{x} > \frac{x}{x^2} > \cdots > \frac{x}{x^n} > \cdots$ is infinitely decreasing.

There is two possible cases for relation between HMs of consecutive elements. We have $\mathcal{S}(h_n)|\mathcal{S}(h_{n+1})$, so they are either equal or $\deg(h_n) < \deg(h_{n+1})$. For the first case $\mathrm{HM}(h_{n+1}) < \mathrm{HM}(h_n)$ with the equal total degrees for the other case $\mathrm{HM}(h_{n+1}) > \mathrm{HM}(h_n)$ because total degrees are different. So the sequence of infinite S-pair-chain HMs consists of blocks with fixed total degrees where HMs inside a block are strictly decreasing. Block lengths can be equal to one and the total degree of blocks are increasing. This leads to the following properties: S-pair-chain $\{h_i\}$ can't contain elements with equal HMs and $\mathrm{HM}(h_i)|\mathrm{HM}(h_j)$ is possible only for $i < j$ and $\deg(h_i) < \deg(h_j)$.

This allows us to use technique analogous to Proposition 14 from [1]: consider HMs of infinite S-pair-chain $\{h_i\}$. They form an infinite sequence in T, so by Dickson's lemma there exists 2 polynomials in sequence with $\mathrm{HM}(h_i)|\mathrm{HM}(h_j)$. Therefore from the previous result we have $i < j$ and with the S-pair chain properties we get $\mathcal{S}(h_i)|\mathcal{S}(h_{i+1})|\cdots|\mathcal{S}(h_j)$ and $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)} > \frac{\mathrm{HM}(h_{i+1})}{\mathcal{S}(h_{i+1})} > \cdots > \frac{\mathrm{HM}(h_j)}{\mathcal{S}(h_j)}$. So, we get the following result:

Theorem 8. *If the algorithm does not terminate for some input then after some finite step the set G contains a pair of labeled polynomials f, g with g generated after f that satisfies the following 3 properties:*

$$\mathrm{HM}(f)|\mathrm{HM}(g),$$
$$\mathcal{S}(f)|\mathcal{S}(g),$$

$$\frac{\mathrm{HM}(f)}{\mathcal{S}(f)} > \frac{\mathrm{HM}(g)}{\mathcal{S}(g)},$$

where the $<$ corresponds to the term ordering.

The proof of theorem 8 does not rely on any correspondence between orderings on signatures and terms. But the original algorithm F_5 uses the same ordering for both cases and now we utilize this fact and make a transition from one to other to get relation on signatures $\mathcal{S}(g) \succ t \cdot \mathcal{S}(f)$, where $t = \frac{\mathrm{HM}(g)}{\mathrm{HM}(f)} \in T$. The last inequality shows that tf can be used as a reducer for g in TopReduction from the signature point of view – i.e. it satisfy checks (a) and (d) inside IsReducible and it's signature is smaller. In the absence of criteria this would directly lead to contradiction because at the time g was added to the set G labeled polynomial f already had

been there, so the TopReduction should had reduced g by f. So we get the following result about termination:

Fact 9. *If no polynomials are rejected by criteria checks (b) and (c) inside IsReducible the algorithm does terminate.*

But the existence of criteria allow the situation in which tf is rejected by criteria checks in (b) or (c) inside IsReducible. The idea is to show that even in this case there can be found another possible reducer for g that is not rejected and anyway lead to contradiction, so for the following parts of paper g is treated as some fixed labeled polynomial with signature index 1. If the algorithm does not terminate then g is assumed additionally to be a polynomial from theorem 8. Most of the following theory is formulated for the objects with signatures $\prec \mathcal{S}(g)$.

4. S-pairs with signatures smaller than $\mathcal{S}(g)$

To prove the idea let us work with algorithm state just before adding g to $Done$ during call to AlgorithmF5 with $i = 1$. Consider a finite set $G_1 \cup Done$ of labeled polynomials at that moment. This set contains positions of labeled polynomials in R, so it's elements can be ordered according to position in R and written as an ordered integer sequence $G_g = \{b_1, \ldots, b_N\}$ with $b_j < b_{j+1}$. It should be noted that this order does correspond to the order of labeled polynomials in the sequence produced by concatenated rule arrays $Rule[m] : Rule[m-1] : \cdots : Rule[1]$ because addition of new polynomial to R is always followed by addition of corresponding rule. But this order may differ from the order polynomials were added to $G_1 \cup Done$ because polynomials with same total degree are added to $Done$ in the increasing signature order, while the addition polynomials with same total degree to R is performed in quite random order inside Spol and TopReduction procedures. For the simplicity we will be speaking about labeled polynomials b_j from G_g, assuming that G_g is not the ordered positions list but the ordered list of labeled polynomials themselves corresponding to those positions. In this terminology we can say that all input polynomials $\{f_1, \ldots, f_m\}$ do present in G_g, because they all present in G_1 at the moment of its creation.

S-pairs of G_g elements are processed in two paths in the algorithm. The main path is for S-pairs with total degree greater than total degrees of polynomials generated it. Such S-pairs are processed in the following order:

- in the AlgorithmF5 they are passed in CritPair function while moving elements to G_i from $R_d = Done$ or while processing input polynomial r_i.
- The CritPair function either discards them by normal form check φ or adds to P
- The S-pair is taken from P and passed to SPol function
- The SPol function either discards them by Rewritten? check or adds S-polynomial to $F = ToDo$
- At some iteration the Reduction procedure takes S-polynomial from $ToDo$, performs some signature-safe reductions and adds result to $Done$.

