TERMINATION OF ORIGINAL F5

VASILY GALKIN

Abstract. The original F5 algorithm introduced by Faugère is formulated for any homogeneous polynomial set input. The correctness of output is shown for any input that terminates the algorithm, but the termination itself is proved only for the case of input being regular polynomial sequence. This article shows that algorithm correctly terminates for any homogeneous input without any reference to regularity. 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 F5. 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 F5 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 [Faugère, 2002] and detailed investigations in [Stegers, 2006] 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. This approach is perfectly strict but the obtained result is termination proof of a modified version of F5 algorithm which contain additional checks and therefore can be more complex for implementation and possibly slower for some input cases. Examples are [Eder et al., 2010, Ars, 2005, Gash, 2008].

The another approach is termination proof of custom F5-based algorithms followed by attempt to reformulate original F5 in the terms of this custom algorithm. The main problem of this approach arise during reformulation: attempts to describe F5 in another terms may inadvertently introduce some changes in behavior which are hard to discover but require additional proofs to show equivalence. For example [Pan et al., 2012] proofs the termination of the F5GEN algorithm which differs from original F5 by absence of criteria check during reductor selection. The [Huang, 2010] gives the proof of TRB-F5 algorithm termination which has two main differences realized by the author with great help of discussions with John Perry. The first is another rule building scheme which eventually lead to the ordering by signature the rules in Rule array during TRB-F5 execution. The second is the absence of applying in TRB-F5 the normal form φ before reduction, which leads to an effect opposite to the difference with F5GEN: criteria checks are applied for elements with greater signature index which are used in original F5 in the normal from operator as reductors without the checks. The author thinks that these algorithms may be changed to exactly reproduce the original F5 behavior and apply the termination proofs can be applied to such changed versions. But the approach with algorithms equivalent to F5 has a drawback: it makes harder to understand how the theorems used in termination proof can be expressed in terms of original F5 behavior.

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 [Eder and Perry, 2009] 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 [Faugère, 2002], 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.

Proof. Let's suppose that there is an input $\{f_1, \ldots, f_m\}$ over $\mathcal{K}[x_1, \ldots, x_n]$ for which original F5 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_j the value of d on j-th cycle iteration and extend it to $d_0 = -1$. The simple property of d_j is it's non-strict growth: $d_j \geqslant d_{j-1}$. It holds because on the $j-1$-th iteration all polynomials in R_d have degree d_{j-1} and therefore all new generated critical pairs have degree at least d_{j-1}. Now suppose that j is number of some fixed 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 in the cycle over R_d. 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 next iteration of 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 [Faugère, 2002] is a bit unclear at this point, but the source code of procedure FindReductor attached to [Stegers, 2006] 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:** *ToDo* **finiteness.** This section aims to prove the following

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

Proof. The calls to AlgorithmF5 corresponding to polynomials $f_2, \ldots, f_m$ are known to terminate, so we will study the only left call of AlgorithmF5 corresponding to processing of input sequence item f_1. Firstly we need to get some facts about polynomials in *ToDo* and *Rules* sets inside j-th iteration of that call to AlgorithmF5. All the critical pairs created initially by CritPair inside the AlgorithmF5 have greater S-pair part with signature index 1. All other critical pairs are generated with signature index corresponding to *ToDo* elements moved to *Done* set. All elements of *ToDo* are generated either from critical pairs or in the TopReduction procedure. The polynomials generated by TopReduction has signatures greater than the signature of polynomial the function tries to top-reduce which is *ToDo* element. So, there is no place in algorithm flow where a polynomial or critical pair with signature index different from 1 can be generated in the AlgorithmF5 call. From the other hand all polynomials inside *ToDo* has the same total degree d_j. Together with index equality this shows that the total degree of signature monomials is equal to the $d_j - \deg(f_1)$ for all *ToDo* elements.

