Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Complexity Analysis of the Bivariate Buchberger Algorithm in *Theorema*

Alexander Maletzky

DK-Report No. 2014-10 09 2014

A–4040 LINZ, ALTENBERGERSTRASSE 69, AUSTRIA

Supported by
Austrian Science Fund (FWF) Upper Austria

DK sponsors:

- **Johannes Kepler University Linz** (JKU)
- **Austrian Science Fund** (FWF)
- **Upper Austria**

Complexity Analysis of the Bivariate Buchberger Algorithm in *Theorema*[*]

Alexander Maletzky

DK Computational Mathematics / RISC

Johannes Kepler University Linz

alexander.maletzky@dk-compmath.jku.at

October 2014

Abstract

In this report we describe the formalization and formal, semi-automated verification of a part of Gröbner bases theory, namely the complexity analysis of Buchberger's algorithm in the bivariate case, in the computer system *Theorema*. We not only explain the individual steps we carried out to systematically explore the theory, but also the design principles we followed for creating a new *Theorema* special prover, as well as the improvements (regarding generality and simplicity) we achieved compared to the original pencil-and-paper elaboration of the theory by Buchberger. Up to our knowledge, there does not exist any other formal treatment of exactly this theory in any other computer system.

1 Introduction

This report presents a major case study in how mathematical theory exploration can be carried out in the *Theorema* system: The theory that is explored is the complexity analysis of Buchberger's algorithm with chain criterion in the bivariate case, as investigated more than 30 years ago by Buchberger in [12, 3, 4]. Hence, neither was the underlying theory developed only recently, nor were the main theorems proved only with the help of the computer system – All those ingredients have already been there before. Rather, the achievement of our research is

*The research was funded by the Austrian Science Fund (FWF): grant no. W1214-N15, project DK1

the formal treatment of the theory, including both *formalization* and *formal verification* by means of *semi-automated proving*, such that eventually we obtained a "polished", computer-verified version of the original pencil-and-paper elaboration. Moreover, the close investigation of the hand-crafted proofs that was necessary for achieving the formal verification led to some theoretical improvements, too (both generalizations and simplifications).

Although there are quite some examples of formalizations of Gröbner bases in computer systems, like the one of Coquand and Persson [15], Thery [28] and Jorge [19] in Coq [14], of Medina-Bulo et al. [24, 25] in ACL2 [20], and of Schwarzweller [27] in Mizar [29], we are not aware of any formalizations that target precisely the fragment of Gröbner bases we considered, namely the complexity analysis of Buchberger's algorithm in the bivariate case. This is true in particular also for all existing formalizations of Gröbner bases in *Theorema* (c. f. for instance [6]).

Theorema [11, 9] is a system for mathematical theory exploration, which was initiated by Bruno Buchberger in the mid-nineties and is now developed in his *Theorema* group at RISC. It uses the computer algebra system *Mathematica* [33] as software frame, in the sense that it is basically a *Mathematica* package. Its user interface is currently re-designed and -implemented (*Theorema* Version 2.0), and whenever we refer to *Theorema* we actually mean *Theorema* 2, because the formalization was already carried out in the new system. One of the main paradigms underlying *Theorema* is the idea of supporting "working mathematicians" in all aspects of their everyday-work, ranging from developing theories, over doing computations, until even writing papers – *Theorema* notebooks themselves look much more like nicely-formatted journal articles than plain source code.

The research described in this report covers part of the author's PhD project, which is about formalizing the foundations of Gröbner bases theory (Main Theorem on S-polynomials, correctness of Buchberger's algorithm, …). The largest part of this project is still ongoing work, and we are convinced that it will benefit from the successful treatment of the complexity analysis in many respects (see also Section 6 for more information). Furthermore, this work was already presented at [21].

The report is organized as follows: Section 2 gives an overview of the underlying theory, i. e. it presents the algorithm this report is all about, introduces some basic notions, and states the main theorems. Also, the aforementioned theoretical improvements of our formalization are explained there in detail. Section 3 presents the individual steps that were followed in our computer-supported theory exploration, and how the resulting formalization is organized. Section 4 provides a detailed description of the new *Theorema* special prover that was created for verifying the theory, and Section 5 describes the "path" of lemmas and theorems in the formalization that eventually leads to the main theorems of the complexity analysis (this path slightly deviates from the one in the original elaboration).

2 Underlying Theory

The theoretical foundations underlying the formalization were investigated by Bruno Buchberger around 1980 in [2, 12, 3, 4]. Hence, it has to be pointed out that this report – and, in particular, this section – does not present essentially new results obtained only recently, but rather summarizes known results the formalization in *Theorema* is based upon. Still, it also must be mentioned that indeed some minor improvments (i. e. generalizations and simplifications) compared to the original elaboration could be achieved which will be made explicit in the upcoming paragraphs. Therefore, this section might be regarded a summary of the original papers by Buchberger.

2.1 The Algorithm

In order for this report to be self-contained, we state here the algorithm whose complexity we are interested in: Algorithm 1. As usual in papers about Gröbner bases, we fix now some arbitrary admissible term ordering $\preceq$ and let $\mathrm{lt}(p)$ and $\mathrm{lcm}(\sigma, \tau)$ denote the leading term of polynomial p (w. r. t. $\preceq$) and the least common multiple of terms σ and τ, respectively.

chainCrit is the so-called *chain criterion* introduced in [2], formally defined as

$$\textsc{chainCrit}(p, q, G) :\Leftrightarrow \neg\exists_{g \in G} \bigwedge \begin{cases} \mathrm{lt}(g) | r \\ \deg(\mathrm{lcm}(\mathrm{lt}(p), \mathrm{lt}(g))) < \deg(r) \\ \deg(\mathrm{lcm}(\mathrm{lt}(q), \mathrm{lt}(g))) < \deg(r) \end{cases} \tag{2.1}$$

for all polynomials p and q and sets of polynomials G, where $r := \mathrm{lcm}(\mathrm{lt}(p), \mathrm{lt}(q))$.

Algorithm 1 is *Buchberger's algorithm with chain criterion*, invented by Buchberger in [1, 2], which computes Gröbner bases of ideals generated by finite sets of polynomials w. r. t. admissible term orderings. Although the algorithm can be applied on sets of polynomials in arbitrarily many indeterminates, all the complexity results stated here only hold in the bivariate case (which, of course, is also true for the formalization in *Theorema*, see [22]. There, however, some intermediate results could even be proved for arbitrarily many indeterminates). Otherwise, it is well-known that its complexitiy is asymptotically double exponential in the number of indeterminates [23], and, for a fixed number of indeterminates, polynomial in the maximum degree of the input [17, 26].

Please also observe that the chain criterion that is used here is only one variant among many. For instance, alternatively one could compare $\mathrm{lcm}(\mathrm{lt}(p), \mathrm{lt}(g))$, $\mathrm{lcm}(\mathrm{lt}(q), \mathrm{lt}(g))$ and r not w. r. t. their degrees, but directly w. r. t. the term order $\preceq$. The reason for defining chainCrit as above is that we will mainly restrict $\preceq$

Algorithm 1 Buchberger's algorithm with chain criterion

Input: $F = \{f_1, \dots, f_n\} \subseteq K[x, y]$
Output: $G \subseteq K[x, y]$ s. t. $\text{ideal}(F) = \text{ideal}(G)$ and G is Gröbner basis

```
function GB(F)
    P ← {(f_i, f_j) | 1 ≤ i < j ≤ n}
    G ← F
    while P ≠ ∅ do
        choose some (p, q) from P
        P ← P \ {(p, q)}
        if chainCrit(p, q, G) then
            h ← sPoly(p, q)
            h ← totalReduce(h, G)
            if h ≠ 0 then
                P ← P ∪ {(g, h) | g ∈ G}
                G ← G ∪ {h}
            end if
        end if
    end while
    return G
end function
```

to *graded* orderings anyway (see following paragraphs). Moreover, the differences between the individual variants are comparatively small.

2.2 Complexity of the Algorithm

In order to obtain bounds on the complexity of Buchberger's algorithm in the bivariate case in terms of the number of elementary operations that are executed for given input F, it turns out to be sufficient to only know bounds on the degrees of the polynomials in the resulting Gröbner basis G, as shown in [2]:

Theorem 1. *For any finite $F \subset K[x, y]$ let*

$$D_F := \max\{\deg(\text{lt}(g)) \mid g \in \text{GB}(F)\}$$

i. e. the maximum degree of all leading terms in the Gröbner basis computed by Algorithm 1. Furthermore, let $C_F := \frac{(D_F+2)(D_F+1)}{2}$. Then at most

$$\binom{|F| + C_F}{2} \cdot \left(C_F \left(|F| + C_F\right) + \binom{C_F}{2} \right)$$

additions, multiplications and comparisons (w. r. t. $\preceq$) of polynomials are needed to compute $\mathrm{GB}(F)$.

Because of Theorem 1 the remaining part of this section is all about obtaining good (i. e. tight) bounds for D_F in terms of the degrees of the polynomials in F. Moreover, this is precisely what the formalization in *Theorema* deals with exclusively.

2.3 General Proof Strategy

In this subsection we describe the general strategy for obtaining and proving suitable bounds for D_F, pursued both in Buchberger's original papers as well as in the *Theorema* formalization.

At the very beginning, the case of arbitrary admissible term orderings is reduced to the case of *graded* orderings, i. e. orderings where the first criterion to decide which of two terms is greater is their degree. Knowing a bound for such orderings one can easily derive a bound that holds for any admissible ordering, if the corresponding ideal is 0-dimensional. For more details on this we refer to [4].

Summarizing, from now on we assume that $\preceq$ is a graded admissible term ordering, which, furthermore, is the only case that is treated in the formalization. The subsequent paragraphs describe the individual steps of the general strategy.

1. Exponent Vectors First of all, the problem of estimating the degrees of polynomials is reduced from a commutative-algebra- to a *combinatorial* problem, by mapping each non-zero polynomial to the exponent vector of its leading term (w. r. t. the graded ordering $\preceq$). This is justified by the fact that in Algorithm 1 it is only the leading terms of polynomials that influence the behaviour of the algorithm and the resulting Gröbner basis, be it when forming S-polynomials or in reductions. Exponent vectors are pairs of natural numbers, meaning that from now on we work exclusively in the space $\mathbb{N}^2$, and no appeal needs to be made to polynomials any more. This, in fact, is now precisely where the formalization in *Theorema* starts: There, everything is about exponent vectors (and tuples thereof) rather than about polynomials. As functions and predicates like lcm, $\deg$, divisibility and CHAINCRIT, defined for polynomials, in fact only depend on their arguments' corresponding exponent vectors, the same functions/predicates, by abuse of notation, will also be used for exponent vectors. For instance, if p and q are two exponent vectors, then $p|q$ iff $p_i \leq q_i$ for all $i = 1, 2$, where p_i and q_i refer to the i-th component of p and q, respectively. This notation will be used throughout the rest of this report.