The other processing path is for special S-pairs corresponding to reductions forbidden by algorithm – the case when S-pair is generated by polynomials r_{i_1} and r_{j_1} such that $\mathrm{HM}(r_{j_1})|\mathrm{HM}(r_{i_1})$ so that S-polynomial has a form $poly(r_{i_1}) - v_1 poly(r_{j_1})$. Such situation is possible for two G_g elements if reduction of r_{i_1} by r_{j_1} was forbidden in TopReduction or IsReducible. So for this case the path of S-pair "processing" is the following:

- The S-pair part $v_1 r_{j_1}$ is checked in IsReducible. (a) is satisfied because $\mathrm{HM}(r_{j_1})|\mathrm{HM}(r_{i_1})$. It can be rejected by one of the other checks:
    - Rejection by check (b) correspond to the normal form check φ for $v_1 r_{j_1}$
    - Rejection by check (c) correspond to the Rewritten? check for $v_1 r_{j_1}$

- Rejection by check (d) means that either $v_1 r_{j_1}$ or $1 r_{i_1}$ can be rewritten by other, so if S-pair was not rejected by check (c) this type of rejection means that S-pair part $1 r_{i_1}$ fails to pass Rewritten? check.
- The non-rejected in IsReducible S-pair is returned in the TopReduction. Signature comparison in TopReduction forbids the reduction of r_{i_1} by r_{j_1} and returns computed S-polynomial corresponding to the S-pair in the set $ToDo_1$
- The Reduction procedure add this polynomial in $ToDo$
- The last step is equal to for both processing paths: at some iteration the Reduction procedure takes S-polynomial from $ToDo$, performs some signature-safe reductions and adds result to $Done$

It can be seen that after S-pair processing termination every S-pair is either reduced and added to $Done$ or one of its S-pair parts is rejected by normal form φ or Rewritten? check. Some S-pairs can be processed by processing paths multiple times, for example this is done in the second iteration inside Algorithm F5 with same d value. But this doesn't matter because the start and finish of the processing path one time guarantees that S-pair is rejected by Rewritten? check in all other processing attempts.

The processing path is not a single procedure and for the case of algorithm infinite cycling some S-pairs are always staying in the middle of the path having S-pair queued in P or S-polynomial in $ToDo$. So we have to select S-pairs which processing is already finished at the fixed moment we studying. The elements from P and $ToDo$ in AlgorithmF5 and Reduction procedures are taken in the order corresponding to growth of their signatures. So S-pairs with signature smaller than $\mathcal{S}(g)$ can be split in the following classes:

- S-pairs with signature w such that $index(w) > index(\mathcal{S}(g)) = 1$. They were processed on previous calls of AlgorithmF5.
- S-pairs with signature w such that $index(w) = index(\mathcal{S}(g)) = 1, \deg(w) < \deg(\mathcal{S}(g))$. They were processed on previous iterations inside the call to AlgorithmF5 that is processing g.
- S-pairs with signature w such that $index(w) = index(\mathcal{S}(g)) = 1, \deg(w) = \deg(\mathcal{S}(g)), w \prec \mathcal{S}(g)$ .They were processed on previous iterations inside the call to Reduce that is processing g.

S-pairs from these classes can't be at the middle of processing path because at the studied state of algorithm the processing is just finished for g so P and $ToDo$ sets does not contain any non-processed elements with signatures smaller $\mathcal{S}(g)$. The only left thing to show is proof that processing was started at least one time for all for S-pairs. This is true for first two classes: the processing of corresponding S-pairs was started at least one time with the call to CritPair inside AlgorithmF5 just before the greatest of S-pair generators was added to G. For S-pairs of third class the situation depend on the total degrees of its generators. If both generators of S-pair have total degrees $< \deg(g)$ then its processing is started in CritPair like for the S-pairs from first two classes. But some S-pairs from the third class can have a signature-greater generator polynomial r_{i_1} such that $\deg(r_{i_1}) = \deg(g), \mathcal{S}(r_{i_1}) \prec \mathcal{S}(g)$. They are processed with the second mentioned processing path so the processing for such S-pairs is not yet started at the beginning of last Reduction call. Fortunately, their processing starts inside Reduction before fixed moment we studying: the procedure selects polynomials from $ToDo$ in the signature increasing order, so r_{i_1} is reduced before g and during r_{i_1} reduction just before putting r_{i_1} to $Done$ a call to IsReducible starts processing for all such S-pairs.

This finishes proof of the following result mentioning properties which are very similar to the properties used in Theorem 21 in [3]:

Theorem 10. *At the moment of adding g to Done every S-pair of G_g elements which signature is smaller than $\mathcal{S}(g)$ satisfies one of three properties:*

(1) S-pair has a part that is rejected by the normal form check φ (in CritPair or in IsReducible). Such S-pairs will be referenced as *S-pairs with a part that satisfies F5 criterion*.

(2) S-pair has a part that is rejected by the Rewritten? check (in SPol or in IsReducible). Such S-pairs will be referenced as *S-pairs with a part that satisfies Rewritten criterion*.

(3) S-pair was not rejected, so it's S-polynomial was signature-safe reduced by G_g elements and the result is stored in G_g. Such S-pairs will be referenced as *S-pairs with a computed G_g-representation*.