Every addition of polynomial in the *Rules* set correspond to the addition of *ToDo* element. So, The elements added to *Rules* in j-th iteration have total degree equal to d_j. Combining this with non-strict d_j growing we get that at j-th iteration all elements of *Rules* with signature index 1 have total degree $\leqslant d_j$ and total degree of signature $\leqslant d_j - \deg(f_1)$. In addition this gives the

fact about order of *Rules* elements with signature index 1: their total degrees are non-strictly increasing.

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*.

The algorithm performs only signature-safe reductions: the TopReduction function performs reduction by non-rejected reductor if it is signature-safe and adds new element to $ToDo$ otherwise. The elements of $ToDo$ are processed in signature increasing order, so no elements of $G \cup Done$ has signature greater than signature of polynomial r_k being reduced in TopReduction. If the reductor r_m has $\deg(r_m) = \deg(r_k)$ we have $t = 1$ and $\mathcal{S}(r_k) \succ \mathcal{S}(r_m)$ which ensures the signature-safety of such reduction because the case $\mathcal{S}(r_k) = \mathcal{S}(r_m)$ would have been rejected by (d) check in IsReducible. Thus we get that all additions in $ToDo$ in TopReduction correspond to reductions that are not signature-safe and have $\deg(r_m) < \deg(r_k)$. The signature of polynomial added such way is $t \cdot \mathcal{S}(r_m)$ and the fact that r_m was not rejected by IsRewritten? check in IsReducible ensures that no polynomial with signature $t \cdot \mathcal{S}(r_m)$ were generated yet because such polynomial would have rule corresponding to it in *Rules* with greater total degree than rule corresponding to r_m and r_m would be rejected in IsReducible.

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 initially generated by Spol and then extended by new elements during TopReduction execution. Spol generates finite number of elements because it terminates and the TopReduction adds elements with distinct signatures having index 1 so their number is limited by the count of different signatures of total degree $d_j - \deg(f_1)$, so only finite number of elements is added in $ToDo$. We will show that all types of steps that perform Reduction can be performed only finitely number of times:
  - the step when IsReducible returns empty set correspond to transferring $ToDo$ element in *Done* and the number of such steps is limited by number of elements added in $ToDo$
  - the step when IsReducible returns reductor which is not signature-safe correspond to adding new element in $ToDo$ and the number of such steps is limited by the number of possible additions.
  - the step when IsReducible returns reductor which is signature-safe correspond to reduction of some $ToDo$ element. This can be done only finite number of times because there are finitely many polynomials added in $ToDo$ and no polynomial can be top-reduced infinite number of times because its HM is $\prec$-decreasing during reduction and monomials are well-ordered with $\prec$.

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

This gives the result about algorithm behavior for the non-terminated case:

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

Proof. Follows from combination of claims 1 and 2. □

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 algorithm generates an infinite sequence of labeled polynomials $\{r_1, r_2, \ldots, r_m, \ldots, r_l, \ldots\}$ 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 already present in the list. We will write l^* and l_* for the indexes of the polynomials used to generate l-th element and $\overline{u_l}$, $\underline{u_l}$ for monomials they are multiplied. Note that l^* correspond to the part with greater signature: $poly(r_l) = \overline{u_l} poly(r_{l^*}) - \underline{u_l} poly(r_{l_*})$ and $\mathcal{S}(r_l) = \overline{u_l}\mathcal{S}(r_{l^*}) \succ \underline{u_l}\mathcal{S}(r_{l_*})$. 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} is an S-polynomial generated by $r_{k_{n-1}} = r_{k_n^*}$ and some other polynomial corresponding to smaller by signature S-pair part, so $\mathcal{S}(r_{k_n}) = \overline{u_{k_n}}\mathcal{S}(r_{k_{n-1}})$ and

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

Definition 5. 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_{n^*} which was used to generate r_n as $Spoly(r_{n^*}, r_{n_*})$. The resulting S-pair-chain has the form $\{r_q, \ldots, r_{l^{**}}, r_{l^*}, r_l\}$ 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.