2. Loop Invariant For each $G \subseteq \mathbb{N}^2$ (corresponding to the current basis in Buchberger's algorithm) the quantity $M_G + W_G$ is shown to be some kind of "loop invariant" of the main loop in Algorithm 1, in the sense that it does not *increase* (it may decrease, though). M_G and W_G are defined as

$$M_G := \max\{\deg(\mathrm{lcm}(a,b)) \mid a, b \in G \wedge \textsc{chainCrit}(a,b,G)\} \tag{2.2}$$

respectively

$$W_G := \min\{e_1 \mid e \in G\} + \min\{e_2 \mid e \in G\} \tag{2.3}$$

and the goal of this second step is to show

$$M_{G'} + W_{G'} \leq M_G + W_G \tag{2.4}$$

where G' is obtained from G by adding a new exponent vector h, corresponding to line 12 of Algorithm 1 where a new polynomial is added to the current basis. Of course, h is not completely arbitrary but has some specific properties, like the very important $\deg(h) \leq M_G$ since $\preceq$ is graded and h corresponds to a polynomial that is obtained by reducing the S-polynomial of two polynomials p and q for which the chain criterion holds, meaning that by definition of M_G we know $\deg(\textsc{sPoly}(p,q)) \leq M_G$.

3. Maximum Degree For each $F \subseteq \mathbb{N}^2$, $\mathrm{maxdeg}(F)$ is shown to be bounded from above by M_F, i. e.

$$\mathrm{maxdeg}(F) \leq M_F \tag{2.5}$$

$\mathrm{maxdeg}(F)$ is defined as the maximum degree of all exponent vectors in F.

4. Degree Bound The quantity $M_F + W_F$ that was shown not to increase in the course of Algorithm 1 in step 2 is now shown to be bounded from above by $2 \cdot \mathrm{maxdeg}(F)$, for all $F \subseteq \mathbb{N}^2$, i. e.

$$M_F + W_F \leq 2 \cdot \mathrm{maxdeg}(F) \tag{2.6}$$

As soon as all this is established, the whole elaboration is finished, as we can now conclude

$$\mathrm{maxdeg}(G) \leq_{(2.5)} M_G \leq M_G + W_G \leq_{(*)} M_F + W_F \leq_{(2.6)} 2\,\mathrm{maxdeg}(F)$$

where F corresponds to the input of Buchberger's algorithm and G to its output. $(*)$ is justified by an inductive argument exploiting formula (2.4).

The final result of the complexity analysis is summarized in the following theorem:

Theorem 2. *For all finite $F \subset K[x, y]$ and all graded term orderings $\preceq$: The Gröbner basis G computed by Algorithm 1 on input F w. r. t. $\preceq$ satisfies*

$$\mathrm{maxdeg}(G) \leq 2\,\mathrm{maxdeg}(F) \tag{2.7}$$

$\mathrm{maxdeg}(G)$ *and* $\mathrm{maxdeg}(F)$ *refer to the maximum total degree of the polynomials in G respectively F.*

2.4 Improvements in the Formalization

As indicated already before, some improvements in the formalization in *Theorema* could be achieved compared to the original elaboration of the theory by Buchberger in the cited papers. Here, we list and discuss them in detail.

2.4.1 Ground Domain

By definition, exponents in polynomials are non-negative integers. Hence, it is only natural to carry out steps 2 – 4 of the general proof strategy in the space $\mathbb{N}^2$, just as described in the previous subsection; This is exactly how it was done in Buchberger's original papers.

However, since absolutely no appeal to polynomials is made in those steps and really everything happens in the "space of exponent vectors" $\mathbb{N}^2$, there is nothing that hinders us from trying to generalize the ground domain from $\mathbb{N}$ to wider classes of mathematical structures (even though this might not make sense when going back to polynomials in the end). And indeed, a detailed analysis of the proofs of the various formulas revealed that only quite a few properties of $\mathbb{N}$ are actually needed, therefore allowing us to replace $\mathbb{N}$ by the much wider class of so-called *totally-ordered commutative monoids*, defined as follows:

Definition 3. $(D, +, \leq)$ is a totally-ordered commutative monoid iff

1. $(D, +)$ is a commutative monoid
2. $(D, +)$ is cancellative, i. e.

   $$x + z = y + z \Rightarrow x = y$$

   for all $x, y, z \in D$
3. $(D, \leq)$ is a total ordering
4. $+$ is monotonic w. r. t. $\leq$ on D, in the sense

   $$x \leq y \Rightarrow x + z \leq y + z$$

   for all $x, y, z \in D$

As can easily be seen, apart from $\mathbb{N}$ there are many other well-known mathematical structures that are totally-ordered commutative monoids, among them $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ with any order relation that corresponds to an admissible term ordering (e. g. lexicographic). So, this really is a massive generalization.

There is one important thing to note, though: Apparently, totally-ordered commutative monoids are not required to have a least element, or even to be well-ordered by $\leq$. This might appear strange at first sight, since domains that are not well-ordered make it impossible to draw any conclusions about the complexity of Buchberger's algorithm, even when knowing bounds on the degrees of the polynomials in the final Gröbner basis. But keep in mind that steps 2 – 4 of the general strategy are only concerned with finding exactly such degree bounds, but *not* with the actual complexity of the algorithm. The degree bound $2\,\mathrm{maxdeg}(F)$ is valid for all finite $F \subset D^2$ (if D is a totally-ordered commutative monoid and a graded ordering on D^2 is used), but deriving actual complexity results, as it is done in Theorem 1, is indeed only possible if much stronger requirements on D are imposed.

2.4.2 Number of Indeterminates

Although the main result $\mathrm{maxdeg}(G) \leq 2\,\mathrm{maxdeg}(F)$ was only proved for the case of two indeterminates, or, in exponent vector parlance, in the space of exponent vectors in two dimensions, many intermediate auxiliary results could be proved in arbitrary dimension. This will certainly prove a huge benefit if the complexity of Buchberger's algorithm in a higher number of indeterminates, or even arbitrarily many indeterminates, is investigated by similar means sometime.

2.4.3 Cover of Space of Exponent Vectors

The next improvment is a bit technical: In the proof of formula (2.4) various cases are distinguished depending on where the new vector h lies in the two-dimensional space of exponent vectors, w. r. t. the current set G. To this end, the space of exponent vectors (or, in short, *exponent space*) is partitioned into several sets: In the original elaboration in [12] these sets are above G, below G, interior of G, and exterior of G (Strictly speaking, the exterior is again divided into two sets that can be dealt with by symmetric arguments, though).

In the formalization in *Theorema*, a different splitting of the exponent space is pursued: Above G (same as in original elaboration), rectangular region of G, and far exterior of G (again divided into two "symmetric" sets). This splitting is only a cover of the exponent space, since the set above G and the rectangular region of G are not disjoint in general.

The reason for this deviation from the original papers is the following: The new rectangular region of G comprises the whole set below G, interior of G, and parts of exterior of G from the "old" partition. Hence, what have been three cases previously are now only one single case that, furthermore, can be dealt with by a very nice new argument that in fact proves correct the much stronger claim $M_{G'} \leq M_G$ (if the new element h is in the rectangular region of G). For more details on all that please see Section 5.

2.4.4 Simplification of Proof of (2.6)

Originally, formula (2.6) was proved in [4] by first reducing the case of arbitrary sets F to the case of so-called *contours* and then proving the formula only for contours. The latter part is easy, but the reduction of arbitrary sets to contours is very cumbersome and involves many tedious case distinctions, making up in total eleven pages of Buchberger's original paper. However, a close investigation revealed that all this cumbersome reduction to contours is *not needed at all*, since the proof of the second part given for the case of contours works more or less in exactly the same way for any set of exponent vectors. Since it is really short, we spell it out in full detail (even for totally-ordered commutative monoids):

Proof of (2.6). Choose $F \subseteq D^2$ arbitrary but fixed, where D is a totally-ordered commutative monoid. We define the auxiliary notions M_i and m_i for $i = 1, 2$ as

$$a_i := \max\{e_i \mid e \in F\}$$

and

$$b_i := \min\{e_i \mid e \in F\}$$

With this definition, we apparently have $W_F = b_1 + b_2$. Moreover, since M_F is the maximum degree of the least common multiples of some exponent vectors in F, it can certainly be bounded from above by $a_1 + a_2$. Hence:

$$M_F + W_F = M_F + b_1 + b_2 \leq a_1 + a_2 + b_1 + b_2 = (a_1 + b_2) + (a_2 + b_1)$$

We show that both summands in the last expression can be bounded from above by $\mathrm{maxdeg}(F)$, which finishes the proof. W. l. o. g. we only consider the first summand $a_1 + b_2$; The other one can be treated analogously. Since a_1 is the maximum first component of all vectors in F, there must be some vector $e \in F$ with $e_1 = a_1$. By definition of b_2, the second component of e, e_2, must be at least as big as b_2, hence we can conclude

$$a_1 + b_2 = e_1 + b_2 \leq e_1 + e_2 = \deg(e) \leq \mathrm{maxdeg}(F)$$

□

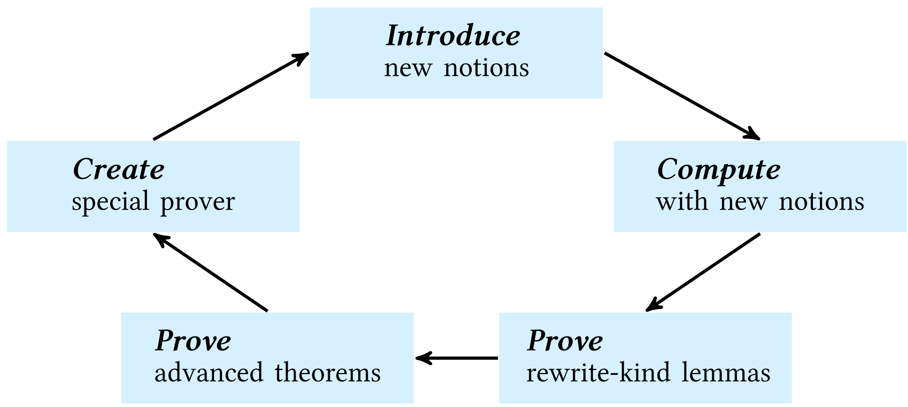

Figure 1: Theory Exploration Cycle

3 Theory Exploration Cycle

The so-called *Theory Exploration Cycle* is a structured method for developing mathematical theories in a systematic way, suitable both for working "with pencil and paper" as well as for formalizing mathematics in computer systems such as *Theorema*. Although such very general methods usually come in many different "flavours", with some minor variations here and there, we present here what in our perception seems to be a good and reasonable approach. In particular, what we present here is exactly the kind of theory exploration cycle we followed in the formalization of the complexity analysis in *Theorema*.