The ideas of *satisfying F5 criterion* and *satisfying Rewritten criterion* can be extended to arbitrary monomial-multiplied labeled polynomial sh, $h \in G_g$:

Definition 11. The monomial-multiplied labeled polynomial sr_i, $r_i \in G_g$ is called *satisfying F5 criterion* if $\varphi_{index(r_i)+1}(s\mathcal{S}(r_i)) \neq s\mathcal{S}(r_i)$, where $\varphi_{index(r_i)+1}$ is operator of normal form w.r.t $G_{index(r_i)+1}$.

This definition is equivalent to sr_i being non-normalized labeled polynomial according to definition 2 in part 5 of [8].

Definition 12. The monomial-multiplied labeled polynomial sr_i, $r_i \in G_g$ is called *satisfying Rewritten criterion* if $\exists j > i$ such that $\mathcal{S}(r_j)|s\mathcal{S}(r_i)$.

For the case sr_i is the S-pair part these definitions are equivalent to the checks in the algorithm in a sense that S-pair part is rejected by the algorithm if and only if it satisfies the definition as monomial-multiplied labeled polynomial. Note that for both criteria holds important property that if sr_i satisfies a criteria then a further multiplied s_1sr_i satisfies it too.

5. Representations

5.1. **Definition.** The idea of representations comes from proof of Theorem 21 in [3]. Representations are used to order all possible ways how a labeled polynomial p can be written as an element of (G_g) ideal. The single representation corresponds to writing a labeled polynomial p as any finite sum of the form

$$(5.1) \qquad p = \sum_k m_k \cdot b_{i_k}, \ b_{i_k} \in G_g$$

with coefficients $m_k = c_k t_k \in \mathcal{K} \times T$.

Definition 13. Sum of the form 5.1 with all pairs (t_k, b_{i_k}) distinct is called *G_g-representation* of p. The symbolic products $m_k \cdot b_{i_k}$ are called the *elements* of representation. If we treat this symbolic product as multiplication we get an labeled polynomial $m_k b_{i_k}$ corresponding to the representation element. So p is equal to sum of labeled polynomials, corresponding to elements of its representation. Also the term *element signature* will be used for signature of labeled polynomials corresponding to the element. Two representations are equal if the sets of their elements are equal.

Most representations we are interested in have the following additional property limiting elements signature:

Definition 14. The G_g-representation of p is called *signature-safe* if $\forall k \, \mathcal{S}(m_k b_{i_k}) \prec \mathcal{S}(p)$.

5.2. **Examples.**

Example 15. The first important example of a G_g-representation is trivial: the labeled polynomial from G_g is equal to sum of one element, identity-multiplied itself:

$$b_j = 1 \cdot b_j.$$

This G_g-representation is signature-safe. The prohibition of two elements which have same monomial t_k and polynomial b_{i_k} ensures that all elements of representation that differ only in field coefficient c_k are combined together by summing field coefficients. So expressions like $b_j = -1 \cdot b_j + 2 \cdot b_j$ and $b_j = 2x \cdot b_k + 1 \cdot b_j - 2x \cdot b_k$ are not valid G_g-representations.

Example 16. A labeled polynomial $b_j \in G_g$ multiplied by arbitrary polynomial h also have a simple G_g-representation arising from splitting h into terms: $h = \sum_k m_k$, $m_k \in \mathcal{K} \times T$. This G_g-representation has form

(5.2)
$$b_j h = \sum_k m_k \cdot b_j$$

and is signature-safe too.

A labeled polynomial can have arbitrary number of representations: for example we can add elements corresponding to a syzygy to any representation and combine elements with identical monomials and polynomials to get the correct representation. The result will be representation of the same polynomial because sum of syzygy elements is equal to 0.

Example 17. The product of two polynomial from G_g has two representations of the form (5.2) which differs in syzygy addition:

$$b_j b_i = \sum_k m_{i_k} \cdot b_j = \sum_k m_{i_k} \cdot b_j + 0 = \sum_k m_{i_k} \cdot b_j + \left(\sum_k m_{j_k} \cdot b_i - \sum_k m_{i_k} \cdot b_j \right) = \sum_k m_{j_k} \cdot b_i,$$

where m_{i_k} are terms of b_i and m_{j_k} are terms of b_j.

Example 18. The zero polynomial has an empty representation and an representation for every syzygy:

$$0 = \sum_{\emptyset} (\text{empty sum}) = \sum_k m_{j_k} \cdot b_i + \sum_k (-m_{i_k}) \cdot b_j,$$

where m_{i_k} and m_{j_k} are same as above.

Another important example of G_g-representation comes from ideal and signature definitions. All labeled polynomials computed by the algorithm are elements of ideal $(f_1, \ldots, f_m)$. So any labeled polynomial p can be written as $\sum_i f_i g_i$, where g_i are homogeneous polynomials. All input polynomials f_i belong to G_g, so $f_i g_i$ has G_g-representations of the form (5.2).

Example 19. Those representations sum give the following signature-safe representation:

$$p = \sum_k m_k \cdot b_{i_k}, \, m_k \in \mathcal{K} \times T, \, b_{i_k} \in \{f_1, \ldots, f_m\} \subset G_g.$$

Definition 20. This particular case of G_g-representation where b_{i_k} are limited to input polynomials will be called *input-representation*.

The definition of signature combined with this input-representation gives the following result:

Claim 21. A labeled polynomial p with known signature $\mathcal{S}(p)$ has an input-representation consisting of an element $c\mathcal{S}(p) \cdot f_{index(p)}$ and some other elements with smaller signatures.

This property is special to input-representations because generic G_g-representations can have multiple elements with same maximal signature because it is possible to have $m_1\mathcal{S}(b_{i_1}) = m_2\mathcal{S}(b_{i_2})$ while $i_1 \neq i_2$.

