Mathematical Theory Exploration in Theorema: Reduction Rings

Alexander Maletzky*

Doctoral Program "Computational Mathematics" and RISC
Johannes Kepler University Linz, Austria
alexander.maletzky@dk-compmath.jku.at

Abstract. In this paper we present the first-ever computer formalization of the theory of Gröbner bases in reduction rings in Theorema. Not only the formalization, but also the formal verification of all key results has already been fully completed by now; this, in particular, includes the generic implementation and correctness proof of Buchberger's algorithm in reduction rings. Thanks to the seamless integration of proving and computing in Theorema, this implementation can now be used to compute Gröbner bases in various different domains directly within the system. Moreover, a substantial part of our formalization is made up solely by "elementary theories" such as sets, numbers and tuples that are themselves independent of reduction rings and may therefore be used as the foundations of future theory explorations in Theorema.
In addition, we also report on two general-purpose Theorema tools we developed for efficiently exploring mathematical theories: an interactive proving strategy and a "theory analyzer" that already proved extremely useful when creating large structured knowledge bases.

Keywords: Gröbner bases, reduction rings, computer-supported theory exploration, automated reasoning, Theorema

1 Introduction

This paper reports on the formalization and formal verification of the theory of reduction rings in Theorema that has recently been completed. Reduction rings, introduced by Buchberger in [3], generalize the domains where Gröbner bases can be defined and algorithmically computed from polynomial rings over fields to arbitrary commutative rings with identity, and may thus become more and more an important tool in computational commutative algebra, just as Gröbner bases in the original setting already are. Since definitions, theorems and proofs tend to be technical and lengthy, we are convinced that our formalization in a mathematical assistant system has the potential to facilitate the further development of the theory in the future (e. g. to non-commutative reduction rings).

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

To the best of our knowledge, reduction rings have never been the subject of formal theory exploration[1] in *any* software system so far; Gröbner bases in polynomial rings over fields have already been formalized in ACL2 [9], Coq and OCaml [6, 15] and Mizar [12], though. Moreover, a formalization in Isabelle by the author of this paper is currently in progress, and the purely algorithmic aspect (no theorems and proofs) of a variation of reduction rings has already been implemented in Theorema in [4]. Theorema is also the software system we chose for our formalization, or, more precisely, Theorema 2.0 (see [5, 18] for an overview and [5] for a brief comparison to other systems). Note that Theorema 2.0 is quite new: it was released only two years ago, in summer 2014, meaning that it still lacks a couple of useful features that are available in many other proof assistants. This, however, was not a reason for not using the system for our work, but just the converse is true: on the one hand, we wanted to demonstrate what *can* be done with Theorema 2.0 already, and on the other hand we wanted to find out what exactly is still missing for effectively and efficiently formalizing mathematics in the system (some of these features have already been implemented in the meantime, see Sect. 5). Besides that, another motivation for using Theorema 2.0 was to formalize a handful of elementary mathematical theories (about sets, numbers, tuples, ...) as well, that may form the foundations of future theory explorations in the system.

The rest of this paper is organized as follows: Section 2 introduces the most important concepts of reduction rings and states the Main Theorem of the theory. Section 3 presents Buchberger's algorithm for computing Gröbner bases in reduction rings as well as its implementation in Theorema, and briefly gives an idea about its correctness proof. Section 4 describes the overall formalization of the theory and its individual components in a bit more detail, and Section 5 presents the interactive proving strategy and the TheoryAnalyzer tool that we developed and already heavily used in the course of the formalization and that will be useful also in future theory explorations. Section 6, finally, summarizes our findings and contains an outlook on future work.

2 Gröbner Bases and Reduction Rings

In this section we review the main concepts of the theory whose formal treatment in Theorema is the content of this paper. To this end, we first give a short motivation of Gröbner bases and reduction rings, and then present the most important definitions and results of the theory. A far more thorough introduction can be found in the literature, e. g. in [1].

Originally, the theory of Gröbner bases was invented for multivariate polynomial rings over fields. There, it can be employed to decide the ideal membership problem, to solve systems of algebraic equations, and many more, and hence is of great importance in computer algebra and many other areas of mathematics, computer science, engineering, etc.

[1] As one reviewer pointed out, *theory exploration* can be understood in several ways. In this paper, we use it as a mere synonym for *formalization of mathematical theories*.

Because of their ability to solve non-trivial, frequently occurring problems in mathematics, it is only natural to try to generalize Gröbner bases from polynomial rings over fields to other algebraic structures. And indeed, nowadays quite some generalizations exist: to non-commutative polynomial rings, to polynomial rings over the integers and other Euclidean- or integral domains, and many more. Reduction rings are a generalization as well, but in a slightly different spirit: in contrast to the other generalizations, reduction rings do not require the domain of discourse to have any polynomial structure. Instead, *arbitrary* commutative rings with identity element may in principle be turned into reduction rings, only by endowing them with some additional structure (see below). It must be noted, however, that not *every* commutative ring with identity can be made a reduction ring; known examples of reduction rings are all fields, the integers, quotient rings of integers modulo arbitrary $n \in \mathbb{N}$ (which may contain zero-divisors!), and polynomial rings over reduction rings.