Theorem 6. *Every labeled S-polynomial can participate as the first element only in finite number of S-pair-chains of length 2.*

Proof. 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 is checking that itself, 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 with signature $\mathcal{S}(r_L) = s$ and an ordered by generating time subset $\{r_{l_1}, \ldots, r_{l_i}, \ldots\}$ of labeled polynomials with signatures satisfying $\mathcal{S}(r_{l_i}) = v_i\mathcal{S}(r_L)$. From the signature divisibility point of view all of the possibly infinite number of pairs $\{r_L, r_{l_i}\}$ can be S-pair-chains of length 2. But the ideal (v_i) in T is finitely generated by Dickson's lemma, so after some step i_0 we have $\forall i > i_0\, \exists j \leqslant i_0$ such that $v_j | v_i$. So $\forall i > i_0$ the sequence

$\{r_L, r_{l_i}\}$ is not S-pair-chain because $\mathcal{S}(r_L) \cdot u_i$ is rewritten by $\mathcal{S}(r_{l_j}) \cdot \frac{v_i}{v_j}$ and no more than i_0 S-pair-chains of length 2 with first element r_L exist. □

Definition 7. The finite set of ends of 2-length S-pair-chains starting with r_L will be called *S-pair-descendants* of r_L.

Theorem 8. *If the algorithm does not terminate for some input then there exists infinite S-pair-chain* $\{h_i\}$.

Proof. Some caution is required while dealing with infinities, so we give the following definition.

Definition 9. 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 $\{r_1, \ldots, r_{l^{**}}, r_{l^*}, r_l\}$.

Now assume that some labeled polynomial r_l is known to be a chain generator. Then one of the finite number of S-pair-descendants of r_l 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_l 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_l. So, if labeled polynomial r_l is chain-generator, we can select another chain generator from it's S-pair-descendants. In a such way we can find infinite S-pair-chain starting with r_1 and consisting of chain generators which proves the theorem. □

For the next theorem we need to introduce monomial quotients order by transitively extending the monomial ordering: $\frac{m_1}{m_2} >_q \frac{m_3}{m_4} \Leftrightarrow m_1 m_4 > m_3 m_2$.

Theorem 10. *If the algorithm does not terminate for some input then after some finite step the set* G *contains a pair of labeled polynomials* f', f *with* f *generated after* f' *that satisfies the following 3 properties:*

$$\mathrm{HM}(f') | \mathrm{HM}(f),$$

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

$$\mathcal{S}(f') | \mathcal{S}(f).$$

Proof. For working with S-pair-chains 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 keeps the same $poly(h_n)$ value after it was used for some 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:

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

From the first inequality in 3.2 we can get $\frac{\mathrm{HM}(h_n)}{\mathcal{S}(h_n)} >_q \frac{\mathrm{HM}(h_{n+1})}{\mathcal{S}(h_{n+1})}$, so in the S-pair-chain the quotients $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)}$ are strictly descending according to quotients 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 $\frac{x}{x} >_q \frac{x}{x^2} >_q \cdots >_q \frac{x}{x^n} >_q \cdots$ 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 [Arri and Perry, 2010]: 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 paragraph 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 $\frac{\mathrm{HM}(h_i)}{\mathcal{S}(h_i)} >_q \frac{\mathrm{HM}(h_{i+1})}{\mathcal{S}(h_{i+1})} >_q \cdots >_q \frac{\mathrm{HM}(h_j)}{\mathcal{S}(h_j)}$, so we take $f' = h_i$ and $f = h_j$. □

The last property about signature division from the theorem claim is the consequence of dealing with S-pair-chains and is not used in the following. But the first to properties are used to construct a signature-safe reductor.

Fact 11. *If no polynomials are rejected by criteria checks (b) and (c) inside IsReducible the algorithm does terminate.*

Proof. The above proof of theorem10 does not rely on any correspondence between orderings on signatures and terms. But the original algorithm F5 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 for polynomials from theorem 10 claim:

$$\mathcal{S}(g) \succ t \cdot \mathcal{S}(f), \text{ where } t = \frac{\mathrm{HM}(g)}{\mathrm{HM}(f)} \in T.$$

The last inequality with HM's division property from theorems result shows that tf can be used as a reductor 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 checks (b) and (c) 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. □

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 reductor for g that is not rejected and anyway lead to contradiction and the following parts of paper aim to prove it.