Example 22. The last example comes from S-pairs with a computed G_g-representation. S-polynomial of b_{i_l} and b_{j_l} from G_g is $p = u_l poly(b_{i_l}) - v_l poly(b_{j_l})$. It is known from reduction process that for such S-pairs p is signature-safe reduced and the result is added to G_g as some labeled polynomial b_l. So the G_g-representation is:

$$p = \sum_k m_k \cdot b_{n_k} + 1 \cdot b_l,$$

where signatures of $m_k \cdot b_{n_k}$ elements are smaller than $\mathcal{S}(b_l) = \mathcal{S}(p)$. The value of l is position of b_l in ordered list G_g. In this representation l is greater than i_l and j_l because corresponding labeled polynomial b_l is added to R at the moment of S-polynomial computation in Spol or TopReduction so the polynomials b_{i_l} and b_{j_l} used to create S-pair already present in R at that moment and the order of G_g correspond to order of R.

5.3. Ordering representations.

Definition 23. To order G_g-representations we start from *representation elements ordering*: $c_i t_i \cdot b_i \gg c_j t_j \cdot b_j$ iff $t_i \mathcal{S}(b_i) \succ t_j \mathcal{S}(b_j)$ or $t_i \mathcal{S}(b_i) = t_j \mathcal{S}(b_j)$ and $i < j$ (note the opposite order).

This ordering is based only on comparison of signatures and positions of labeled polynomials in the ordered list G_g but does not depend on the field coefficient. The only case in which two elements can't be ordered is equality of both signatures $t_i \mathcal{S}(b_i) = t_j \mathcal{S}(b_j)$ and positions in list $i = j$. Position equality means $b_i = b_j$ which in conjunction with signature equality gives $t_i = t_j$. So any two elements that belong to a single G_g-representation are comparable with $\ll$ order because they have distinct (t_k, b_k) by definition. Below are given some examples of $\ll$ element ordering for the 3-element list $G_g = \{b_1, b_2, b_3\}$ with ordering $x\mathbf{F}_i \succ y\mathbf{F}_i$ and signatures $\mathcal{S}(b_1) = \mathbf{F}_1, \mathcal{S}(b_2) = \mathbf{F}_2, \mathcal{S}(b_3) = x\mathbf{F}_1$.

- $y \cdot b_1 \gg 100y \cdot b_2$ because signature of left side is $\succ$
- $x \cdot b_1 \gg y \cdot b_1$ because signature of left side is $\succ$
- $-x \cdot b_1$ and $2x \cdot b_1$ are not comparable because signatures and list indices are equal
- $y^2 \cdot b_1 \ll y \cdot b_3$ because signature of left side is $\prec$
- $x^2 \cdot b_1 \gg x \cdot b_3$ because signatures are equal and the list position of left side's labeled polynomial is 1 which is smaller than right side's position 3.

To extend this order to entire G_g-representations consider *ordered form* of representation consisting of all its elements written in a list with $\gg$-decreasing order. This form can be used for equality testing because if two representations are equal then they have exactly equal ordered forms.

Definition 24. With ordered forms the *G_g-representations ordering* can be introduced: the representation $\sum_k m_{i_k} \cdot b_{i_k}$ is $\ll$-smaller than $\sum_k m_{j_k} \cdot b_{j_k}$ iff the ordered form of the first representation is smaller than second's according to lexicographical extension of $\ll$ ordering on elements. For the corner case of the one ordered form being beginning of the other the shorter form is $\ll$-smaller. If the greatest different elements of ordered forms differ only in field coefficient the representations are not comparable.

Some examples of this ordering are given for the same as above 3-element G_g list. Note that all G_g-representations are already written in ordered forms:

- $x^2 \cdot b_1 + xy \cdot b_1 + y^2 b_1 \gg x^2 \cdot b_1 + 100y^2 \cdot b_1$ because $xy \cdot b_1 \gg y^2 \cdot b_1$

- $x^2 \cdot b_1 + 100y^2 \cdot b_1 \succ x^2 \cdot b_1$ because the right ordered form is beginning of the left
- $x^2 \cdot b_1 \succ xy \cdot b_1 + y^2 \cdot b_1 + x^2 \cdot b_2$ because $x^2 \cdot b_1 \succ xy \cdot b_1$
- $xy \cdot b_1 + y^2 \cdot b_1 + x^2 \cdot b_2 \succ y \cdot b_3 + y^2 \cdot b_1 + x^2 \cdot b_2$ because $xy \cdot b_1 \succ y \cdot b_3$
- $y \cdot b_3 + y^2 \cdot b_1 + x^2 \cdot b_2$ and $2y \cdot b_3 + y^2 \cdot b_2$ are not comparable because the greatest different elements are $y \cdot b_3$ and $2y \cdot b_3$.

The ordering is compatible with signature-safety:

Theorem 25. *If two representations of p has a relation $\sum_k m_{i_k} \cdot b_{i_k} \lessdot \sum_k m_{j_k} \cdot b_{j_k}$ and the second one is signature-safe representation then the first one is signature-safe too.*

This theorem quickly follows from a fact that elements of a $\lessdot$-smaller representation can't has signatures $\succ$-greater than signatures of $\gtrdot$-greater representation.

The number of different labeled polynomial positions is finite because it is equal to $|G_g|$ which is finite for fixed g. So the existence of infinite $\gtrdot$-descending sequence of representation elements would lead to existence of infinite $\succ$-descending sequence of signatures. Combining this with well-orderness of signatures with ordering $\prec$ we get the proof for well-orderness of elements with ordering $\lessdot$.