2.1 Reduction Rings

Reduction rings were first introduced by Buchberger in 1984 [3] and later further generalized by Stifter in the late-1980s [13,14]; our formalization is mainly based on [14]. Here, we only recall the key ideas and main definitions and results of the theory. For this, let in the sequel $\mathcal{R}$ be a commutative ring with identity (possibly containing zero-divisors).

In order to turn $\mathcal{R}$ into a reduction ring, it first and foremost has to be endowed by two additional entities: a function $M : \mathcal{R} \to \mathcal{P}(\mathcal{R})$ that maps every ring element c to a set of ring elements (denoted by M_c) called the *set of multipliers* of c, and a partial Noetherian (i. e. well-founded) order relation $\preceq$. With these ingredients it is possible to introduce the crucial notion of reduction rings, namely that of *reduction*:

Definition 1 (Reduction). *Let $C \subseteq \mathcal{R}$. The reduction relation modulo C, denoted by $\to_C$, is a binary relation on $\mathcal{R}$ such that $a \to_C b$ iff $b \prec a$ and there exists some $c \in C$ and some $m \in M_c$ such that $b = a - m\,c$.*
As usual, $\to_C^*$ and $\leftrightarrow_C^*$ denote the reflexive-transitive- and the symmetric-reflexive-transitive closure of $\to_C$, respectively. Moreover, for a given $z \in \mathcal{R}$, a and b are said to be connectible below *z, denoted by $a \leftrightarrow_C^{\prec z} b$, iff $a \leftrightarrow_C^* b$ and all elements in the chain between a and b are strictly less than z (w. r. t. $\preceq$).*

Of course, the function M and the relation $\preceq$ cannot be chosen arbitrarily but, together with the usual ring operations, have to satisfy certain non-trivial constraints, the so-called *reduction ring axioms*. In total, there are 14 of them, with some being quite simple (0 must be the least element w. r. t. $\preceq$, for instance), others are extremely technical. The complete list underlying our formalization is omitted here because of space limitations but can be found in [7].

Example 1. In a field K, suitable definitions of M_c and $\preceq$ are $M_c := K \backslash \{0\}$ and $x \preceq y :\Leftrightarrow x = 0$. In $\mathbb{Z}_n = \mathbb{Z}/n\,\mathbb{Z}$, represented as $\{[0]_n, \ldots, [n-1]_n\}$,

we have $M_{[c]_n} := \{[1]_n, \ldots, [k]_n, [n-k]_n, \ldots, [n-1]_n\}$, where k is the least positive integer such that $[c\,(k+1)]_n = [0]_n$; the ordering is simply defined as $[x]_n \preceq [y]_n :\Leftrightarrow x < y$.

In polynomial rings, finally, matters are a bit more complicated. There, the sets M_c and the ordering $\preceq$ not only depend on the respective objects in the coefficient ring, but also on an admissible *term order* [11] on the set of all power-products.

Note that in reduction rings $\leftrightarrow_C^*$ coincides with the congruence relation modulo the ideal generated by C. Hence, if it is possible to decide $\leftrightarrow_C^*$, then the ideal membership problem could effectively be solved—and this is where Gröbner bases come into play.

2.2 Gröbner Bases

We can start with the definition of Gröbner bases in reduction rings right away:

Definition 2 (Gröbner basis). *Let $G \subseteq \mathcal{R}$. Then G is called a* Gröbner basis *iff G is finite and $\rightarrow_G$ is Church-Rosser, i. e. whenever $a \leftrightarrow_G^* b$ there exists a common successor s with $a \rightarrow_G^* s$ and $b \rightarrow_G^* s$.*
For $C \subseteq \mathcal{R}$, G is called a Gröbner basis of C iff it is a Gröbner basis and $\langle G\rangle$ (i. e. the ideal generated by G over $\mathcal{R}$) is the same $\langle C\rangle$.

If reduction can effectively be carried out, i. e. whenever a is reducible modulo C then some b with $a \rightarrow_C b$ can be computed, and for any given $C \subseteq \mathcal{R}$ a Gröbner basis G of C exists and can be computed, then the problem of deciding membership in $\langle C\rangle$ can be solved: a given candidate a simply has to be totally reduced modulo G until an irreducible element h is obtained; then $a \in \langle C\rangle$ iff $h = 0$.

The axioms of reduction rings ensure that for every $C \subseteq \mathcal{R}$ a Gröbner basis does not only exist, but can even be effectively computed (see Section 3). This key result is based on the following

Theorem 1 (Buchberger's Criterion). *Let $G \subseteq \mathcal{R}$ finite. Then G is a Gröbner basis iff for all $g_1, g_2 \in G$ (not necessarily distinct) and all* minimal non-trivial common reducibles *z of g_1 and g_2, we have $a_1 \leftrightarrow_G^{\prec z} a_2$, where $z \rightarrow_{\{g_i\}} a_i$ for $i = 1, 2$. (a_1, a_2) is called a* critical pair *of g_1 and g_2 w. r. t. z.*

The precise definition of *minimal non-trivial common reducible* (mntcr) is slightly technical and omitted here; the interested reader may find it in the referenced literature. Intuitively, a mntcr of g_1 and g_2 is an element that can be reduced both modulo $\{g_1\}$ and modulo $\{g_2\}$ in a *non-trivial* way.