Figure 1 shows the individual steps that form the Theory Exploration Cycle, as well as their order. The remaining part of this section is dedicated to explaining all the steps in more detail, where each step is also illustrated by concrete examples from the complexity-formalization.

For the sake of completeness it has to be mentioned that, naturally, there are many variations of the Theory Exploration Cycle. For instance, one may argue that *inventing problems* is an integral part of exploring mathematical theories as well. Hence, a different, more "algorithm-oriented" Theory Exploration Cycle might have **Introduce new notions** – **Prove rewrite-kind lemmas** – **Invent problems** – **Solve problems** – **Invent algorithms for automatic solution** – **Create special prover incorporating algorithmic solutions** as its individual steps. The aim of this report is neither to present a complete list of all possible variations, nor compre them to each other and argue why one or the other is "best". All what Figure 1 does is illustrating the strategy we pursued in our own formal treatment of one particular mathematical theory, but we are aware that it could have been done differently, too.

3.1 Introducing New Notions

The truth is that we went through the Theory Exploration Cycle not only once, but one-and-a-half times, meaning that node **Introduce new notions** was visited twice.

The first time we introduced very basic notions needed for later building upon them the more "interesting" ones at the heart of the complexity analysis. These basic notions include tuples, total order relations, and totally-ordered commutative monoids, as presented in Section 2.4.1. They all have in common that they are more or less *independent* of the complexity analysis and can easily be used also in completely different formalizations; This, in particular, holds for tuples and total order relations, but also totally-ordered commutative monoids might be of interest elsewhere.

"Introducing new notions" on a formal level means to either define functions or predicates explicitly by means of equalities and equivalences, or to state some of their properties implicitly. For instance, introducing the notion of *strict version* of a total order relation $\leq$ can be done explicitly as

$$\underset{x,y}{\forall} \quad x < y \;\; :\Leftrightarrow \;\; (x \leq y \wedge x \neq y)$$

On the other hand, the notion of totally-ordered commutative monoids can only be introduced by stating its properties implicitly, simply because it is not only one concrete domain, but a whole class of domains.

Remark 1. In the future, knowledge about basic notions like tuples should be collected in so-called *knowledge archives* in *Theorema* and distributed together with the system. This, of course, would save not only the first visit of the **Introduce new notions**-node, but the whole first round of the Theory Exploration Cycle, which is dedicated entirely to the basic notions. However, at the time of our formalization no such archives were available yet.

After the first round of the Theory Exploration Cycle we concentrated on the notions directly related to the complexity analysis, such as exponent vectors, the chain criterion, and the quantities M_F and W_F as discussed in Section 2.3. All of them are defined in terms of the basic notions introduced before.

3.2 Computing

As soon as new notions have been introduced, one wants to do some computations with them. In our case, we considered concrete tuples T of exponent vectors and computed M_T and W_T, and checked whether the chain criterion holds for a given pair of exponent vectors w. r. t. a given tuple of exponent vectors. In all of these computations we restricted the ground domain to $\mathbb{N}$, but we also tried some examples in higher dimension, i. e. exponent vectors corresponding to more than only two indeterminates. This provided us with a counterexample that a certain important theorem does not hold in general, but really only in two dimensions.

The reason why after introducing new notions we can immediately compute with them lies in the fact that computation in *Theorema* is just *simplification modulo equational theories*, i. e. equalities and equivalences (possibly quantified and/or conditional) are automatically turned into rewrite rules that are later used by a built-in rewrite mechanism to simplify expressions to some "normal form". In particular, since CHAINCRIT is defined exactly by a quantified equivalence (see Formula (2.1)), every occurrence of it will be replaced by the right-hand-side of its definition, which is then simplified further. Quantifiers in general, and the existential quantifier in the definition of CHAINCRIT, in particular, can be computed in *Theorema* if the range of the variable they bind is evidently finite.

3.3 Proving Rewrite-Kind Lemmas

This node in the Theory Exploration Cycle is one of the most important ones: Proving rewrite-kind lemmas means collecting all the available explicit and implicit knowledge about the notions introduced before and extending this knowledge by new equalities/equivalences. The lemmas are called "rewrite-kind" because, as already indicated above, equalities/equivalences can be used for rewriting expressions; This does not only happen in computations, but also in proofs, and in fact rewriting is one of the elementary general-purpose proving techniques employed by the special prover we created for the complexity analysis (see Section 4).

A typical example of a rewrite-kind in the current framework lemma is the following: If some value x is added to the minimum over all elements of a tuple A, then x could also be added to each individual element of A *before* taking the minimum. Formally:

$$\underset{x,A}{\forall} \quad \min(A) + x = \min(\langle A_i + x \underset{i=1,\ldots,|A|}{|} \rangle) \tag{3.1}$$

Expressions of the form $\langle t[i] \underset{i=a,\ldots,b}{|} \varphi[i] \rangle$ are so-called *abstraction tuples* (in analogy to abstraction terms in set theory), and $\langle \ | \ \rangle$ is the corresponding *tuple quantifier*. Although they are built-in concepts in *Theorema*, at the time of our formalization no built-in knowledge about them in the form of inference rules was available, which explains why we had to provide such inference rules ourselves (see again Section 4).

Formula (3.1) can be proved formally, and as soon as its correctness has been established, it can be used in other proofs to rewrite expressions of the form $\min(A) + x$. Using quantified rewrite-kind formulas among our assumptions as rewrite rules saves us from finding instantiations for the bound variables, as this is accomplished automatically by purely syntactic matching (Of course, things get much more complicated if rewrite-kind formulas are constrained by conditions; See Section 4.1).

3.4 Proving Advanced Theorems

The core of each theory exploration is first finding and then proving interesting, non-trivial theorems about the new notions introduced in the first step. In our case, we were of course mostly interested in proving the main theorem $\operatorname{maxdeg}(G) \leq 2 \operatorname{maxdeg}(F)$, but we also regard other results that are needed to establish this bound interesting enough to categorize them as "advanced theorems". An example of such a result is the fact that M_A does not increase when adding some new exponent vector x to A if x lies in a particular region of the whole space of exponent vectors; More information on this can be found in Section 5) where the overall flow of the proof of the main theorem, as well as the most important auxiliary results, are explained in detail.

Please note that for proving the advanced theorems heavy use was made of the rewrite-kind lemmas described in the previous paragraph. This should not come as a surprise, as it is precisely why we stated and proved them.

3.5 Creating Special Provers

The last step of the Theory Exploration Cycle consists of creating a new *special prover* that incorporates all the knowledge about the notions introduced in the very first step in a neat and efficient way, such that later, when traversing the cycle again and building upon the theory just explored, all these notions can be handled in a *natural* and *efficient* way. It should not be necessary later on to fall back to the very definitions of, or at least lemmas about, "old" notions introduced long ago, if this would result in extremely long and/or complicated proofs, and if "lifting" knowledge about those notions to the level of inferencing would give short and elegant proofs – That, at least, is the idea behind special provers and proving *by* intermediate principles, summarized in [7].

The special prover we created for the complexity analysis is able to handle the basic notions (tuples, total order relations, etc.) in an efficient way that allows to put the focus in proofs really on the interesting notions like CHAINCRIT, without having to fiddle around with a multitude of formulas over and over just to use, say, associativity of $+$ in totally-ordered commutative monoids at some point. Section 4 describes the prover in detail.

However, we did not create a special prover also in the next round of the Theory Exploration Cycle, i. e. a special prover incorporating knowledge about the specific notions for the complexity analysis. This is simply not necessary for proving the main theorem $\mathrm{maxdeg}(G) \leq 2\,\mathrm{maxdeg}(F)$, but it would most likely be necessary when developing a new theory upon the complexity analysis.

4 Complexity Prover

For the formal verification of the complexity analysis in *Theorema* a new *special prover*, called *Complexity Prover*, was designed and implemented. The paradigm of creating special provers for individual theories has been an integral part of the philosophy of *Theorema* ever since (c. f. [9]), as indicated already in Section 3.5.

Provers in *Theorema* consist of two parts: A collection of *inference rules* and a *proving strategy*, which are, however, mostly independent of each other. Since they operate on formulas, and formulas are elements of *Theorema*'s object level, provers themselves necessarily have to be elements of the meta level of *Theorema*, which is *Mathematica*. Hence, "implementing a prover in *Theorema*" actually means implementing inference rules and/or proving strategy in *Mathematica*. The meta-theoretical consequences of such an approach, its drawbacks and possible solutions, are addressed in [18, 16]; Here, we do not deal with any of the issues presented there and exclusively concentrate on the design of the Complexity Prover.

Special provers necessarily consist of two different kinds of inference rules: General-purpose inference rules and special inference rules (making the prover "special"). The former ones are always needed when reasoning in (higher-order) predicate logic (logical connectives, logical quantifiers), whereas the latter ones deal with specific notions and concepts in the theory currently explored (c. f. the Theory Exploration cycle in Section 3). Inference rules of the Complexity Prover of either of the two kinds will be explained in more detail in the next two subsections.

4.1 General-Purpose Inference Rules

Apart from the usual inference rules of predicate logic sequent calculus, like introducing "arbitrary but fixed" constants for universally quantified variables in the proof goal, there are also other general-purpose rules that are part of the Complexity Prover: *Interactive* inference rules and *rewrite*-rules.

Interactive inference rules require some sort of user interaction when they are about to be applied. The first example of such a rule that comes to one's mind probably is a rule that instantiates universally quantified formulas in the knowledge base or existentially quantified formulas in the goal. Finding suitable instantiations, in general, is a non-trivial task, and a human operator "guiding" the search might have more insight (and certainly more intuition) and therefore might be more likely to provide the prover with suitable terms. And indeed, two of the in total four interactive inference rules of the Complexity Prover are precisely of that kind. The third interactive rule allows the user to exchange the current goal ψ with the negation of an assumption φ (such that instead of ψ one proves $\neg\varphi$, assuming $\neg\psi$), and the fourth interactive rule allows the user to select any implication in the knowledge base, whose premise is then proved in a subproof and whose conclusion is added to the knowledge base in the main branch of the proof (this is useful if *modus ponens* does not apply).