The straightforward proof for well-orderness of representations with $\lessdot$ following from well-orderness of elements with $\lessdot$ is not very complex but to skip its strict details the theorem 2.5.5 of [5] will be used. It states well-orderness of finite multiset with an lexicographically extended ordering of well-ordered elements. This applies to the representations because they form a subset in the finite multiset of representation elements. So, we have

Theorem 26. *The representations are well-ordered with $\lessdot$ ordering.*

5.4. **Sequence of representations.** The idea of this part is constructing a finite sequence of strictly $\lessdot$-descending signature-safe G_g-representations for a given labeled polynomial mh, $m \in \mathcal{K} \times T$, $h \in G_g$ with $\mathcal{S}(mh) \prec \mathcal{S}(g)$. The first signature-safe representation in the sequence is $mh = m \cdot h$, the last representation is $mh = \sum_k m_k \cdot b_{i_k}$ with elements having the following properties $\forall k$:

(1) $m_k b_{i_k}$ does not satisfy F5 criterion.
(2) $m_k b_{i_k}$ does not satisfy Rewritten criterion.
(3) $\mathrm{HM}(m_k b_{i_k}) \leqslant \mathrm{HM}(mh)$

The proof of such sequence existence is very similar to Theorem 21 of [3] and is based on a fact, that if a some signature-safe representation of mh contains an element $m_{k_0} \cdot b_{i_{k_0}}$ not having one of the properties then a $\lessdot$-smaller representation can be constructed. The exact construction differ for three cases but the replacement scheme is the same:

- a some element $m_{k'_0} \cdot b_{i_{k'_0}}$ in mh representation is selected. Note that k'_0 in some cases is not equal to k_0
- some representation $m_{k'_0} b_{i_{k'_0}} = \sum_l m_l \cdot b_{i_l}$ is constructed for this element.
- it is shown that constructed representation is $\lessdot$-smaller than representation $m_{k'_0} b_{i_{k'_0}} = m_{k'_0} \cdot b_{i_{k'_0}}$

Now $m_{k'_0} \cdot b_{i_{k'_0}}$ in $mh = \sum_k m_k \cdot b_{i_k}$ is replaced by $\sum_l m_l \cdot b_{i_l}$ and after combining coefficients near elements with both monomial and polynomial equal a modified representation for mh appears. Is is $\lessdot$-smaller than $mh = \sum_k m_k \cdot b_{i_k}$ because all elements $\gtrdot$-greater than $m_{k_0} \cdot b_{i_{k_0}}$ are identical in both representations if they present. The element $m_{k'_0} \cdot b_{i_{k'_0}}$ is contained in original representation but not in the modified. And all other elements in representations are $\lessdot$-smaller than $m_{k'_0} \cdot b_{i_{k'_0}}$ so they doesn't influence the comparison. The comparison holds even in a corner case when all elements are discarded while combining coefficients. This case can appear if the original

representation is equal to $mh = m_{k_0'} \cdot b_{i_{k_0'}} + \sum_l (-m_l) \cdot b_{i_l}$ what leads to modified representation $mh = 0$ with zero elements which is $\prec$-smaller than any non-empty representation. This result can be expressed in the following

Lemma 27. *If an element $m_{k_0'} \cdot b_{i_{k_0'}}$ of signature-safe representation $mh = \sum_k m_k \cdot b_{i_k}$ has an representation $m_{k_0'} b_{i_{k_0'}} = \sum_l m_l \cdot b_{i_l}$ which is $\prec$-smaller than representation $m_{k_0'} b_{i_{k_0'}} = m_{k_0'} \cdot b_{i_{k_0'}}$ then mh has a signature-safe representation $\prec$-smaller than $mh = \sum_k m_k \cdot b_{i_k}$.*

Now it will be shown that replacement scheme can be performed if the representation contains an element not satisfying at least one of three properties. An element not having the first property does satisfy the F5 criterion and the idea is to use that $m_{k_0} \mathcal{S}(b_{i_{k_0}})$ is not the minimal signature of $m_{k_0} b_{i_{k_0}}$ like in Theorem 20 of [3]. $k_0' = k_0$ is taken for this case. Consider input-representation of $m_{k_0} b_{i_{k_0}}$ with signature of $\succ$-maximal element equal to $m_{k_0} \mathcal{S}(b_{i_{k_0}}) = s_0 \mathbf{F}_{j_0}$:

$$(5.3) \qquad m_{k_0} b_{i_{k_0}} = c_0 s_0 \cdot f_{j_0} + \sum_l m_l \cdot f_{i_l}.$$

From the satisfying F5 criterion s_0 can be expressed like $s_0 = s_1 \mathrm{HM}(f_{j_1})$, $j_1 > j_0$ so $s_0 f_{j_0} = s_1 f_{j_0} f_{j_1} - s_1(f_{j_1} - \mathrm{HM}(f_{j_1})) f_{j_0}$. From this we can write another representation for $m_{k_0} b_{i_{k_0}}$, assuming m_{0i} are sorted terms of f_{j_0}, m_{1i} are sorted terms of f_{j_1} and N_0, N_1 are number of terms in those polynomials:

$$m_{k_0} b_{i_{k_0}} = \sum_{i=1}^{N_0} c_0 s_1 m_{0i} \cdot f_{j_1} + \sum_{i=2}^{N_1} -c_0 s_1 m_{1i} \cdot f_{j_0} + \sum_l m_l \cdot f_{i_l}.$$