Example 2. In a field K, the set of mntcrs of any two non-zero field elements is just $K\backslash\{0\}$. In $\mathbb{Z}_n$, the only mntcr of two non-zero elements $[c]_n$ and $[d]_n$ is $[\max\{\gcd(c,n), \gcd(d,n)\}]_n$. In $\mathcal{R}[X]$, the mntcrs of two non-zero polynomials p and q are all monomials of the form $c\,\tau$, where c is a mntcr of the leading coefficients of p and q in $\mathcal{R}$ and τ is the least common multiple of the leading power-products of p and q, w. r. t. the chosen term order.

Example 3. Let us consider $\mathbb{Z}_{24}[x, y]$ and the singleton $C := \{p := 16xy+2\}$ (we write 16 and 2 instead of $[16]_{24}$ and $[2]_{24}$, respectively, for the sake of brevity). No matter which term order we choose, the leading power-product of p is xy and its leading coefficient is 16, meaning that the only mntcr of p and p is $\gcd(16, 24)xy = 8xy$. Reducing $8xy$ modulo p once (in two different ways) yields the critical pair $(8xy-2(16xy+2), 8xy-17(16xy+2)) = ([20]_{24}, [14]_{24})$. Neither of the two constituents of the critical pair can be reduced further modulo C, meaning that the critical pair cannot be connected below $8xy$, and hence C is no Gröbner basis.

2.3 Contributions to the Theory

Before moving on to Buchberger's algorithm, we want to point out two contributions we managed to make to the theory of reduction rings itself. Namely, during the formalization, when turning to the computer-assisted verification of the results, we discovered two problems in the literature on reduction rings. The first of these problems is related to the notion of *irrelativity* as introduced in [14]: without going into details here, irrelativity basically is a binary relation on the set of all elements of a reduction ring, which clearly ought to be symmetric. Irrelativity according to [14], however, is *not* symmetric, and a close look at the proofs of the main results revealed that they contain a very subtle error mainly because of that reason. Therefore, the definition of irrelativity had to be adjusted in order to proceed with the formal verification, which we finally managed to do. More details can be found in [7].

The second problem concerns fields as reduction rings: in an infinite field, two elements have *infinitely many* mntcrs (see Ex. 2), although for an algorithmic treatment one axiom of reduction rings requires the number of mntcrs to be finite. Although this problem was already known in [3], no attempts have been made to fix it so far. We solved it by introducing an equivalence relation in reduction rings and weakening said axiom to require only the number of *equivalence classes* of mntcrs to be finite.

3 Buchberger's Algorithm

Theorem 1 not only yields a finite criterion for checking whether a given set G is a Gröbner basis or not, but it even gives rise to an algorithm for actually *computing* Gröbner bases. This algorithm, presented in Fig. 1, is a critical-pair/completion algorithm that, given an input set $C \subseteq \mathcal{R}$, basically checks the criterion of Thm. 1 for all pairs of elements of C, and if it fails for a pair (C_i, C_j), then C is *completed* by a new element h that makes the criterion hold for (C_i, C_j). Of course, afterward all pairs involving the new element h have to be considered as well.

Figure 1 presents the algorithm as implemented in a functional style in Theorema. Function `GB` is the main function that takes as input the tuple[2] C a

[2] `GB` is implemented for tuples rather than sets, for practical reasons.

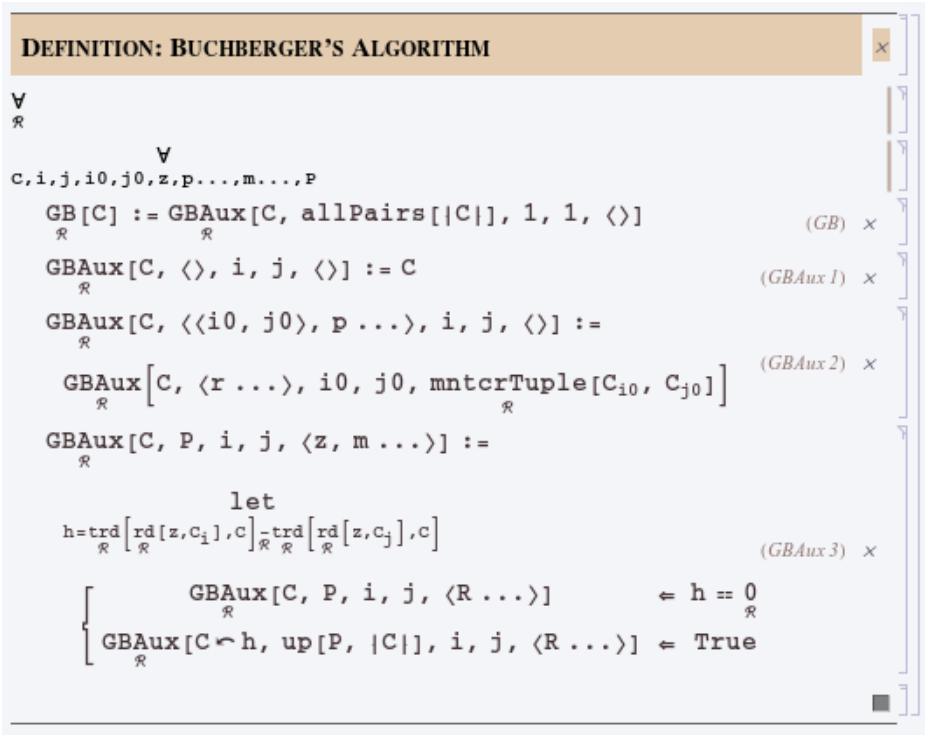

Fig. 1. Buchberger's algorithm in Theorema.