The purpose of rewrite-rules, on the other hand, is precisely to take the instantiation of (certain) universal formulas in the knowledge base off the user's shoulders. Namely, rewrite-kind formulas (c. f. Section 3.3) are internally turned into rewrite rules that can be used to replace terms by equal ones, respectively formulas by equivalent ones. Universally quantified variables are turned into patterns and the applicability of a rewrite-rule can then simply be determined by syntactic pattern matching (at least if no additional conditions occur; see below). For instance, consider formula

$$\underset{x,y}{\forall} x < y \Leftrightarrow (x \leq y \wedge x \neq y) \tag{4.1}$$

This formula can be used to rewrite, say, $a < 4$ into $a \leq 4 \wedge a \neq 4$, without a human operator having to instantiate it with $x \leftarrow a$ and $y \leftarrow 4$ himself. In fact, the driving engine behind rewriting in proofs is exactly the same engine that performs computations in *Theorema* in general. [10] describes the concept of computation in proofs in *Theorema* in detail, albeit only for *Theorema* 1.

A problem arises in connection with conditional rewrite rules: In practice, formula (4.1) will be constrained by the condition on x and y being elements of some set A on which $<$ and $\leq$ are defined. Hence, it should rather read as

$$\underset{x,y}{\forall} (x \in A \wedge y \in A) \;\; \Rightarrow \;\; x < y \Leftrightarrow (x \leq y \wedge x \neq y) \tag{4.2}$$

and this is now really more or less a formula that actually appears in the formalization of the complexity analysis. Still, it can be used for rewriting $a < 4$, but only if $a \in A$ and $4 \in A$ hold. The important question here is *how*, and more precisely, *when* this condition is checked. There are basically two alternatives:

1. Require $a \in A$ and $4 \in A$ to be known when the rewrite is attempted.

2. Prove $a \in A$ and $4 \in A$ in a separate subproof, if they are not known already.

From the purely logical point of view, there is no difference between the two alternatives: In either case, the proof can only succeed if the condition really holds, otherwise it fails. Hence, both give rise to a correct inference rule. From the *efficiency* point of view, however, there is a difference: Alternative 1 never applies a rewrite that cannot be applied, and no time is wasted trying to prove a condition that may not even hold. However, rewrites that *are* applicable in principle might not be carried out either. Alternative 2, on the other hand, does not miss any conditional rewrites (at least those whose conditions can be proved), but might waste time proving invalid conditions, too. Therefore, none of the two possibilities is optimal.

The way we tackled this problem of conditional rewrite-rules is straightforward, though not very elegant: By default, the first alternative is employed, and whenever we encountered a formula that should rather be treated according to the second alternative, we simply made a new special inference rule out of it – We "lifted" it to inference level (see next subsection). A better way to solve the problem in general would be to develop a mechanism for attaching some kind of "meta information" to conditions that tells the rewrite engine how to proceed, i. e. which of the two possible strategies to pursue. In our opinion, *definedness conditions* are a natural candidate for the second alternative: If they do not hold, then a certain expression is not even *defined* in the sense that absolutely nothing is known about it – And this is something one usually would not expect to happen in a proof (But still, even this is checked!).

The development of such a mechanism and a closer investigation of how to deal with inference rules constrained by conditions in general is work in progress.

4.2 Special Inference Rules

"Special" inference rules are rules that handle specific notions and concepts at the foundations of a particular theory in a neat and efficient way, such that in proving one can concentrate on the more "interesting" notions one is currently exploring (recall the Theory Exploration cycle in the previous section). A typical example of such a special inference rule, which is also part of our Complexity Prover, is a rule that deals with associative-commutative-cancellative functions: If $+$ is a binary function known to be associative, commutative and cancellative, and, say,

$$a + (4 + b) < (b + a) + 4 \tag{4.3}$$

has to be shown, then it should not be necessary to fall back to the very definitions of associativity, commutativity and cancellativity in order to rewrite the formula in several steps into

$$a < 4$$

if this is not where the focus of the current proof lies. Rather, the special inference rule exploits all the properties of $+$ at once and therefore is able to deal with formulas like (4.3) directly, without any tedious intermediate steps. In some sense, the formulas describing associativity, commutativity and cancellativity are "lifted" from the object- to the inference level. At the moment, in *Theorema* this lifting process still has to be carried out manually, i. e. the inference rules have to be implemented in *Mathematica* without any reference to the formulas they actually originate from, and hence without any justification regarding their correctness. Therefore it is clear that a mechanism that automates the lifting (at least for a certain class of formulas) would be a great benefit [8].

Lifting is also needed for another kind of concepts: *Quantifiers*. At the moment, quantifiers can only be introduced at the meta level, meaning that their *syntax* has to be hard-coded in the implementation of *Theorema*, and their *semantics* has to be defined by means of inference rules in provers; Both tasks have to be carried out in *Mathematica*. Clearly, the ability of introducing quantifiers directly at the object level would prove to be a huge improvement compared to the current status, and investigating the various possibilities for providing *Theorema*'s meta level with functionality to handle quantifiers introduced at the object level (in computations and as inference rules in proofs) is work in progress. A promising approach seems to be the use of higher-order functions/predicates that are then turned into quantifiers, a technique that was already introduced in [13] and is now implemented, for instance, in the Isabelle/Isar proof assistant [30].

In the formalization of the complexity analysis we made use of three quantifiers (different from $\forall$ and $\exists$): $\mathrm{argmin}\cdot$, $\mathrm{argmax}\cdot$ and the tuple-quantifier $\langle\cdot|\cdot\rangle$ (analogous to *abstraction terms* in set theory). A typical inference rule giving semantics to the tuple-quantifier is, for instance, the following:

$$\frac{\vdash \underset{i=a,\ldots,b}{\forall} \varphi[i] \Rightarrow \psi[t[i]]}{\vdash \underset{j=1,\ldots,\langle t[i] \underset{i=a,\ldots,b}{|} \varphi[i]\rangle}{\forall} \psi[\langle t[i] \underset{i=a,\ldots,b}{|} \varphi[i]\rangle_j]}$$

Intuitively, this rule says "If we have to prove that a property ψ holds for all elements of the tuple $\langle t[i] \underset{i=a,\ldots,b}{|} \varphi[i]\rangle$, then we can show instead that ψ holds for all terms $t[i]$ where $\varphi[i]$ holds". It is important that j appears in formula ψ only as subscript (i. e. index) of the tuple.

4.3 Proving Strategy

Theorema provers not only depend on a collection of inference rules, but also on a strategy that guids their application. Although inference rules and strategy are mostly independent of each other, and we therefore could have taken an existing one, we decided, for various reasons, to create our own proving strategy that is especially tailored for the needs of the inference rules of the Complexity Prover. In particular, in the new strategy interactive rules are treated differently than in all other strategies that are currently available.

In *Theorema*, every inference rule is endowed with a so-called *rule priority* that may (or may not) be used by strategies to decide in which order inference rules are tried on proof situations, and how to proceed if several rules are applicable (priority-values range from 1 to 100, where lower value means higher priority). Each rule comes with a predefined default priority that, however, can be changed by the human user when setting up the proof task. Our new proving strategy uses these priorities for partitioning the collection of inference rules into four classes: High-priority rules (1–4), medium-priority rules (5–90), low-priority rules (91–100), and interactive rules.

High-Priority Rules are tried first on proof situations, in an order that respects their priorities. As soon as some rule is applicable, the search for further applicable rules is aborted and only that one rule is applied. No alternative branches in the proof tree are created. Hence, high-priority rules can be viewed as rules that shall be applied whenever possible and whose application certainly does not have any negative effect on the proof search.

Medium-Priority Rules are tried only if no high-priority rule is applicable. Again, they are tried in an order which respects their priorities, but in contrast to high-priority rules *all* of them are tried, and in case more than one is applicable, several alternatives in the proof tree are created, one for each applicable rule.

Low-Priority Rules are tried only if no high- nor medium-priority rule is applicable. They are treated just like the medium-priority rules, i. e. all of them are tried and possibly several alternatives in the proof tree are created. Low-priority rules can be viewed as rules whose application should be avoided whenever possible, because it might have negative effects on the proof search (w. r. t., for instance, efficiency). Sometimes, however, they really have to be applied, of course.

Interactive Rules are, as their name suggests, rules that require some sort of interaction with a human operator. Interactive rules, regardless of their priorities, are always tried last, even after low-priority rules. In case more than one interactive rule is applicable, the human user may choose which one to apply interactively. However, in any case two alternatives in the proof tree are created, such that it is always possible to "go back" to the proof situation before the application of the interactive rule, and "undo", in some sense, wrong interactions.

4.4 Summary

This subsection, and in particular Table 1, provides a summary of all the inference rules of the Complexity Prover: *General-purpose* refers to the rules described in Section 4.1, *Integers* refers to rules dealing with membership in integer intervals, *Tuples* refers to rules dealing with all aspects of tuples that are needed in the verification, *Addition* and *Order relation* refer to rules handling the monoid operation ("$+$") respectively the order relation ("$\leq$") in totally-ordered commutative monoids, and *Minimum/Maximum* refers to rules dealing with $\min$, $\max$, argmin and argmax.

A more detailed description of the individual inference rules can be found in the *Theorema* notebook containing the formalization.

General-purpose	28
Integers	2
Tuples	11
Addition	2
Order relation	7
Minimum/Maximum	7
Total	**57**

Table 1: Number of inference rules in each category.

5 Formalization in *Theorema*

This section is dedicated to describing the formalization, and in particular the development of the proof of the main result (2.7), in more detail; For an overview of the theory exploration we refer to Section 3, and readers interested in the formalization itself (i. e. the *Theorema* notebook) are kindly referred to [22].

For the most part, the proof development is modeled after the one in [12, 4], meaning that whenever not explicitly stated otherwise the ideas underlying a certain step in a proof are taken from there. However, there do exist some deviations, mostly for the sake of simplification, that have partially already been mentioned in Section 2.4.

The following notions, notations, and conventions will be used throughout this section:

- Degree ($\deg$), divisibility ($|$), and least common multiple (lcm) of exponent vectors, as defined in Section 2.

- $\textsc{chainCrit}$, M_A, W_A and $\operatorname{maxdeg}(A)$ as defined in Section 2.3, with the slight modification that they are now defined for *tuples* A of exponent vectors, rather than sets, and that we write $\mathrm{M}(A)$, $\mathrm{W}(A)$ instead of M_A, W_A, respectively.

- Function $\cdot^- : \{1,2\} \to \{1,2\}$, defined as $1^- = 2$, $2^- = 1$.

- Function $\cdot \curvearrowleft \cdot$ that appends its second argument to its first argument, if this is a tuple.

- Function $|\cdot|$ giving the length of its argument, if this is a tuple.

- Exponent vector $\mathrm{L}(A)$ defined as the least common multiple of *all* exponent vectors in A.

- Exponent vector $\mathrm{K}(A)$ defined as the greatest common divisor of *all* exponent vectors in A (such that $\mathrm{W}(A) = \deg(\mathrm{K}(A))$).