This representation is $\prec$-smaller than $m_{k_0} \cdot b_{i_{k_0}}$ because signatures of all elements are smaller than $s_0 \mathbf{F}_{j_0}$. For the elements of the third sum $\sum_l m_l \cdot f_{i_l}$ this follows from **5.3**, where those elements are smaller elements of input-representation. For the elements of the first sum $\sum_{i=1}^{N_0} c_0 s_1 m_{0i} \cdot f_{j_1}$ this follows from the position inequality $j_1 > j_0$. And for the second sum we use the equality in term and signature orderings: all terms m_{1i}, $i \geqslant 2$ are smaller than m_{11}, so the signatures are: $s_1 m_{1i} \mathbf{F}_{j_0} \prec s_1 m_{11} \mathbf{F}_{j_0} = s_0 \mathbf{F}_{j_0}$. This gives

Lemma 28. *If a signature-safe G_g-representation $mh = \sum_k m_k \cdot b_{i_k}$ does not satisfy property 1 then there exists an element $m_{k_0'} \cdot b_{i_{k_0'}}$ having G_g-representation $m_{k_0'} b_{i_{k_0'}} = \sum_l m_l \cdot b_{i_l}$ which is $\prec$-smaller than representation $m_{k_0'} b_{i_{k_0'}} = m_{k_0'} \cdot b_{i_{k_0'}}$.*

For the elements not satisfying case 2 the $\prec$-smaller representation is created in a way used in Proposition 17 of [3]. $k_0' = k_0$ is taken for this case too. Assume that $\mathcal{S}(m_{k_0} b_{i_{k_0}}) = s_0 \mathbf{F}_{j_0}$ and it is rewritten by labeled polynomial $b_{i'}$ from R. Because the representation is signature-safe we have $\mathcal{S}(b_{i'}) \preccurlyeq s_0 \mathbf{F}_{j_0} \preccurlyeq \mathcal{S}(mh) \prec \mathcal{S}(g)$. So $b_{i'}$ was processed in TopReduction before g. Since $b_{i'}$ is rewriter we have $b_{i'} \neq 0$. All this gives the fact that $b_{i'}$ does present not only in R but in G_g too so it can be used as a polynomial of G_g-representation element. From the Rewritten criterion definition we know that $i' > i_{k_0}$ and the existence of $s' \in T$ such that $s' \mathcal{S}(b_{i'}) = s_0 \mathbf{F}_{j_0}$. So, for the $m_{k_0} b_{i_{k_0}}$ there is an input-representation **5.3** and for the $s' b_{i'}$ the input-representation is:

$$s' b_{i'} = c' s_0 \cdot f_{j_0} + \sum_{l'} m_{l'} \cdot f_{i_{l'}}.$$

A G_g-representation for $c_0 s_0 f_{j_0}$ can be acquired with transformation of the above expression:

$$c_0 s_0 f_{j_0} = c'^{-1} c_0 s' \cdot b_{i'} + \sum_{l'} -c'^{-1} c_0 m_{l'} \cdot f_{i_{l'}}.$$

Using this to replace the first element in 5.3 we get the wanted result:

$$m_{k_0} b_{i_{k_0}} = c'^{-1} c_0 s' \cdot b_{i'} + \sum_{l'} -c'^{-1} c_0 m_{l'} \cdot f_{i_{l'}} + \sum_l m_l \cdot f_{i_l}$$

It is $\lessdot$-smaller than $m_{k_0} b_{i_{k_0}} = m_{k_0} \cdot b_{i_{k_0}}$ because elements of both sums has signatures smaller than $s_0 \mathbf{F}_{j_0}$, and for the first element $\mathcal{S}(c'^{-1} c_0 s' \cdot b_{i'}) = \mathcal{S}(m_{k_0} \cdot b_{i_{k_0}}) = s_0 \mathbf{F}_{j_0}$ but $i' > i_{k_0}$, so applying the $\lessdot$-comparison rule for equal signatures and different list positions we get that element $c'^{-1} c_0 s' \cdot b_{i'}$ is $\lessdot$-smaller than $m_{k_0} \cdot b_{i_{k_0}}$ too. This gives

Lemma 29. *If a signature-safe G_g-representation $mh = \sum_k m_k \cdot b_{i_k}$ with $\mathcal{S}(mh) \prec \mathcal{S}(g)$ does not satisfy property 2 then there exists an element $m_{k_0'} \cdot b_{i_{k_0'}}$ having G_g-representation $m_{k_0'} b_{i_{k_0'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{k_0'} b_{i_{k_0'}} = m_{k_0'} \cdot b_{i_{k_0'}}$.*

Now assume, that the all representation elements satisfy properties 1 and 2, but there exists at least one element $m_{k_0} \cdot b_{i_{k_0}}$ that does not satisfy property 3. Let $m_{\max}$ be the maximal HM of labeled polynomials corresponding to representation elements and $H_{\max}$ be a list of elements where $m_{\max}$ is achieved. Select k_0' to be the index of the $\gg$-greatest representation element in $H_{\max}$. We have $\mathrm{HM}(m_{k_0'} b_{i_{k_0'}}) = m_{\max} \geqslant \mathrm{HM}(m_{k_0} b_{i_{k_0}}) > \mathrm{HM}(mh)$, so the HM of sum of all elements except k_0' is equal to $\mathrm{HM}(mh - m_{k_0'} b_{i_{k_0'}}) = \mathrm{HM}(m_{k_0'} b_{i_{k_0'}}) = m_{\max}$, so there is another element k_1' having $\mathrm{HM}(m_{k_1'} b_{i_{k_1'}}) = m_{\max}$. So, $m_{k_1'} \cdot b_{i_{k_1'}} \in H_{\max}$ and $m_{k_1'} \cdot b_{i_{k_1'}} \lessdot m_{k_0'} \cdot b_{i_{k_0'}}$ because of $m_{k_0'} \cdot b_{i_{k_0'}}$ $\gg$-maximality in $H_{\max}$.