Gröbner basis shall be computed for. It then calls `GBAux` with suitable initial arguments, whose first argument serves as the accumulator of the tail-recursive function. Its second argument is the tuple of all pairs of indices of C that have not been dealt with yet, and its third and fourth arguments are the indices i and j of the elements currently under consideration. The last argument, finally, is the tuple of all mntcrs of C_i and C_j that still have to be checked. Formula (GBAux 3) is the crucial one: The constituents of the critical pair originating from C_i and C_j and mntcr z are totally reduced modulo the current basis C, and the difference is assigned to h. If $h = 0$, the critical pair can be connected below z according to the condition in Thm. 1, so nothing else has to be done in this case. Otherwise, h is added to C, ensuring connectibility below the new basis, and the index-pair-tuple is updated to include also the pairs involving the new element h.

Buchberger's algorithm, or, more precisely, function `GB`, can be proved to behave according to the following specification:

> If $\mathcal{R}$ is a reduction ring and C is a tuple of elements of $\mathcal{R}$, `GB` terminates and returns again a tuple G of elements of $\mathcal{R}$. G is a Gröbner basis of C.

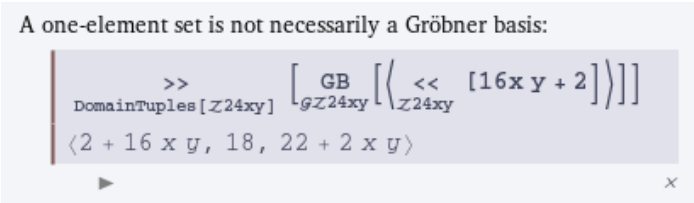

Fig. 2. A sample computation in Theorema. The "<<" and ">>" are only responsible for the in- and output of polynomials and do not affect the actual computation.

The proof of this claim was carried out formally in Theorema. It heavily depends on Thm. 1, of course, but also quite some other technicalities (concerning the indices, for instance) have to be taken into account. Furthermore, termination of `GBAux` is by no means obvious: its second argument, which must eventually become empty, is enlarged in the second case of (GBAux 3), meaning that this case must be shown to occur only finitely often. A separate reduction ring axiom is needed to ensure this.

Function `GB` is not only of theoretical interest for our formalization, but can also be executed on concrete input to actually compute Gröbner bases, provided that the underlying domain $\mathcal{R}$ is a reduction ring and implements a couple of auxiliary functions `GB` depends upon (most importantly, the usual ring operations). At the moment, the following domains included in the formalization meet these requirements; the proofs thereof are part of the formalization, of course (see also Sect. 4.2):

- all fields, in particular the Theorema built-in fields $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$,
- $\mathbb{Z}$,
- $\mathbb{Z}_n$ for arbitrary $n \in \mathbb{N}$,
- multivariate polynomial rings over the aforementioned domains.

Function `GB` always returns provenly correct results when used in these domains. Figure 2 shows a sample computation in $\mathbb{Z}_{24}[x, y]$, carried out directly within Theorema 2.0: as discussed in Ex. 3, $\{16xy + 2\}$ is no Gröbner basis, because the constituents of the critical pair $([20]_{24}, [14]_{24})$ cannot be connected. Therefore, their difference $[14 - 20]_{24} = [18]_{24}$ must be added to the basis in a first step. Figure 2 reveals that this is still not sufficient, since one further element must be added afterward.

For the sake of completeness we have to point out that Buchberger's algorithm and Thm. 1 as presented here were simplified a bit compared to our actual formalization. For one thing, the sets of multipliers M_c have to be split into several (finitely many) indexed subsets M_c^i, and the notion of mntcr depends on these indices; mntcrs for *all* pairs of indices have to be considered separately, both in the theorem and in the algorithm. Also, the actual implementation of `GB` employs the so-called *chain criterion* for avoiding useless reductions; this criterion, hence, increases efficiency and works in reduction rings in pretty much the

same way as in the original setting of polynomials over fields, see [2]. The interested reader is referred to [7] for an unsimplified statement of Thm. 1, and to [8] for a more detailed discussion of Buchberger's algorithm in our formalization.

4 Structure of the Formalization

In this section we have a closer look at the formalization of all of reduction ring theory in Theorema. In particular, the emphasis is on how the theory is split into smaller sub-theories, what these sub-theories consist of, how they are related to each other, and how big they are in terms of formulas and proofs.

Before, however, some remarks on theory exploration in Theorema 2.0 *in general* are in place. Theorema theories are essentially *Mathematica* notebooks consisting of both formal (mathematical formulas) and informal (explanatory text, diagrams, tables, etc.) content. Users are free to compose such notebooks in whatever way they want, making use of *Mathematica*'s rich typesetting capabilities, yielding nicely-formatted documents. Proving proceeds by first setting up *proof tasks* and then either calling an automatic prover or an interactive proof strategy (see Sect. 5). In any case, the resulting proofs are stored as abstract *proof objects* in external files; they can be inspected in automatically generated *proof documents* displaying the proofs in a human-readable form that closely resembles the way how proofs are usually presented in mathematical text-books (again, heavily relying on *Mathematica*'s typesetting capabilities). Since this paper does not aim at presenting Theorema 2.0, and in particular how theory exploration in the system proceeds, in detail, the interested reader is referred to our recent article [5] instead.