- Predicates $\mathrm{isAbove}$, $\mathrm{inRectangle}$ and $\mathrm{inFarExterior}$ as defined below. These are the predicates defining the (new) cover of the exponent space, as indicated already in Section 2.4.

- The dimension of the space of exponent vectors will be denoted by n. Recall that the dimension corresponds to the number of indeterminates in the underlying polynomial ring.

- Typed variables: A, G for tuples of exponent vectors, x for exponent vectors, k for elements of $\{1,2\}$

Definition 4.

$$\mathrm{isAbove}(x, A) \;:\Leftrightarrow\; \underset{i=1,\ldots,|A|}{\exists} A_i | x \tag{5.1}$$

$$\mathrm{inRectangle}(x, A) \;:\Leftrightarrow\; x | L_A \tag{5.2}$$

$$\mathrm{inFarExterior}(x, A, k) \;:\Leftrightarrow\; x_k < K(A)_k \wedge x_{k^-} > L(A)_{k^-} \tag{5.3}$$

Please note that the definitions of $\mathrm{isAbove}$ and $\mathrm{inRectangle}$ are completely independent of the dimension n of the exponent space, whereas for $\mathrm{inFarExterior}$ n must be at least 2 (in fact, it only makes sense if $n = 2$).

Figure 2 illustrates the notions defined above in the exponent space of dimension 2 ($\mathrm{isAbove}$ is not explicitly shown, but it is just the whole region “above” the staircase).

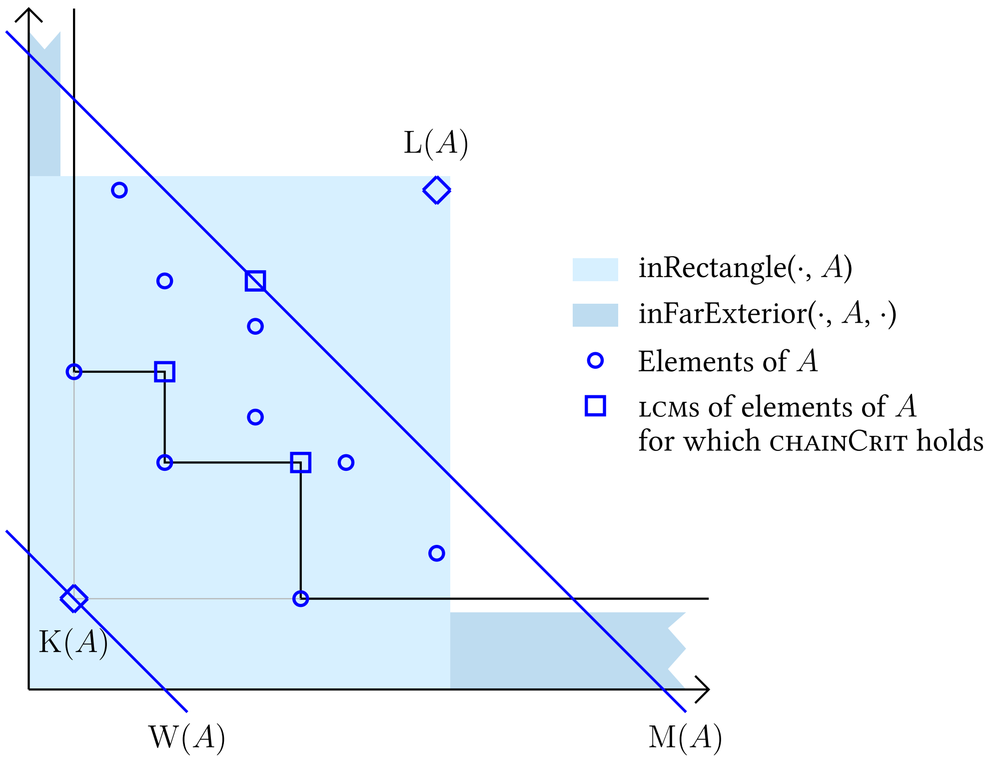

Figure 2: Cover of the two-dimensional exponent space

5.1 Cover of the Exponent Space

Figure 2 suggests that, if $n = 2$, the three predicates really cover the whole exponent space (w. r. t. any non-empty tuple A), in the sense that for each exponent vector at least one of them holds. And indeed, this is really the case:

Lemma 5. *Assume* $n = 2$. *For every non-empty tuple* A *of exponent vectors and every exponent vector* x *at least one of the following holds:*

(i) $\mathrm{isAbove}(x, A)$

(ii) $\mathrm{inRectangle}(x, A)$

(iii) $\mathrm{inFarExterior}(x, A, 1)$

(iv) $\mathrm{inFarExterior}(x, A, 2)$

Proof. A formal, semi-automatically generated proof of Lemma 5 can be found in the *Theorema*-notebook containing the formalization: Formula (*cover*) in Section "Lemmata on Specific Notions" / "Special Case $n = 2$" / "Cover of Exponent Space". See also Figure 3. □

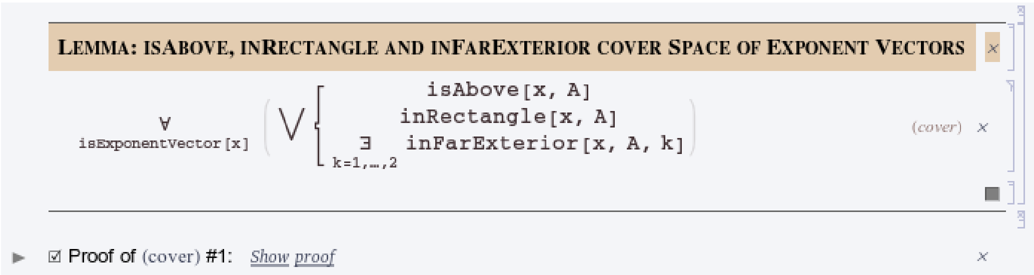

Figure 3: Lemma 5, formalized in *Theorema*

5.2 Bounding $\mathrm{M}(A \curvearrowleft x)$

The following theorem states a very important property of $\mathrm{M}(A \curvearrowleft x)$:

Theorem 6. *In all dimensions n: For all non-empty tuples A of exponent vectors and exponent vectors x the following inequality holds:*

$$\mathrm{M}(A \curvearrowleft x) \leq \max(\mathrm{M}(A), \mathrm{M}(x, A), \deg(x))$$

where $\mathrm{M}(x, A)$ *is defined as*

$$\mathrm{M}(x, A) \;:=\; \max(\deg(\mathrm{lcm}(x, A_i)) \underset{i=1,\ldots,|A|}{|} \textsc{chainCrit}(x, A_i, A))$$

Proof. A formal, semi-automatically generated proof of Theorem 6 can be found in the *Theorema*-notebook containing the formalization: Formula ($M\curvearrowleft$) in Section "Lemmata on Specific Notions" / "General Case" / "M". See also Figure 4. □

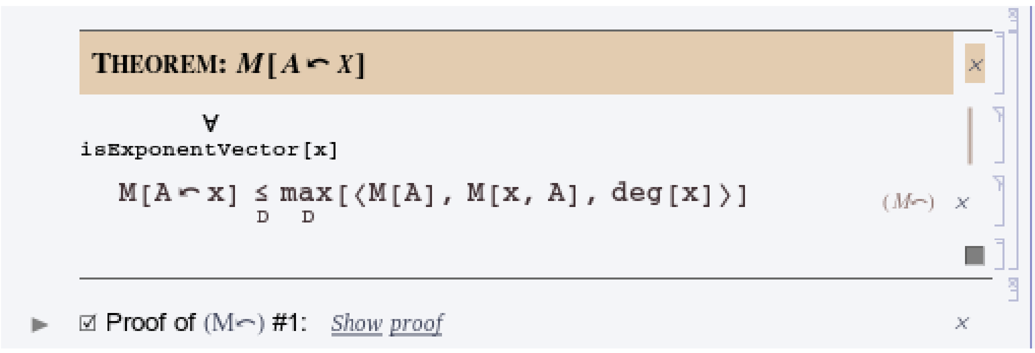

Figure 4: Theorem 6, formalized in *Theorema*

5.3 Bounding $\operatorname{maxdeg}(A)$

The theorem in this subsection states an inequality that is of interest on its own:

Theorem 7. *In all dimensions n: For all non-empty tuples A of exponent vectors the following inequality holds:*

$$\operatorname{maxdeg}(A) \leq \mathrm{M}(A)$$

Proof. A formal, semi-automatically generated proof of Theorem 7 can be found in the *Theorema*-notebook containing the formalization: Formula (*maxdeg≤M*) in Section “Theorems” / “General Case”. See also Figure 5. □

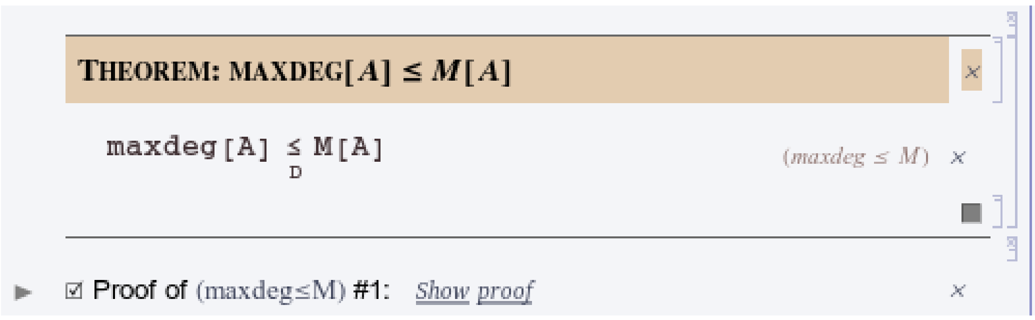

Figure 5: Theorem 7, formalized in *Theorema*

5.4 Bounding $\mathrm{M}(A \curvearrowleft x)$ if x in Rectangular Region

The theorem below is one of the two theorems needed for proving Theorem 10. Please note that it is nowhere explicitly stated in [12, 4].

Theorem 8. *If $n = 2$: For all non-empty tuples A of exponent vectors and all exponent vectors x with* $\deg(x) \leq \mathrm{M}(A)$ *and* $\operatorname{inRectangle}(x, A)$ *the following inequality holds:*

$$\mathrm{M}(A \curvearrowleft x) \leq \mathrm{M}(A)$$

Proof. A formal, semi-automatically generated proof of Theorem 8 can be found in the *Theorema*-notebook containing the formalization: Formula (*88*) in Section “Theorems” / “Special Case $n = 2$” / “Rectangular Region”. See also Figure 8.