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

In this and following sections g is treated as some fixed labeled polynomial with signature index 1 added to *Done* in some algorithm iteration. 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 can be processed in a different ways inside the algorithm but the main fact we need to know about their processing is encapsulated in the following properties which correspond to the properties used in Theorem 21 in [Eder and Perry, 2009] but are taken during arbitrary algorithm iteration without requirements of termination.

Theorem 12. *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.*

Proof. 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_{l^*} and r_{l_*} such that $\mathrm{HM}(r_{l^*})|\mathrm{HM}(r_{l_*})$ so that S-polynomial has a form $\overline{u_l} \cdot poly(r_{l^*}) - 1 \cdot poly(r_{l_*})$. Such situation is possible for two G_g elements if reduction of r_{l_*} by r_{l^*} was forbidden by signature comparison in TopReduction or by checks in IsReducible. So for this case the path of S-pair "processing" is the following:

- The S-pair part $\overline{u_l} \cdot r_{l^*}$ is checked in IsReducible. (a) is satisfied because $\mathrm{HM}(r_{l^*})|\mathrm{HM}(r_{l_*})$. It can be rejected by one of the other checks:
  - Rejection by check (b) correspond to the normal form check φ for $\overline{u_l} \cdot r_{l^*}$
  - Rejection by check (c) correspond to the Rewritten? check for $\overline{u_l} \cdot r_{l^*}$
  - Rejection by check (d) means that either $\overline{u_l} \cdot r_{l^*}$ or $1 \cdot r_{l_*}$ can be rewritten by other, so if S-pair was not rejected by check (c) this type of rejection means that S-pair part $1 \cdot r_{i_1}$ 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_{l_*} by r_{l^*} 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_l such that $\deg(r_l) = \deg(g)$, $\mathcal{S}(r_l) \prec \mathcal{S}(g)$. 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_l is reduced before g and during r_l reduction just before putting r_l to $Done$ a call to IsReducible starts processing for all such S-pairs. □

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 13. 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 [Faugère, 2002].

Definition 14. 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 $s_1 sr_i$ satisfies it too.

5. Representations

5.1. **Definition.** The idea of representations comes from proof of Theorem 21 in [Eder and Perry, 2009]. Representations are used to describe 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

$$p = \sum_k m_k \cdot b_{i_k},\ b_{i_k} \in G_g \tag{5.1}$$

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

Definition 15. 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 16. The G_g-representation of p is called *signature-safe* if $\forall k\, \mathcal{S}(m_k b_{i_k}) \preccurlyeq \mathcal{S}(p)$.

5.2. **Examples.**

Example 17. 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 18. 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

$$b_j h = \sum_k m_k \cdot b_j \tag{5.2}$$

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 19. 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 20. 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 21. 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 22. This particular case of G_g-representation where b_{i_k} are limited to input polynomials will be called *input-representation*.

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

The following claim makes important connection between signatures and input-representations:

Claim 23. An admissible 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.

Proof. The claimed fact follows from the admissible polynomial definition in [Faugère, 2002] referring to function v which correspond to summing representation elements. □

The theorem 1 of [Faugère, 2002] states that all polynomials in the algorithm are admissible, do the above claim will be applied to all appeared polynomials.

Example 24. The last example comes from S-pairs with a computed G_g-representation. S-polynomial of b_{l^*} and b_{l_*} from G_g is $p = \overline{u_l} poly(b_{l^*}) - \underline{u_l} poly(b_{l_*})$. 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 l^* and 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_{l^*} and b_{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 25. To order G_g-representations we start from *representation elements ordering* $\succcurlyeq_1$: $c_i t_i \cdot b_i \succcurlyeq_1 c_j t_j \cdot b_j$ 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 $\preccurlyeq_1$ order because they have distinct (t_k, b_k) by definition. Below are given some examples of $\preccurlyeq_1$ 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 $\mathcal{S}(b_1) = \mathbf{F}_1, \mathcal{S}(b_2) = \mathbf{F}_2, \mathcal{S}(b_3) = x\mathbf{F}_1$.