Although the paper has only been about reduction rings so far, it must be noted that a substantial part of our formalization is actually concerned with rather basic concepts, such as sets, algebraic structures, numbers, tuples (or lists) and sequences that are themselves independent of reduction ring theory and merely serve as its logical backbone. In this respect, our formalization can also be regarded a major contribution to a structured knowledge base of elementary mathematical theories in Theorema 2.0 that can be reused in future theory explorations. Such a knowledge base did not exist in Theorema 2.0 before, which justifies, in our opinion, presenting it just alongside the formal treatment of reduction rings in this section (only superficially, though).

Figure 3 shows the dependencies of the individual sub-theories on each other. Each node represents a sub-theory, contained in a separate Theorema notebook, and a directed edge from theory A to theory B means that B logically depends on A in the sense that formulas (i. e. definitions or theorems) contained in A were used in the proof of a theorem in B. Theories corresponding to framed nodes are directly related to reduction rings (see Sect. 4.2), whereas all other theories belong to the knowledge base of elementary theories (see Sect. 4.1). Note also that transitive edges are omitted for better readability, e. g. theory Numbers.nb not only depends *indirectly* on theory LogicSets.nb (via AlgebraicStructures.nb), but also *directly*; this fact is not reflected in Fig. 3.

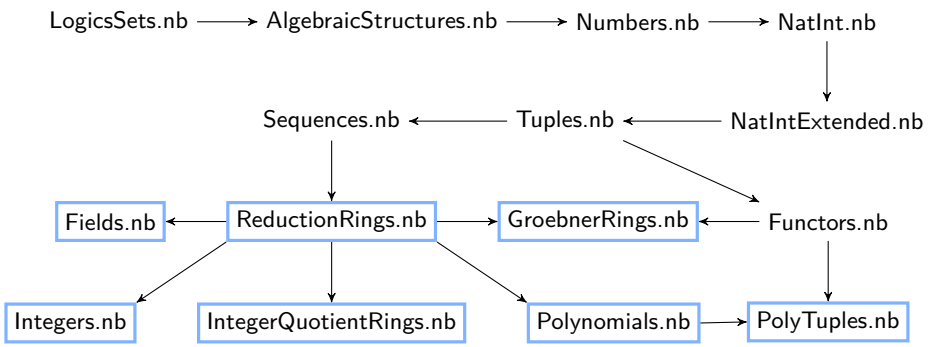

Fig. 3. The theory dependency graph.

The total number of proved theorems in the whole formalization in 2464, the total number of unproved definitions and axioms is 484. Hence, the total number of formulas is **2948**. The complete formalization is available online from `http://www.risc.jku.at/people/amaletzk/Formalizations.html`.

4.1 Elementary Theories

Most of the sub-theories in this category have rather self-explanatory names, and we will not go into details regarding their contents. Some remarks are still in place, though.

Theories Numbers.nb, NatInt.nb and NatIntExtended.nb are all about natural numbers and integers: the very definition of natural numbers by purely set-theoretic means, as well as the definition of integers as some quotient domain of pairs of natural numbers are contained in Numbers.nb, and the other two theories basically consist of hundreds of results about linear and non-linear arithmetic, division with quotient and remainder, the greatest common divisor, finite sums and mappings from $\mathbb{N}$ to $\mathbb{N}$ (needed for infinite sequences).

Theory Functors.nb contains a couple of general Theorema functors, mainly for constructing product domains from given ones.[3] The most important functor in this theory, `LexOrder`, maps two ordered domains to their lexicographic product; this functor was needed for proving termination of function `GB` (see Sect. 3). Functors.nb also proves that the order in the new domain is still partial/total/Noetherian if the orders in the original domains are.

4.2 Reduction Ring Theory

ReductionRings.nb contains the definitions of several auxiliary notions in reduction rings, like reducibility, the reduction relation (and its various closures) and properties of binary relations (confluence, local confluence, Church-Rosser), as

[3] For information on functors and domains in Theorema, see [4, 17]

well as the definitions of reduction rings and Gröbner bases. Reduction rings are defined through a unary predicate, `isReductionRing`, that is simply the conjunction of all reduction ring axioms together with the axioms of commutative rings with identity.

Besides these definitions, the main contents of ReductionRings.nb are the Main Theorem of reduction ring theory, Thm. 1, and the theorem that states that the symmetric-reflexive-transitive closure of the reduction relation modulo a set C coincides with ideal congruence modulo the same set C, together with their proofs. The proof of Thm. 1 is non-trivial and lengthy, which is reflected by the fact that many auxiliary lemmas were needed before it could finally be completed, and one of these lemmas in fact deserves special attention: the *Generalized Newman Lemma*. The Generalized Newman Lemma is a general result about sufficient conditions for binary relations to be confluent (and thus Church-Rosser) that was first introduced in [19].

Please note that everything in this theory is *non-algorithmic* in the sense that no single algorithm is implemented or specified. All algorithmic aspects of our formal reduction ring theory, in particular Buchberger's algorithm for computing Gröbner bases, are part of GroebnerRings.nb.