However, since the proof of the theorem cannot be found in the literature, we also sketch it here:

Let A be an arbitrary but fixed non-empty tuple of exponent vectors, and let x be an arbitrary but fixed exponent vector with

$$\deg(x) \leq M(A) \tag{A.1}$$

$$\mathrm{inRectangle}(x, A) \tag{A.2}$$

We have to show $\mathrm{M}(A \curvearrowleft x) \leq \mathrm{M}(A)$, which is accomplished by showing $\deg(\mathrm{lcm}(a, x)) \leq \mathrm{M}(A)$ for any element a of A such that chainCrit holds for a and x (follows readily from the definition of M). Hence, we choose some a. b. f. element a of A, assume

$$\textsc{chainCrit}(a, x, A) \tag{A.3}$$

and show

$$\deg(\mathrm{lcm}(a, x)) \leq \mathrm{M}(A) \tag{G.1}$$

Now we distinguish four cases, depending on the relative positions of a and x in the two-dimensional exponent space.

Case I: $x_1 \leq a_1$ and $x_2 \leq a_2$.
In this case we obviously have $\mathrm{lcm}(a, x) = a$ and hence also $\deg(\mathrm{lcm}(a, x)) = \deg(a)$. Together with Theorem 7 we get the desired result.

Case II: $a_1 \leq x_1$ and $a_2 \leq x_2$.
In this case we obviously have $\mathrm{lcm}(a, x) = x$ and hence also $\deg(\mathrm{lcm}(a, x)) = \deg(x)$. Together with assumption (A.1) we get the desired result.

Case III: $a_1 < x_1$ and $x_2 < a_2$.
In order to prove (G.1) it is sufficient to find an element b of A with

$$\deg(\mathrm{lcm}(a, x)) \leq \deg(\mathrm{lcm}(a, b)) \tag{G.2}$$

$$\textsc{chainCrit}(a, b, A) \tag{G.3}$$

Let $C := \{c \mid c \text{ is an element of } A, x_1 \leq c_1\}$. C is finite, and because of assumption (A.2) it is also non-empty, meaning that it contains an element b such that b_1 is minimal among all c_1 for $c \in C$ (c. f. Figure 6). Of course, in general such a b might not be unique, but this does not matter.

We claim that b witnesses (G.2) and (G.3). (G.2) is trivially witnessed by b, since

$$\begin{aligned} \deg(\mathrm{lcm}(a, x)) & \underset{\text{case assumption}}{=} x_1 + a_2 \leq \\ & \underset{x_1 \leq b_1}{\leq} b_1 + a_2 \leq \\ & \leq \deg(\mathrm{lcm}(a, b)) \end{aligned}$$

For proving (G.3) we again distinguish two cases.

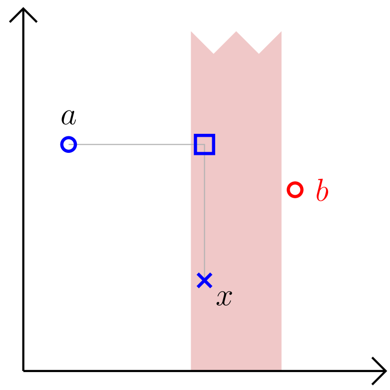

Figure 6: The relative positions of a, b and x. No element of A lies in the shaded region.

Case III.A: $a_2 \leq b_2$.
In this case we have $a|b$, and therefore chainCrit(a, b, A) certainly holds (as always if one point divides the other, as can easily be verified).

Case III.B: $b_2 < a_2$.
We have to prove that there does not exist an element d of A with

$$d \mid \operatorname{lcm}(a, b) \tag{G.4}$$

$$\deg(\operatorname{lcm}(a, d)) < b_1 + a_2 \tag{G.5}$$

$$\deg(\operatorname{lcm}(b, d)) < b_1 + a_2 \tag{G.6}$$

(c. f. the definition of chainCrit in Formula (2.1)). We assume the opposite, i. e. there exists some d with all these properties. In fact, as one can easily prove, (G.4), (G.5) and (G.6) can only be satisfied if d fulfills

$$d_1 < b_1 \tag{A.4}$$

$$d_2 < a_2 \tag{A.5}$$

(A.4) together with the definition of b (minimality of b_1) implies now

$$d_1 < x_1 \tag{A.6}$$

Figure 7 illustrates the possible positions of d.

However, the existence of d satisfying both (A.5) and (A.6) contradicts (A.3), as can be proved easily.

Case IV: $x_1 < a_1$ and $a_2 < x_2$.
Analogous to case III. □

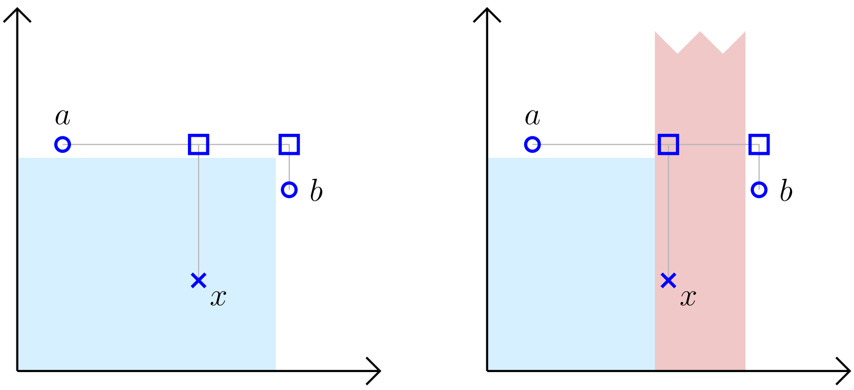

Figure 7: The blue-shaded region is where d might lie, before (left) and after (right) taking into account the definition of b.

Unfortunately, Theorem 8 does *not* hold in arbitrary dimension, as can be seen from the following counterexample for $n = 3$: If $A = \langle(1, 7, 0), (5, 3, 6), (4, 1, 1)\rangle$ and $x = (3, 3, 6)$, then both conditions of the theorem are fulfilled, but

$$\mathrm{M}(A \curvearrowleft x) = 16 \not\leq 14 = \mathrm{M}(A)$$

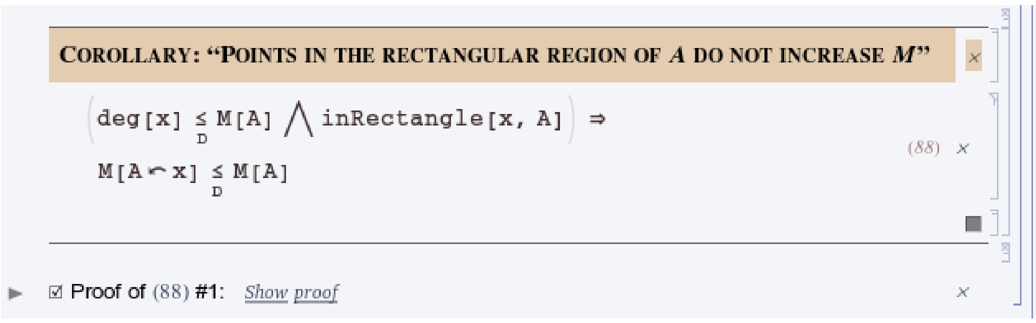

Figure 8: Theorem 8, formalized in *Theorema*

5.5 Bounding $\mathrm{M}(A \curvearrowleft x) + x_k$ if x in Far Exterior

The following theorem is also needed for proving Theorem 10. In contrast to Theorem 8 above, it is *not* "new" in the sense that it is stated and proved in the literature, in [12].

Theorem 9. *If $n = 2$: For all non-empty tuples A of exponent vectors and all exponent vectors x with $\deg(x) \leq \mathrm{M}(A)$ and $\mathrm{inFarExterior}(x, A, k)$ the following inequality holds:*

$$\mathrm{M}(A \curvearrowleft x) + x_k \leq \mathrm{M}(A) + \mathrm{K}(A)_k$$

Proof. A formal, semi-automatically generated proof of Theorem 9 can be found in the *Theorema*-notebook containing the formalization: Formula (*99*) in Section "Theorems" / "Special Case $n = 2$" / "Far Exterior". See also Figure 9. □

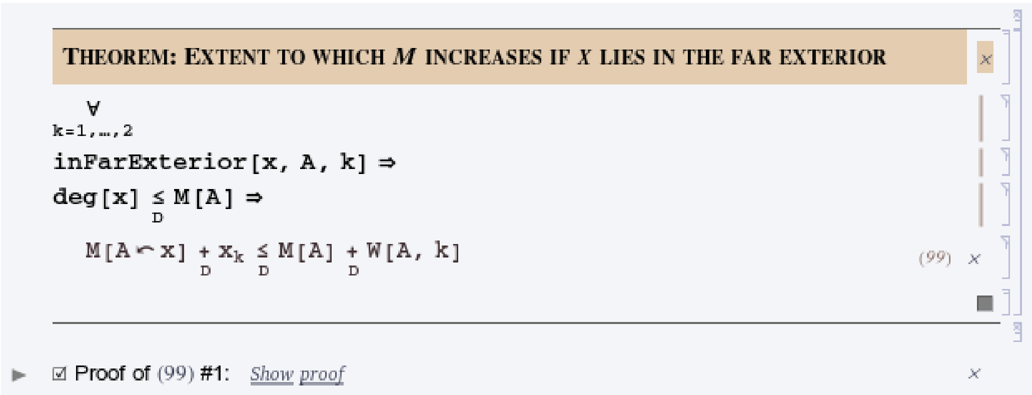

Figure 9: Theorem 9, formalized in *Theorema*

5.6 $\mathrm{M}(A) + \mathrm{W}(A)$ Does Not Increase

The theorem in this subsection is the first of the three main theorems of the whole formalization of the complexity analysis. Note that it follows readily from Lemma 5 and Theorems 8 and 9. Originally, it was proved in [12].