- $y \cdot b_1 \succcurlyeq_1 100y \cdot b_2$ because signature of left side is $\succ$
- $x \cdot b_1 \succcurlyeq_1 y \cdot b_1$ 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
- $y^2 \cdot b_1 \preccurlyeq_1 y \cdot b_3$ because signature of left side is $\prec$
- $x^2 \cdot b_1 \succcurlyeq_1 x \cdot b_3$ 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 $\succcurlyeq_1$-decreasing order. This form can be used for equality testing because if two representations are equal then they have exactly equal ordered forms.

Definition 26. With ordered forms the *G_g-representations ordering* can be introduced: the representation $\sum_k m'_k \cdot b_{i'_k}$ is $\preccurlyeq$-smaller than $\sum_k m_k \cdot b_{i_k}$ iff the ordered form of the first representation is smaller than second's according to lexicographical extension of $\preccurlyeq_1$ ordering on elements. For the corner case of the one ordered form being beginning of the other the shorter form is $\preccurlyeq$-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:

- $x^2 \cdot b_1 + xy \cdot b_1 + y^2 b_1 \succcurlyeq x^2 \cdot b_1 + 100y^2 \cdot b_1$ because $xy \cdot b_1 \succcurlyeq y^2 \cdot b_1$
- $x^2 \cdot b_1 + 100y^2 \cdot b_1 \succcurlyeq x^2 \cdot b_1$ because the right ordered form is beginning of the left
- $x^2 \cdot b_1 \succcurlyeq xy \cdot b_1 + y^2 \cdot b_1 + x^2 \cdot b_2$ because $x^2 \cdot b_1 \succcurlyeq xy \cdot b_1$
- $xy \cdot b_1 + y^2 \cdot b_1 + x^2 \cdot b_2 \succcurlyeq y \cdot b_3 + y^2 \cdot b_1 + x^2 \cdot b_2$ because $xy \cdot b_1 \succcurlyeq y \cdot b_3$
- $y \cdot b_3 + y^2 \cdot b_1 + x^2 \cdot b_2$ 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 27. *If two representations of p has a relation $\sum_k m'_k \cdot b_{i'_k} \preccurlyeq \sum_k m_k \cdot b_{i_k}$ and the second one is signature-safe representation then the first one is signature-safe too.*

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

The key fact allowing to take $\lessdot$-minimal element is well-orderness:

Theorem 28. *The representations are well-ordered with $\lessdot$ ordering.*

Proof. 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 $\gtrdot_1$-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 $\lessdot_1$.

The straightforward proof for $\lessdot$-well-orderness of representations following from $\lessdot_1$-well-orderness of elements is not very complex but to skip its strict details the theorem 2.5.5 of [Baader and Nipkow, 1998] will be referenced. 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. □

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 [Eder and Perry, 2009] and is based on a fact, that if a some signature-safe representation of mh contains an element $m_K \cdot b_{i_K}$ not having one of the properties then a $\lessdot$-smaller representation can be constructed. Though the exact construction differ for three cases but the replacement scheme is the same:

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

Construction of such representation for $m_{K'} \cdot b_{i_{K'}}$ allows application of the following lemma:

Lemma 29. *If an element $m_{K'} \cdot b_{i_{K'}}$ of signature-safe representation $mh = \sum_k m_k \cdot b_{i_k}$ has an representation $m_{K'} b_{i_{K'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{K'} b_{i_{K'}} = m_{K'} \cdot b_{i_{K'}}$ then mh has a signature-safe representation $\lessdot$-smaller than $mh = \sum_k m_k \cdot b_{i_k}$.*