GroebnerRings.nb contains all the algorithmic aspects of the formalization, like the implementation and specification of Buchberger's algorithm. More precisely, the theory contains a functor called `GroebnerRing` that extends a given input domain D by the function `GB` that implements Buchberger's algorithm and can thus be used for computing Gröbner bases. `GB` is defined in terms of auxiliary functions provided by the underlying domain D, such as the basic ring operations and the partial Noetherian ordering in reduction rings. However, following a general principle of functors and domains in Theorema, D can be completely arbitrary: it does not need to be a reduction ring, nor even a ring, meaning that some operations used in function `GB` are possibly undefined – and this is perfectly fine, except that one cannot expect to obtain a Gröbner basis when calling the function. But if D *is* a reduction ring, i. e. `isReductionRing[`D`]` holds, then the function really behaves according to its specification. The proof of this claim is non-trivial, even if Thm. 1 is already known, and also contained in GroebnerRings.nb.

In addition to the implementation, specification and correctness proof of Buchberger's algorithm, various sample computations of Gröbner bases in different domains ($\mathbb{Z}_{24}$, $\mathbb{Z}_{24}[x, y]$, $\mathbb{Q}[x, y, z]$, for instance) are included in GroebnerRings.nb as well.

Fields.nb contains a Theorema functor, `ReductionField`, that takes an input domain K and extends it by those objects (function M and relation $\preceq$) that turn K into a reduction ring. These new objects are defined in such a way that if K is a field, then the extension really *is* a reduction ring – otherwise nothing can be said about it. The proof of this claim is of course also contained in Fields.nb, and actually it is quite straight-forward.

Integers.nb contains a Theorema functor, `ReductionIntegers`, that does not take any input domains but simply constructs a new domain whose carrier is $\mathbb{Z}$ and that provides the additional objects for turning $\mathbb{Z}$ into a reduction ring, following [3]. The proof of this claim is included in the theory as well.

IntegerQuotientRings.nb contains a Theorema functor, `ReductionIQR`, that takes a positive integer n and constructs a new domain whose carrier is the set $\{0, \ldots, n-1\}$ and that provides the additional objects for turning $\mathbb{Z}_n$, represented by $\{0, \ldots, n-1\}$, into a reduction ring, following [13]. The proof of this claim is of course included in the theory as well. Surprisingly, although turning $\mathbb{Z}_n$ into a reduction ring is more involved than $\mathbb{Z}$[4], fewer auxiliary results were needed in IntegerQuotientRings.nb than in Integers.nb. This is due to the fact that the reduction ring ordering $\preceq$ in $\mathbb{Z}_n$ is much simpler than in $\mathbb{Z}$.

Polynomials.nb contains the general result that the n-variate polynomial ring over a reduction ring is again a reduction ring, if the sets of multipliers and the order relation are defined appropriately. This is accomplished by first introducing the class of *reduction polynomial domains* over a coefficient domain $\mathcal{R}$ and a power-product domain $\mathcal{T}$. A domain $\mathcal{P}$ belongs to this class iff it provides the usual ring operations, a coefficient function that maps each power-product from $\mathcal{T}$ to a coefficient in $\mathcal{R}$, a set of multipliers for each element in $\mathcal{P}$ (i. e. the function M), and an order relation $\preceq$, and all these objects satisfy certain constraints (e. g. the coefficient function must have finite support and must interact with $+$ and $\cdot$ in the usual way, the sets of multipliers must be of a particular form, and the ordering must be defined in a certain way). These constraints, whose precise formulations can be found in [3], ensure that if $\mathcal{R}$ is a reduction ring and $\mathcal{T}$ is a domain of commutative power-products, then $\mathcal{P}$ is a reduction ring as well. This is one of the fundamental results of reduction ring theory, and its proof is very complicated and tedious (even more complicated than the proof of Thm. 1). Nevertheless, it has been entirely completed already and is also part of Polynomials.nb.

Note that all definitions and results in this theory are on a very abstract level: no concrete representation of multivariate polynomials, be it as tuples of monomials, as iterated univariate polynomials, or whatsoever, is ever mentioned in the whole theory, but instead polynomials are essentially viewed as functions from $\mathcal{T}$ to $\mathcal{R}$ with finite support. This approach has the advantage that the results can easily be specialized to many *different* representations of polynomials, if necessary, and this is just what is made use of in theory PolyTuples.nb.

PolyTuples.nb contains a functor, `PolyTuples`, that takes two domains $\mathcal{R}$ and $\mathcal{T}$ as input and constructs the domain $\mathcal{P}$ of reduction-polynomials over coefficient domain $\mathcal{R}$ and power-product domain $\mathcal{T}$ represented as ordered (w. r. t. the ordering on $\mathcal{T}$) tuples of monomials. Monomials, in turn, are represented as pairs of coefficients and power-products. $\mathcal{P}$ provides the additional functions and

[4] The first attempt in [3] was erroneous.

relations needed to prove that it belongs to the class of reduction polynomial domains, and thus is a reduction ring thanks to the key result in Polynomials.nb.[5] The proof of this claim is part of the theory, of course.