Theorem 10. *If $n = 2$: For all non-empty tuples A of exponent vectors and all exponent vectors x with $\deg(x) \leq \mathrm{M}(A)$ and $\neg\mathrm{isAbove}(x, A)$ the following inequality holds:*

$$\mathrm{M}(A \frown x) + \mathrm{W}(A \frown x) \leq \mathrm{M}(A) + \mathrm{W}(A)$$

Proof. A formal, semi-automatically generated proof of Theorem 10 can be found in the *Theorema*-notebook containing the formalization: Formula (*invariant*) in Section "Main Results" / "$\mathrm{M}[A]+\mathrm{W}[A]$ Does Not Increase". See also Figure 10. □

Going back again to the domain of polynomials, the statement of Theorem 10 is as follows: In Algorithm 1, whenever we have some so-far computed set G (corresponds to A) and we add some new polynomial h (corresponds to x) to it in line 12, yielding the new set G', then $M_{G'} + W_{G'}$ is certainly not greater than

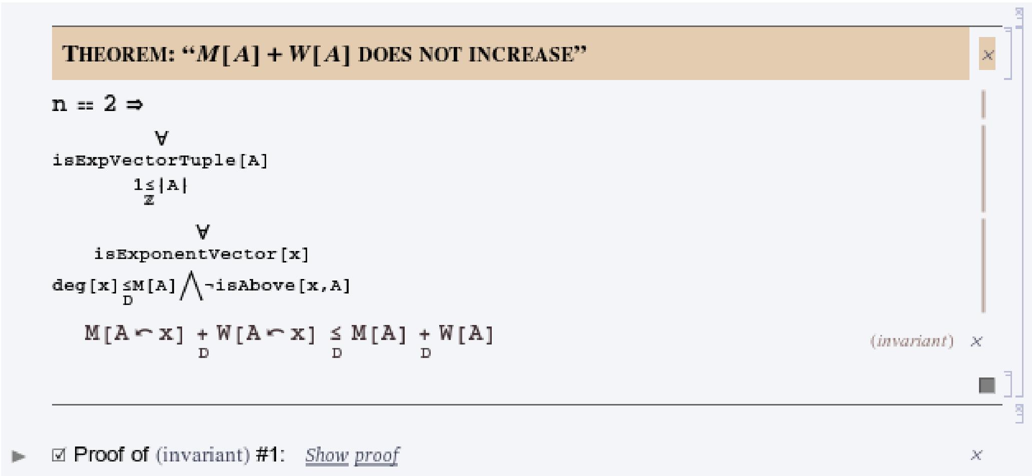

Figure 10: Theorem 10, formalized in *Theorema*

$M_G + W_G$. This we can infer from Theorem 10, since h is not reducible modulo G ($\rightsquigarrow \neg\mathrm{isAbove}(x)$), and the degree of the leading power product of h is not greater than M_G, since h results from reducing the S-polynomial of two polynomials in G for which ChainCrit holds, and a *graded* term ordering is used ($\rightsquigarrow \deg(x) \leq \mathrm{M}(A)$).

5.7 $\mathrm{M}(A) + \mathrm{W}(A) \leq 2\,\mathrm{maxdeg}(A)$

The theorem in this subsection is the second of the three main theorems of the whole formalization of the complexity analysis. Originally it was proved in [4], although a much shorter proof exists (see Section 2.4.4).

Theorem 11. *If $n = 2$: For all non-empty tuples A of exponent vectors the following inequality holds:*

$$\mathrm{M}(A) + \mathrm{W}(A) \leq 2\,\mathrm{maxdeg}(A)$$

Proof. A formal, semi-automatically generated proof of Theorem 11 can be found in the *Theorema*-notebook containing the formalization: Formula (*bound*) in Section "Main Results" / "$\mathrm{M}[A] + \mathrm{W}[A] \leq 2\,\mathrm{maxdeg}[A]$". See also Figure 11. □

5.8 Main Result

Now we are able to state and prove the third and last of the three main theorems of the whole formalization of the complexity analysis, which follows readily from

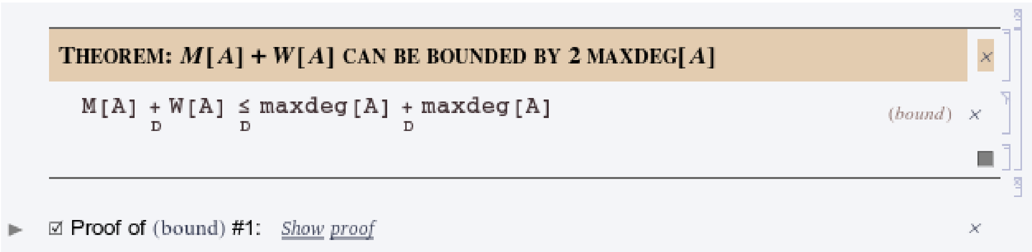

Figure 11: Theorem 11, formalized in *Theorema*

all the previous theorems. It is, essentially, the analogue of Theorem 2 phrased for exponent vectors.

Theorem 12. *If* $n = 2$*: For all non-empty tuples* F *and* G *of exponent vectors with* $\mathrm{M}(G) + \mathrm{W}(G) \leq \mathrm{M}(F) + \mathrm{W}(F)$ *the following inequality holds:*

$$\operatorname{maxdeg}(G) \leq 2 \operatorname{maxdeg}(F)$$

Proof. A formal, semi-automatically generated proof of Theorem 12 can be found in the *Theorema*-notebook containing the formalization: Formula (*main theorem*) in Section "Main Results" / "$\operatorname{maxdeg}[G] \leq 2 \operatorname{maxdeg}[F]$". See also Figure 12. ☐

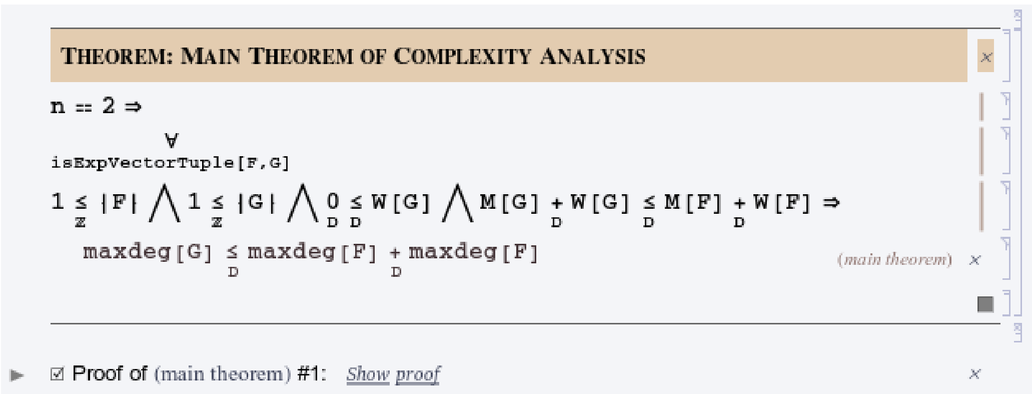

Figure 12: Theorem 12, formalized in *Theorema*

5.9 Summary

The formalization in *Theorema* consists in total of 292 formulas, of which 230 have been proved semi-automatically in *Theorema* with the Complexity Prover described in Section 4. The remaining 62 formulas are definitions that do not require any kind of proofs (including well-definedness proofs). Table 2 lists the numbers of the various kinds of formulas corresponding to the various steps of the Theory Exploration Cycle in more detail.

	Definitions	Lemmas	Theorems
Basic notions	37	70	0
Specific notions	25	127	33
Total	**62**	**197**	**33**

Table 2: Number of formulas in the formalization.

6 Conclusion and Future Work

The work presented in this report shows how non-trivial pieces of mathematics can be formalized and formally verified in an elegant, read- and understandable, but nontheless rigorious way in the *Theorema* system. We are confident that future work on related topics, like the author's PhD thesis on formalizing the fundamentals of Gröbner bases theory (Main Theorem on S-polynomials, correctness of Buchberger's algorithm, ...), will certainly benefit from the insights gained during the formal treatment of the complexity analysis. Below, we list the most important ones:

Checking Conditions of Inference Steps As mentioned already in Section 4.1, inference rules whose applicability is constrained by conditions on the available knowledge in the current proof situation may cause problems regarding the *efficiency* of the prover: Some formulas should be *required* to be known already, whereas others should rather be *proved* in separate subproofs, in order to ensure an efficient proof search. This is a problem not especially related to our complexity analysis, but a problem that we will most likely encounter also in future theory-formalizations with *Theorema*. Therefore, we are currently working on a feasible, *Theorema*-wide solution.

Lifting Functions to Quantifiers The issue of lifting functions to quantifiers was raised in Section 4.2. Similar as with the problem of condition-checking in inferences, also here a *Theorema*-wide solution is desirable, such that it is not necessary to always implement inference rules "by hand" in *Mathematica* when introducing a new higher-order function that gives rise to a quantifier.

Functors The concept of *functors* is essential in the philosophy behind *Theorema*, see for instance [5, 31]. In short, functors can be used for building hierarchies of domains in a structured way, and in fact we could have made use of them in the formalization of the complexity analysis as well: The space of exponent vectors obviously can be regarded a domain (a carrier set together with operations), and domains in *Theorema* are usually defined by means of functors. The reason why we did not pursue this approach is because using a functor for introducing in total only *one* domain in the whole theory would probably unnecessarily complicate the formalization. In a structured, generic formalization of the foundations of Gröbner bases, however, functors will be inevitable.

6.1 Future Work

The work described in this report could be extended in many different ways, including, but not restricted to, the following:

- Consider the *trivariate* case (c. f. [32]), building upon the parts of the present formalization that were already shown for arbitrarily many indeterminates.
- Consider the case of arbitrarily many indeterminates (c. f. [17, 26]).
- Formalize the whole theory of Gröbner bases in *Theorema*, in the same spirit as the complexity analysis. The author's PhD project aims at doing this, at least, for the very foundations of the theory.

Acknowledgements

The author gratefully acknowledges the many valuable discussions about all aspects of the formalization of mathematics and related topics with Bruno Buchberger, and about *Theorema* with *Theorema*'s chief developer Wolfgang Windsteiger.

The research was funded by the Austrian Science Fund (FWF): grant no. W1214-N15, project DK1

References

[1] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero*

Dimensional Polynomial Ideal). PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475–511, 2006.

[2] Bruno Buchberger. A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. In E. W. Ng, editor, *Proceedings of the EUROSAM 79 Symposium on Symbolic and Algebraic Manipulation, Marseille, June 26-28, 1979*, volume 72 of *Lecture Notes in Computer Science*, pages 3–21. Copyright: Springer, Berlin - Heidelberg - New York, 1979.

[3] Bruno Buchberger. A Note on the Complexity of Constructing Gröbner-Bases. In J.A. van Hulzen, editor, *Computer Algebra (Proceedings of EUROCAL 83, European Computer Algebra Conference, London, March 28-30, 1983)*, volume 162 of *Lecture Notes in Computer Science*, pages 137–145. Copyright: Springer- Verlag Berlin - Heidelberg - New York - Tokyo, 1983.

[4] Bruno Buchberger. Miscellaneous Results on Gröbner-Bases for Polynomial Ideals II. Technical Report 83-1, Department of Computer And Information Sciences, University of Delaware, 1983.

[5] Bruno Buchberger. Mathematica as a Rewrite Language. In T. Ida, A. Ohori, and M. Takeichi, editors, *Functional and Logic Programming (Proceedings of the 2nd Fuji International Workshop on Functional and Logic Programming, November 1-4, 1996, Shonan Village Center)*, pages 1–13. Copyright: World Scientific, Singapore - New Jersey - London - Hong Kong, 1996.