The $\mathrm{HM}(m_{k_1'} b_{i_{k_1'}}) = \mathrm{HM}(m_{k_0'} b_{i_{k_0'}})$ means that a critical pair of $b_{i_{k_0'}}$ and $b_{i_{k_1'}}$ has the form $[m'^{-1} m_{\max}, m'^{-1} m_{k_0'}, b_{i_{k_0'}}, m'^{-1} m_{k_1'}, b_{i_{k_1'}}]$ where $m' = \gcd(m_{k_0'}, m_{k_1'})$. Let q be corresponding S-polynomial. Then $m' \mathcal{S}(q) \preccurlyeq \mathcal{S}(mh) \prec \mathcal{S}(g)$ because $m' \mathcal{S}(q) = \mathcal{S}(m_{k_0'} b_{i_{k_0'}})$ and the representation is signature-safe. The S-polynomial parts $m'^{-1} m_{k_0'} b_{i_{k_0'}}$ and $m'^{-1} m_{k_1'} b_{i_{k_1'}}$ doesn't satisfy F5 and Rewritten criteria because their forms multiplied by m' are $m_{k_0'} b_{i_{k_0'}}$ and $m_{k_1'} b_{i_{k_1'}}$ – labeled polynomials corresponding to elements which are known not to satisfy both criteria by assumption. Therefore $m' \mathcal{S}(q) \prec \mathcal{S}(g)$ and $\mathcal{S}(q) \prec \mathcal{S}(g)$. It follows from this with theorem 10 that the S-pair $(b_{i_{k_0'}}, b_{i_{k_1'}})$ is S-pair with computed G_g-representation, what means that there is an representation described in example 22 :

$$q = 1 \cdot b_{i'} + \sum_l m_l \cdot b_{i_l},$$

satisfying the properties shown after that example: $\mathcal{S}(q) = \mathcal{S}(b_{i'})$, $\forall l \, \mathcal{S}(q) \succ \mathcal{S}(m_l b_{i_l})$ and $i' > k_0'$.

From the other hand we have $m'q = c_0 m_{k_0'} b_{i_{k_0'}} - c_1 m_{k_1'} b_{i_{k_1'}}$, so we get the following representation:

$$m_{k_0'} b_{i_{k_0'}} = c_0^{-1} c_1 m_{k_1'} \cdot b_{i_{k_1'}} + c_0^{-1} m' \cdot b_{i'} + \sum_l c_0^{-1} m' m_l \cdot b_{i_l}.$$

It is $\lessdot$-smaller than $m_{k_0'} b_{i_{k_0'}} = m_{k_0'} \cdot b_{i_{k_0'}}$:

- $m_{k_1'} \cdot b_{i_{k_1'}}$ was already compared to $m_{k_0'} \cdot b_{i_{k_0'}}$
- $m' \cdot b_{i'}$ has the same signature but greater position $i' > i_{k_0'}$
- the last sum contains elements with signatures smaller than $m' \mathcal{S}(b_{i'}) = \mathcal{S}(m_{k_0'} \cdot b_{i_{k_0'}})$.

This gives

Lemma 30. *If a signature-safe representation $mh = \sum_k m_k \cdot b_{i_k}$ with $\mathcal{S}(mh) \prec \mathcal{S}(g)$ satisfies properties 1 and 2 but does not satisfy property 3 then there exists an element $m_{k_0'} \cdot b_{i_{k_0'}}$ having representation $m_{k_0'} b_{i_{k_0'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{k_0'} b_{i_{k_0'}} = m_{k_0'} \cdot b_{i_{k_0'}}$.*

Four previous lemmas together produce a theorem about existence $\lessdot$-smaller representation:

Theorem 31. *A signature-safe representation $mh = \sum_k m_k \cdot b_{i_k}$ with $\mathcal{S}(mh) \prec \mathcal{S}(g)$ either satisfies properties 1-3 or there exists a signature-safe representation $mh = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than $\sum_k m_k \cdot b_{i_k}$.*

In conjunction with representation well-orderness this means that from any given signature-safe representation we can produce a finite $\lessdot$-descending sequence which ends with an signature-safe representation satisfying properties 1-3. This leads to main result:

Theorem 32. *For any labeled polynomial mh, $m \in \mathcal{K} \times T$, $h \in G_g$ with $\mathcal{S}(mh) \prec \mathcal{S}(g)$ there exists a signature-safe G_g-representation $mh = \sum_k m_k \cdot b_{i_k}$ that satisfies properties 1-3.*

This result may be interesting by itself, but for the purposes of proving termination only one corollary is needed:

Corollary 33. *If there exists a signature-safe reductor f for the labeled polynomial f_1 with $\mathcal{S}(f)\frac{\mathrm{HM}(f_1)}{\mathrm{HM}(f)} \prec \mathcal{S}(g)$ then G_g contains a signature-safe reductor for f_1 that is not rejected by F5 and Rewritten criteria.*