Besides functor `PolyTuples`, three additional functors for constructing domains of commutative power-products are also contained in PolyTuples.nb: one for a purely lexicographic term order, one for a degree-lexicographic term order, and one for a degree-reverse-lexicographic term order. In either case, power-products are represented as tuples of natural numbers.

5 New Tools

In this section we present two useful tools that we developed in the course of the formalization of reduction rings: an interactive proof strategy and a mechanism for analyzing the logical structure of Theorema theories. As will be seen in the following two subsections, the tools are general-purpose tools and thus completely independent of our concrete formalization, and hence may be used in any other theory exploration in Theorema as well. For that reason, they are planned to be integrated into the official version of the system in the near future.

5.1 Interactive Proof Strategy

In contrast to most other proof assistants, the interactive proof strategy in Theorema 2.0 described below is not text-based, but *dialog-oriented* (similar to the one in Theorema 1 [10]): whenever a new proof situation that cannot be handled automatically[6] arises during the proof search, a dialog window pops up. This window displays the current proof situation, characterized by the current proof goal and the current set of assumptions, and asks the user how to proceed. He may now either

- choose an inference rule to apply,
- choose a different pending proof situation where to continue with the proof search,
- inspect the proof *so far*, in a nicely-formatted proof document,
- inspect the internal representation of the proof object for debugging,
- save the current status of the proof in an external file,
- adjust the configuration of the prover (maybe even switching from the interactive mode to a fully automatic one), or
- abort the proof attempt.

When choosing an inference rule that shall be applied (or, more precisely, *tried*), the user even has the possibility to indicate the formula(s) to be considered by the rule (for instance, if one of several universally quantified assumptions is to

[5] Once again, this is only true if $\mathcal{R}$ is a reduction ring and $\mathcal{T}$ is a domain of commutative power-products.

[6] So, there is still *some* automation of very trivial tasks.

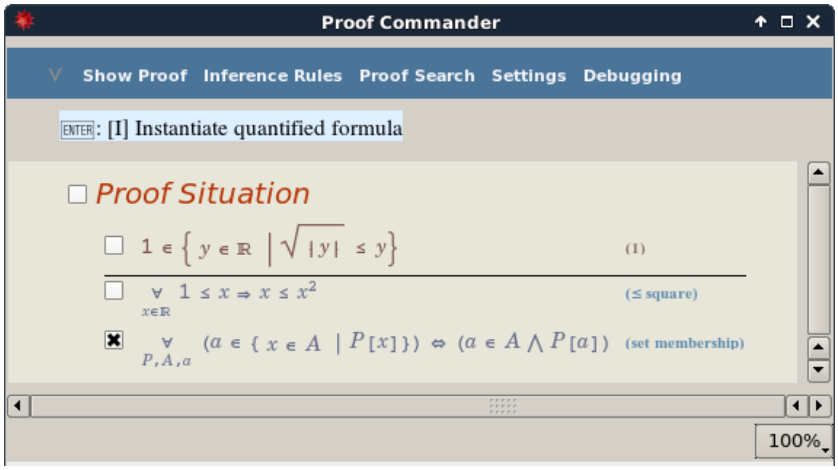

Fig. 4. A "Proof Commander" dialog window for interactive proving.

be instantiated). Furthermore, he may then be asked to provide further information about the concrete application of the rule (like specifying the concrete term a formula shall be instantiated with); this, however, solely depends on the implementation of the inference rule and is thus not affected by our interactive proof strategy.

Figure 4 shows a screen-shot of the interactive dialog window. In the middle, the current goal (top) and the current assumptions (bottom) are displayed. Above, the inference rule to be applied next, as chosen by the user, is indicated, and the menu bar is located at the very top.

5.2 TheoryAnalyzer

The TheoryAnalyzer is a *Mathematica* package that provides a collection of functions for analyzing the logical structure of Theorema theories and the logical dependencies of formulas on each other. If theories grow big, as in our case, it becomes more and more difficult to keep track of which formulas were used in the proofs of which other formulas, which formulas are affected when another formula is modified, and whether the order of formulas in a notebook agrees with their logical order. It is clear, however, that these questions are of utmost importance for a consistent, coherent and systematic development of a mathematical theory; after all, if a formula φ is modified, then all of its consequences (that is, the theorems that use φ as an assumption in their proofs) *must* be re-proved, and so one needs to know what these consequences are in the first place—and this was the main motivation for the development of the TheoryAnalyzer.

Summarizing, the TheoryAnalyzer allows to automatically

- inspect all direct or indirect assumptions of a given theorem,
- inspect all direct or indirect consequences of a given formula,

– ensure that theories do not contain circular arguments,
– check whether the order of formulas in a notebook agrees with their logical order, and
– draw nicely-formatted theory-dependency-graphs (as the one in Fig. 3) and formula-statistics-diagrams.

6 Conclusion

The work described in this paper is expected to have, and already had, various positive effects on theory explorations in Theorema 2.0 and on reduction ring theory: the existing formalization, in particular of the elementary mathematical theories, may serve as the basis of future theory explorations, perhaps even in completely different areas of mathematics. The tools presented in Sect. 5 proved extremely useful already and will definitely be of use for other users as well, once they are integrated into the system. And, finally, the contributions to the theory of reduction rings mentioned in Sect. 2.3 give evidence to the claim that mathematics profits from being treated formally in computer systems.