[6] Bruno Buchberger. Gröbner Rings in Theorema: A Case Study in Functors and Categories. Technical Report 2003-49, Johannes Kepler University Linz, Spezialforschungsbereich F013, November 2003.

[7] Bruno Buchberger. Proving by First and Intermediate Principles, November 1-2 2004. Invited talk at Workshop on Types for "Mathematics / Libraries of Formal Mathematics", University of Nijmegen, The Netherlands.

[8] Bruno Buchberger. Personal Communication, 2014. RISC, Johannes Kepler University Linz.

[9] Bruno Buchberger, Adrian Craciun, Tudor Jebelean, Laura Kovacs, Temur Kutsia, Koji Nakagawa, Florina Piroi, Nikolaj Popov, Judit Robu, Markus Rosenkranz, and Wolfgang Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 4(4):470–504, 2006.

[10] Bruno Buchberger, Daniela Vasaru, and Tudor Jebelean. The Theorema System: Current Status and the Proving-Solving-Computing Cycle. RISC Report Series 00-37, Research Institute for Symbolic Computation (RISC), Johannes Kepler University of Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, May 2000.

[11] Bruno Buchberger, Wolfgang Windsteiger, et al. *Theorema* – A System for Mathematical Theory Exploration. RISC, Johannes Kepler University Linz. `http://www.theorema.org`.

[12] Bruno Buchberger and Franz Winkler. Miscellaneous Results on the Construction of Gröbner-Bases for Polynomial Ideals. Technical Report 137, Johannes Kepler University, Technisch-Naturwissenschaftliche Fakultaet, Insitut fuer Mathematik, June 1979.

[13] Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.

[14] Thierry Coquand et al. The Coq Proof Assistant. INRIA. `http://coq.inria.fr`.

[15] Thierry Coquand and Henrik Persson. Gröbner Bases in Type Theory. In T. Altenkirch, B. Reus, and W. Naraschewski, editors, *Types for Proofs and Programs (TYPES'98, Kooster Irsee, Germany, March 27-31, 1998)*, volume 1657 of *Lecture Notes in Computer Science*, pages 33–46. Springer Berlin Heidelberg, 1999.

[16] Martin Giese and Bruno Buchberger. Towards Practical Reflection for Formal Mathematics. RISC Report Series 07-05, Research Institute for Symbolic Computation (RISC), University of Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, 2007.

[17] Marc Giusti. Some effectivity problems in polynomial ideal theory. In John Fitch, editor, *International Symposium on Symbolic and Algebraic Computation (EUROSAM 84)*, volume 174 of *Lecture Notes in Computer Science*, pages 159–171. Springer Berlin Heidelberg, 1984.

[18] John Harrison. Metatheory and Reflection in Theorem Proving: A Survey and Critique. Technical report, SRI Cambridge, 1995.

[19] J. Santiago Jorge, Victor M. Guilas, and Jose L. Freire. Certifying properties of an efficient functional program for computing Gröbner bases. *Journal of Symbolic Computation*, 44(5):571–582, May 2009.

[20] Matt Kaufmann, J. Strother Moore, et al. ACL2. University of Texas at Austin. `http://www.cs.utexas.edu/users/moore/acl2/`.

[21] Alexander Maletzky and Bruno Buchberger. Complexity Analysis of the Bivariate Buchberger Algorithm in Theorema. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 41–48, Seoul, Korea, August 5–9 2014. Springer Berlin Heidelberg.

[22] Alexander Maletzky and Bruno Buchberger. Complexity Analysis of the Bivariate Buchberger Algorithm in Theorema [Theorema Notebook]. Technical Report 14-10, RISC, Johannes Kepler University Linz, October 2014.

[23] Ernst W. Mayr and Albert R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, December 1982.

[24] Inmaculada Medina-Bulo, Francisco Palomo-Lozano, José A. Alonso-Jiménez, and José-Luis Ruiz-Reina. Verified Computer Algebra in ACL2 (Gröbner Bases Computation). In B. Buchberger and J.A. Campbell, editors, *AISC 2004*, volume 3249 of *Lecture Notes in Artificial Intelligence*, pages 171–184. Springer-Verlag Berlin Heidelberg, 2004.

[25] Inmaculada Medina-Bulo, Francisco Palomo-Lozano, and Jose-Luis Ruiz-Reina. A verified Common Lisp implementation of Buchberger's algorithm in ACL2. *Journal of Symbolic Computation*, 45(1):96–123, January 2010.

[26] H. Michael Möller and Ferdinando Mora. Upper and lower bounds for the degree of Gröbner bases. In John Fitch, editor, *International Symposium on Symbolic and Algebraic Computation (EUROSAM 84)*, volume 174 of *Lecture Notes in Computer Science*, pages 172–183. Springer Berlin Heidelberg, 1984.

[27] Christoph Schwarzweller. Gröbner Bases – Theory Refinement in the Mizar System. In M. Kohlhase, editor, *Mathematical Knowledge Management (4th International Conference, MKM 2005, Bremen, Germany, July 15-17)*, volume 3863 of *Lecture Notes in Artificial Intelligence*, pages 299–314. Springer Berlin Heidelberg, 2006.

[28] Laurent Thery. A Machine-Checked Implementation of Buchberger's Algorithm. *Journal of Automated Reasoning*, 26:107–137, 2001.

[29] Andrzej Trybulec et al. The Mizar System. University of Bialystok. `http://mizar.uwb.edu.pl/`.

[30] Makarius Wenzel. *The Isabelle/Isar Reference Manual*, August 2014. http://www.cl.cam.ac.uk/research/hvg/Isabelle/index.html.

[31] Wolfgang Windsteiger. Building Up Hierarchical Mathematical Domains Using Functors in THEOREMA. In A. Armando and T. Jebelean, editors, *Electronic Notes in Theoretical Computer Science*, volume 23 of *ENTCS*, pages 401–419. Elsevier, 1999.

[32] Franz Winkler. On the Complexity of the Gröbner-Bases Algorithm over K[x,y,z]. In J. Fitch, editor, *International Symposium on Symbolic and Algebraic Computation (EUROSAM'84)*, pages 184–194, Cambridge, England, July 9-11 1984. Springer-Verlag.

[33] Stephen Wolfram et al. *Mathematica*. Wolfram Research. http://www.wolfram.com/mathematica.

Technical Reports of the Doctoral Program
"Computational Mathematics"

2014

2014-01 E. Pilgerstorfer, B. Jüttler: *Bounding the Influence of Domain Parameterization and Knot Spacing on Numerical Stability in Isogeometric Analysis* February 2014. Eds.: B. Jüttler, P. Paule

2014-02 T. Takacs, B. Jüttler, O. Scherzer: *Derivatives of Isogeometric Functions on Rational Patches* February 2014. Eds.: B. Jüttler, P. Paule

2014-03 M.T. Khan: *On the Soundness of the Translation of MiniMaple to Why3ML* February 2014. Eds.: W. Schreiner, F. Winkler

2014-04 G. Kiss, C. Giannelli, U. Zore, B. Jüttler, D. Großmann, J. Barne: *Adaptive CAD model (re–)construction with THB–splines* March 2014. Eds.: M. Kauers, J. Schicho

2014-05 R. Bleyer, R. Ramlau: *An Efficient Algorithm for Solving the dbl-RTLS Problem* March 2014. Eds.: E. Klann, V. Pillwein

2014-06 D. Gerth, E. Klann, R. Ramlau, L. Reichel: *On Fractional Tikhonov Regularization* April 2014. Eds.: W. Zulehner, U. Langer

2014-07 G. Grasegger, F. Winkler, A. Lastra, J. Rafael Sendra: *A Solution Method for Autonomous First-Order Algebraic Partial Differential Equations* May 2014. Eds.: P. Paule, J. Schicho

2014-08 W. Krendl, W. Zulehner: *A Decomposition Result for Biharmonic Problems and the Hellan-Herrmann-Johnson Method* August 2014. Eds.: U. Langer, V. Pillwein

2014-09 U. Langer, S. Repin, M. Wolfmayr: *Functional a Posteriori Error Estimates for Parabolic Time-Periodic Boundary Value Problems* August 2014. Eds.: V. Pillwein, W. Zulehner

2014-10 A. Maletzky: *Complexity Analysis of the Bivariate Buchberger Algorithm in Theorema* October 2014. Eds.: B. Buchberger, W. Schreiner

2013

2013-01 U. Langer, M. Wolfmayr: *Multiharmonic Finite Element Analysis of a Time-Periodic Parabolic Optimal Control Problem* January 2013. Eds.: W. Zulehner, R. Ramlau

2013-02 M.T. Khan: *Translation of* MiniMaple *to Why3ML* February 2013. Eds.: W. Schreiner, F. Winkler

2013-03 J. Kraus, M. Wolfmayr: *On the robustness and optimality of algebraic multilevel methods for reaction-diffusion type problems* March 2013. Eds.: U. Langer, V. Pillwein

2013-04 H. Rahkooy, Z. Zafeirakopoulos: *On Computing Elimination Ideals Using Resultants with Applications to Gröbner Bases* May 2013. Eds.: B. Buchberger, M. Kauers

2013-05 G. Grasegger: *A procedure for solving autonomous AODEs* June 2013. Eds.: F. Winkler, M. Kauers

2013-06 M.T. Khan *On the Formal Verification of Maple Programs* June 2013. Eds.: W. Schreiner, F. Winkler

2013-07 P. Gangl, U. Langer: *Topology Optimization of Electric Machines based on Topological Sensitivity Analysis* August 2013. Eds.: R. Ramlau, V. Pillwein

2013-08 D. Gerth, R. Ramlau: *A stochastic convergence analysis for Tikhonov regularization with sparsity constraints* October 2013. Eds.: U. Langer, W. Zulehner

2013-09 W. Krendl, V. Simoncini, W. Zulehner: *Efficient preconditioning for an optimal control problem with the time-periodic Stokes equations* November 2013. Eds.: U. Langer, V. Pillwein

The complete list since 2009 can be found at
`https://www.dk-compmath.jku.at/publications/`

Doctoral Program
“Computational Mathematics”

Director:

Prof. Dr. Peter Paule
Research Institute for Symbolic Computation

Deputy Director:

Prof. Dr. Bert Jüttler
Institute of Applied Geometry

Address:

Johannes Kepler University Linz
Doctoral Program “Computational Mathematics”
Altenbergerstr. 69
A-4040 Linz
Austria
Tel.: ++43 732-2468-6840

E-Mail:

office@dk-compmath.jku.at

Homepage:

http://www.dk-compmath.jku.at