To prove the corollary let mf, $m = \frac{\mathrm{HM}(f_1)}{\mathrm{HM}(f)} \in \mathcal{K} \times T$, $f \in G_g$ be a multiplied reductor with $\mathcal{S}(mf) \prec \mathcal{S}(g)$. From the previous theorem we can find representation $mf = \sum_k m_k \cdot b_{i_k}$ that satisfies properties 1-3. Property 3 means that there is no elements with HMs greater than mf so because sum of all elements has HM equal to $\mathrm{HM}(mf)$ there exists an element k_0 that achieves HM equality: $\mathrm{HM}(m_{k_0} \cdot b_{i_{k_0}}) = \mathrm{HM}(mf) = \mathrm{HM}(f_1)$. Since the representation is signature-safe $\mathcal{S}(m_{k_0} \cdot b_{i_{k_0}}) \preccurlyeq \mathcal{S}(mf) \prec \mathcal{S}(f_1)$ so $m_{k_0} b_{i_{k_0}}$ is a signature-reductor for f_1 and properties 1-2 ensure that $m_{k_0} b_{i_{k_0}}$ does not satisfy criteria. The corollary is proved.

6. Finding contradiction with the criteria enabled

Now return to the result of theorem 8 which states for the case of algorithm non-termination existence of a polynomial f such that $\mathrm{HM}(f)|\mathrm{HM}(g)$, $\mathcal{S}(f)|\mathcal{S}(g)$, $\frac{\mathrm{HM}(f)}{\mathcal{S}(f)} > \frac{\mathrm{HM}(g)}{\mathcal{S}(g)}$, where g is the same fixed g as above. Let $t = \frac{\mathrm{HM}(g)}{\mathrm{HM}(f)}$. tf is a signature-safe reductor for g, so $\mathcal{S}(f)t \prec \mathcal{S}(g)$ and the corollary 33 can be applied to find a signature-safe reductor $t_1 f_1$ for g which does not satisfy criteria. Also the reductor f_1 is known to belong to G_g, so during the algorithm execution it was appended to $G \cup Done$ before g. This gives main result for establishing contradiction:

Theorem 34. *If the algorithm does not terminate for some input then after some finite step the set $G \cup Done$ contains a pair of labeled polynomials f_1, g where:*
  -f_1 is added to $G \cup Done$ before g
  - $t_1 f_1$ does not satisfy F5 and Rewritten criteria, where $t_1 = \frac{\mathrm{HM}(g)}{\mathrm{HM}(f_1)}$
  -f_1 is signature-safe reductor for g.

From the other hand existence of such pair of labeled polynomials is not possible: consider the call to TopReduction after which the polynomial g was inserted in *Done*. That call returns polynomial as first part of TopReduction return value, so the value returned by IsReducible is empty set. It means that one of conditions (a) - (d) was not satisfied for all polynomials in $G \cup Done$ including f. This is not possible because:

- (a) is satisfied because f_1 is a reductor for g from last theorem
- (b) and (c) are satisfied because $\frac{\mathrm{HM}(g)}{\mathrm{HM}(f_1)} f_1$ does not satisfy F5 and Rewritten criteria from last theorem
- (d) is satisfied because f_1 is a signature-safe reductor for g from last theorem.

So, the contradiction is proved and the main result too:

Theorem 35. *The original F_5 algorithm as described in* [8] *does terminate for any input.*

7. Conclusions

This paper shows that original F_5 algorithm terminates for any homogeneous input. However, it does not give any limit on number of operations. The simplest proof of the termination of Buchberger algorithm is based on Noetherian property and doesn't give any such limit too. Unfortunately the termination proof given here is quite different in structure compared to the proof of Buchberger algorithm termination, so this proof doesn't show that F_5 is more efficient than Buchberger in any sense. Unlike this the termination of the modified versions of F_5 algorithm in [7, 9, 4] is shown in a way analogous to Buchberger algorithm and there is room for comparison of their efficiency with Buchberger's one.

From the point of view of practical computer algebra computations there is a question about efficiency of the modified versions compared to original F_5. Unfortunately the only strict fact in this area that follows from this paper is that both modified and original versions terminate. The modified versions can spend more time in additional termination checks. But for some cases it is possible that those checks can allow the termination of modified versions before original so the modified version performs smaller number of reductions. So it is possible that for some inputs the original algorithm is faster and for others the modified version. Some experimental timings in Table 1 in [4] shows that both cases are possible in practice but the difference in time is insignificant. So the question about efficiency of original F_5 compared to modified versions is open.

This proof uses three properties of original F_5 that are absent or optional in some F_5-like algorithms: the homogeneity of input polynomials, the presence of Rewritten criterion and the equality of monomial order $<$ and signature order $\prec$. The possibility of extending the termination proof to the modified algorithms without these properties is open question. There is an unproved idea that the proof can be modified to remove reliance on the first two properties but not on the third property of orders equality because it is key point of coming to a contradiction form the result of theorem 8.

References

[1] Alberto Arri and John Perry. The F5 criterion revised
[2] Alexey Zobnin. Generalization of the F5 algorithm for calculating Gröbner bases for polynomial ideals
[3] Christian Eder, John Perry. F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases
[4] Christian Eder, Justin Gash, and John Perry. Modifying Faugère's F5 algorithm to ensure termination
[5] Franz Baader and Tobias Nipkow. Term Rewriting and All That
[6] Gwénolé Ars, Amir Hashemi. Extended F 5 criteria.
[7] Gwénolé Ars. Applications des bases de Gröbner à la cryptographie
[8] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5)
[9] Justin M. Gash. On efficient computation of Gröbner bases
[10] Till Stagers. Faugère's F5 Algorithm Revisited