There are many possibilities for future work. On the theory level, other aspects of, and approaches to, Gröbner bases (again in the original setting) could be formalized, for instance the computation of Gröbner bases by matrix triangularizations [16]. For this, the further improvement of the tools described in Sect. 5 and the development of new tools might be necessary (more flexible interactive proving strategy, proof checker, ...).

Acknowledgments. I thank the anonymous referees for their valuable remarks and suggestions.

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

References

1. Adams, W.W., Loustaunau, P.: An Introduction to Gröbner Bases. American Mathematical Society (1994)
2. Buchberger, B.: A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. In: Ng, E.W. (ed.) Proceedings of the EUROSAM'79 Symposium on Symbolic and Algebraic Manipulation, Marseille, June 26-28, 1979. Lecture Notes in Computer Science, vol. 72, pp. 3–21. Springer-Verlag (1979)
3. Buchberger, B.: A Critical-Pair/Completion Algorithm for Finitely Generated Ideals in Rings. In: Börger, E., Hasenjaeger, G., Rödding, D. (eds.) Logic and Machines: Decision Problems and Complexity (Proceedings of the Symposium "Rekursive Kombinatorik", Münster, Germany, May 23–28). Lecture Notes in Computer Science, vol. 171, pp. 137–161. Springer-Verlag (1984)
4. Buchberger, B.: Gröbner Rings in Theorema: A Case Study in Functors and Categories. Tech. Rep. 2003-49, Johannes Kepler University Linz, Spezialforschungsbereich F013 (November 2003)

5. Buchberger, B., Jebelean, T., Kutsia, T., Maletzky, A., Windsteiger, W.: Theorema 2.0: Computer-Assisted Natural-Style Mathematics. Journal of Formalized Reasoning 9(1), 149–185 (2016)
6. Jorge, J.S., Guilas, V.M., Freire, J.L.: Certifying properties of an efficient functional program for computing Gröbner bases. Journal of Symbolic Computation 44(5), 571–582 (2009)
7. Maletzky, A.: Exploring Reduction Ring Theory in Theorema. Tech. Rep. 2016-06, Doctoral Program "Computational Mathematics", Johannes Kepler University Linz, Austria (July 2015)
8. Maletzky, A.: Verifying Buchberger's Algorithm in Reduction Rings. In: Jebelean, T., Wang, D. (eds.) Proceedings of PAS'2015 (Program Verification, Automated Debugging and Symbolic Computation, Beijing, China, October 21–23 (2015), final version available from `arXiv:1604.08736` [cs.SC]
9. Medina-Bulo, I., Palomo-Lozano, F., Ruiz-Reina, J.L.: A verified Common Lisp implementation of Buchberger's algorithm in ACL2. Journal of Symbolic Computation 45(1), 96–123 (2010)
10. Piroi, F., Kutsia, T.: The Theorema Environment for Interactive Proof Development. In: Sutcliffe, G., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence, and Reasoning (Proceedings of the 12th International Conference, LPAR'05). Lecture Notes in Artificial Intelligence, vol. 3835, pp. 261–275. Springer-Verlag (2005)
11. Robbiano, L.: Term orderings on the polynomial ring. In: Caviness, B.F. (ed.) EUROCAL'85 (European Conference on Computer Algebra, Linz, Austria, April 1–3). Lecture Notes in Computer Science, vol. 204, pp. 513–517. Springer-Verlag (1985)
12. Schwarzweller, C.: Gröbner Bases – Theory Refinement in the Mizar System. In: Kohlhase, M. (ed.) Mathematical Knowledge Management (4th International Conference, MKM 2005, Bremen, Germany, July 15-17). Lecture Notes in Artificial Intelligence, vol. 3863, pp. 299–314. Springer-Verlag (2006)
13. Stifter, S.: A Generalization of Reduction Rings. Journal of Symbolic Computation 4(3), 351–364 (1988)
14. Stifter, S.: The Reduction Ring Property is Hereditary. Journal of Algebra 140(89-18), 399–414 (1991)
15. Thery, L.: A Machine-Checked Implementation of Buchberger's Algorithm. Journal of Automated Reasoning 26, 107–137 (2001)
16. Wiesinger-Widi, M.: Gröbner Bases and Generalized Sylvester Matrices. Ph.D. thesis, Johannes Kepler University Linz (2015), available online from `http://epub.jku.at/obvulihs/content/titleinfo/776913`
17. Windsteiger, W.: Building Up Hierarchical Mathematical Domains Using Functors in Theorema. In: Armando, A., Jebelean, T. (eds.) Proceedings of Calculemus'99, Trento, Italy. Electronic Notes in Theoretical Computer Science, vol. 23, pp. 401–419. Elsevier (1999)
18. Windsteiger, W.: Theorema 2.0: A System for Mathematical Theory Exploration. In: Yap, C., Hong, H. (eds.) Mathematical Software – ICMS 2014 (Proceedings of ICMS'2014, August 5–9, Seoul, Korea). Lecture Notes in Computer Science (LNCS), vol. 8592, pp. 49–52 (2014)
19. Winkler, F., Buchberger, B.: A Criterion for Eliminating Unnecessary Reductions in the Knuth-Bendix Algorithm. In: Colloqium on Algebra, Combinatorics and Logic in Computer Science. pp. 849–869 (1983)