Proof. We replace $m_{K'} \cdot b_{i_{K'}}$ in $mh = \sum_k m_k \cdot b_{i_k}$ by $\sum_l m_l \cdot b_{i_l}$ and combine coefficients near elements with both monomial and polynomial equal, so a modified representation for mh appears. Is is $\lessdot$-smaller than $mh = \sum_k m_k \cdot b_{i_k}$ because all elements $\gtrdot_1$-greater than $m_{K'} \cdot b_{i_{K'}}$ are identical in both representations if they present but the element $m_{K'} \cdot b_{i_{K'}}$ is contained in original representation but not in the modified. And all other elements in representations are $\lessdot_1$-smaller than $m_{K'} \cdot b_{i_{K'}}$ 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 $mh = m_{K'} \cdot b_{i_{K'}} + \sum_l (-m_l) \cdot b_{i_l}$ what leads to modified representation $mh = 0$ with zero elements which is $\lessdot$-smaller than any non-empty representation. □

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.

Lemma 30. *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 $m_{K'} \cdot b_{i_{K'}}$ having G_g-representation $m_{K'} b_{i_{K'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{K'} b_{i_{K'}} = m_{K'} \cdot b_{i_{K'}}$.*

An element not having the first property does satisfy the F5 criterion and the idea is to use that $m_K \mathcal{S}(b_{i_K})$ is not the minimal signature of $m_K b_{i_K}$ like in Theorem 20 of [Eder and Perry, 2009]. $K' = K$ is taken for this case.

Proof. Consider input-representation of $m_K b_{i_K}$ with signature of $\succcurlyeq_1$-maximal element equal to $m_K \mathcal{S}(b_{i_K}) = s_0 \mathbf{F}_{j_0}$:

$$m_K b_{i_K} = c_0 s_0 \cdot f_{j_0} + \sum_l m_l \cdot f_{i_l}. \tag{5.3}$$

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 $m_K b_{i_K}$, 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:

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

This representation is $\lessdot$-smaller than $m_K \cdot b_{i_K}$ 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: $s_1 m_{1i} \mathbf{F}_{j_0} \prec s_1 m_{11} \mathbf{F}_{j_0} = s_0 \mathbf{F}_{j_0}$. □

Lemma 31. *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 $m_{K'} \cdot b_{i_{K'}}$ having G_g-representation $m_{K'} b_{i_{K'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{K'} b_{i_{K'}} = m_{K'} \cdot b_{i_{K'}}$.*

For the elements not satisfying case 2 the $\lessdot$-smaller representation is created in a way used in Proposition 17 of [Eder and Perry, 2009]. $K' = K$ is taken for this case too.

Proof. Assume that $\mathcal{S}(m_K b_{i_K}) = s_0 \mathbf{F}_{j_0}$ 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 $i' > i_K$ and the existence of $s' \in T$ such that $s' \mathcal{S}(b_{i'}) = s_0 \mathbf{F}_{j_0}$. So, for the $m_K b_{i_K}$ 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:

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

It is $\lessdot$-smaller than $m_K b_{i_K} = m_K \cdot b_{i_K}$ because elements of both sums has signatures smaller than $s_0 \mathbf{F}_{j_0}$, and for the first element $\mathcal{S}(c'^{-1} c_0 s' \cdot b_{i'}) = \mathcal{S}(m_K \cdot b_{i_K}) = s_0 \mathbf{F}_{j_0}$ but $i' > i_K$, so applying the $\lessdot_1$-comparison rule for equal signatures and different list positions we get that element $c'^{-1} c_0 s' \cdot b_{i'}$ is $\lessdot_1$-smaller than $m_K \cdot b_{i_K}$ too. □

Lemma 32. *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 $m_{K'} \cdot b_{i_{K'}}$ having representation $m_{K'} b_{i_{K'}} = \sum_l m_l \cdot b_{i_l}$ which is $\lessdot$-smaller than representation $m_{K'} b_{i_{K'}} = m_{K'} \cdot b_{i_{K'}}$.*

Proof. There exists at least one element $m_K \cdot b_{i_K}$ 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' to be the index of the $\geqslant_1$-greatest representation element in $H_{\max}$. We have $\mathrm{HM}(m_{K'} b_{i_{K'}}) = m_{\max} \geqslant \mathrm{HM}(m_K b_{i_K}) > \mathrm{HM}(mh)$, so the HM of sum of all elements except K' is equal to $\mathrm{HM}(mh - m_{K'} b_{i_{K'}}) = \mathrm{HM}(m_{K'} b_{i_{K'}}) = m_{\max}$, so there is another element K'' having $\mathrm{HM}(m_{K''} b_{i_{K''}}) = m_{\max}$. So, $m_{K''} \cdot b_{i_{K''}} \in H_{\max}$ and $m_{K''} \cdot b_{i_{K''}} \lessdot_1 m_{K'} \cdot b_{i_{K'}}$ because of $m_{K'} \cdot b_{i_{K'}}$ $\geqslant_1$-maximality in $H_{\max}$.

The $\mathrm{HM}(m_{K''} b_{i_{K''}}) = \mathrm{HM}(m_{K'} b_{i_{K'}})$ means that a critical pair of $b_{i_{K'}}$ and $b_{i_{K''}}$ has the form $[m'^{-1} m_{\max}, m'^{-1} m_{K'}, b_{i_{K'}}, m'^{-1} m_{K''}, b_{i_{K''}}]$ where $m' = \gcd(m_{K'}, m_{K''})$. Let q be corresponding S-polynomial. Then $m' \mathcal{S}(q) \preccurlyeq \mathcal{S}(mh) \prec \mathcal{S}(g)$ because $m' \mathcal{S}(q) = \mathcal{S}(m_{K'} b_{i_{K'}})$ and the representation is signature-safe. The S-polynomial parts $m'^{-1} m_{K'} b_{i_{K'}}$ and $m'^{-1} m_{K''} b_{i_{K''}}$ doesn't satisfy F5 and Rewritten criteria because their forms multiplied by m' are $m_{K'} b_{i_{K'}}$ and $m_{K''} b_{i_{K''}}$ – 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 12 that the S-pair $(b_{i_{K'}}, b_{i_{K''}})$ is S-pair with computed G_g-representation, what means that there is an representation described in example 24 :

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

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 $i' > K'$.

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

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

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

Theorem 33. *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 $\sum_k m_k \cdot b_{i_k}$.*

Proof. This theorem quickly follows from four previous lemmas together □

This leads to main result:

Theorem 34. *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.*

Proof. Start with representation $mh = m \cdot h$ and begin replacing it by $\ll$-smaller representation from theorem 33 until the representation satisfying properties 1-3 appears. The finiteness of the process is guaranteed by $\ll$-well-orderness. □

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

Corollary 35. *Consider an arbitrary polynomial f without any restrictions on its signature. If there exists a signature-safe reductor $f' \in G_g$ for f with $\mathcal{S}(f')\frac{\mathrm{HM}(f)}{\mathrm{HM}(f')} \prec \mathcal{S}(g)$ then G_g contains a signature-safe reductor for f that is not rejected by F5 and Rewritten criteria.*

Proof. Let mf', $m = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f')} \in \mathcal{K} \times T$, $f' \in G_g$ 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 that achieves HM equality: $\mathrm{HM}(m_K \cdot b_{i_K}) = \mathrm{HM}(mf') = \mathrm{HM}(f'_1)$. Since the representation is signature-safe $\mathcal{S}(m_K \cdot b_{i_K}) \preccurlyeq \mathcal{S}(mf') \prec \mathcal{S}(f)$ so $m_K b_{i_K}$ is a signature-safe reductor for f and properties 1-2 ensure that $m_K b_{i_K}$ does not satisfy criteria. □

6. Finding contradiction with the criteria enabled

Now return to the result of theorem 10 which states for the case of algorithm non-termination existence of a polynomials $f', f \in G$ such that $\mathrm{HM}(f')|\mathrm{HM}(f)$, $\frac{\mathrm{HM}(f')}{\mathcal{S}(f')} > \frac{\mathrm{HM}(f)}{\mathcal{S}(f)}$. Using this result and last corollary we construct two polynomials leading to contradiction for the case of algorithm non-termination.

Theorem 36. *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, f where:*

- *f'_1 is added to $G \cup Done$ before f*
- *$t_1 f'_1$ does not satisfy F5 and Rewritten criteria, where $t_1 = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f'_1)}$*
- *f'_1 is signature-safe reductor for f.*

Proof. Let f', f be polynomials from the theorem 10 an define $t = \frac{\mathrm{HM}(f)}{\mathrm{HM}(f')}$. We have $f \in G$ so the above theory about representations can be applied to the fixed value of g equal to f and we can speak about G_f set and G_f-representations. Because tf' is a signature-safe reductor for f we have $\mathcal{S}(f')t \prec \mathcal{S}(f)$ and the corollary 35 can be applied to find a signature-safe reductor $t_1 f'_1$ for f which does not satisfy criteria. Also it is known to belong to G_f, so during the algorithm execution f'_1 was appended to $G \cup Done$ before f. □

Theorem 37. *The original F5 algorithm as described in* [Faugère, 2002] *does terminate for any input.*

Proof. We are going o show that the existence of polynomials f'_1, f from the theorem 36 leads to contradiction. Consider the call to TopReduction after which the polynomial f was inserted in *Done*. That call returns polynomial f 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'_1. This is not possible because:

- (a) is satisfied because f'_1 is a reductor for f from the theorem 36
- (b) and (c) are satisfied because $\frac{\mathrm{HM}(f)}{\mathrm{HM}(f'_1)} f'_1$ does not satisfy F5 and Rewritten criteria from the theorem 36
- (d) is satisfied because f'_1 is a signature-safe reductor for f from the theorem 36.

□

7. Conclusions

This paper shows that original F5 algorithm terminates for any homogeneous input without introducing intermediate algorithms. 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 F5 is more efficient than Buchberger in any sense. Unlike this the termination of the modified versions of F5 algorithm in [Eder et al., 2010, Ars, 2005, Gash, 2008] 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 F5. 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 [Eder et al., 2010] shows that both cases are possible in practice but the difference in time is insignificant. So the question about efficiency of original F5 compared to modified versions is open.

This proof uses three properties of original F5 that are absent or optional in some F5-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 10.

The author would like to thank Christian Eder, Jean-Charles Faugère, Amir Hashemi, John Perry, Till Stagers and Alexey Zobnin for inspiring me on investigations in this area by their papers and comments. Thanks!

References

[Arri and Perry, 2010] Arri, A. and Perry, J. (2010). The F5 Criterion revised. *ArXiv e-prints*.

[Ars, 2005] Ars, G. (2005). Applications des bases de Gröbner à la cryptographie. Technical report.

[Baader and Nipkow, 1998] Baader, F. and Nipkow, T. (1998). *Term Rewriting and All That*. Cambridge University Press, United Kingdom.

[Eder et al., 2010] Eder, C., Gash, J., and Perry, J. (2010). Modifying Faugère's F5 Algorithm to ensure termination. *ACM Communications in Computer Algebra, Issue 176, vol. 45, no. 2 (June 2011), pgs. 70-89*.

[Eder and Perry, 2009] Eder, C. and Perry, J. (2009). F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases. *ArXiv e-prints*.

[Faugère, 2002] Faugère, J. C. (2002). A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, ISSAC '02, pages 75–83, New York, NY, USA. ACM.

[Gash, 2008] Gash, J. M. (2008). *On efficient computation of grobner bases*. PhD thesis, Indiana University, Indianapolis, IN, USA.

[Huang, 2010] Huang, L. (2010). A new conception for computing gröbner basis and its applications. *ArXiv e-prints*.

[Pan et al., 2012] Pan, S., Hu, Y., and Wang, B. (2012). The Termination of Algorithms for Computing Gröbner Bases. *ArXiv e-prints*.

[Stegers, 2006] Stegers, T. (2006). Faugere's F5 Algorithm Revisited. *IACR Cryptology ePrint Archive*, 2006:404.

Moscow State University
E-mail address: galkin-vv@ya.ru