# Algebraic Requirements for the Construction of Polynomial Rings

*Robert Milewski*[†1]    *Christoph Schwarzweller*[††2]

[†]University of Bialystok
Institute of Computer Science
Sosnowa 64, 15-614 Bialystok, Poland
milewski@cksr.ac.bialystok.pl

[††]University of Tuebingen
Faculty for Computer Science
Sand 13, D-72076 Tuebingen, Germany
schwarzw@informatik.uni-tuebingen.de

**Abstract** - The Mizar construction of polynomials with an arbitrary number of variables has been described in [8]. In this paper, we present an alternative approach for formalizing polynomials with one variable. This approach has the advantage that quite a number of theorems can be proven with fewer requirements on the underlying coefficient ring. In addition, because polynomials with only one variable can be represented more easily, the construction allows for more straightforward proofs.

**Keywords** – formalized mathematics, Mizar, polynomial ring, evaluation homomorphism, algebraic requirements.

## 1. Introduction

The ring of polynomials is usually constructed over a commutative ring with $1 \neq 0$ or even more specifically, over a field, that is, the set $L$ of coefficients for the polynomials forms a ring or a field. The standard construction is to model a polynomial as a function from the set of terms generated by the variables into $L$. However, to establish the set of polynomials itself as a (commutative) ring it is not necessary for the set of coefficients $L$ to be a full commutative ring with $1 \neq 0$. For example, to prove that addition of polynomials is associative, commutativity of multiplication in $L$ is not needed. This has already been observed in [9] where the Mizar formalization of polynomials with an arbitrary (even infinite) number of variables was described. In this paper, we investigate polynomials with one variable. Of course, polynomials with one variable can be constructed using the general approach of [9]. However, for the case of one variable, the construction of the ring of polynomials can be done much easier: terms occur only as powers of the given variable. Hence, polynomials can be represented simply as a function from the natural numbers into $L$. The hope was that even fewer algebraic requirements would be necessary for the domain $L$ than for the construction of polynomials with an arbitrary number of variables.

In the rest of this section, we review the construction of polynomials given in [9] and [11]. The reader familiar with this material may skip to Section 2, where a new approach

Manuscript received October 1, 2001; revised June 10, 2002.

[1] The work of the first author has been partially supported by CALCULEMUS grant HPRN-CT-2000-00102 and TYPES grant IST-1999-29001.

[2] Part of this work was done while the second author visited Bialystok under CALCULEMUS grant HPRN-CT-2000-00102.

for Mizar formalization of polynomials with one variable is presented. Polynomials with an arbitrary number of variables have been modelled in [9] as functions from the set of terms into the underlying structure *L*. A *bag* is a function from the set of terms over a given set of variables *X* into the natural numbers.

```
definition
let X be set;
mode bag of X is natural-yielding finite-support ManySortedSet of X;
end;
```

Then, a formal power series over *X* is nothing more than a function from the set of *bags of X*, called *Bags X*, into an underlying structure and more importantly, a polynomial is just a special formal power series, namely one where almost all terms are mapped to zero. This property is called *finite-Support*.

```
definition
let X be set;
let S be 1-sorted;
mode Series of X,S -> Function of Bags X, S means
  not contradiction;
end;

definition
let n be Ordinal;
let S be non empty ZeroStr;
mode Polynomial of n,S is finite-Support Series of n,S;
end;
```

Note that in order to define polynomials as a refinement of formal power series, the underlying structure has to provide a zero and hence it must be a structure with a carrier and a special zero element, that is, a ZeroStr. Now, the set of polynomials with its corresponding operations can be defined, and it was proven in [9] that this set is a ring if the coefficients *L* are a (not necessarily commutative) ring. However, polynomial evaluation was shown in [11] to be a homomorphism of rings only for the commutative case:

```
definition
let n be Ordinal;
let L be right_zeroed add-associative right_complementable
          Abelian  well-unital distributive non trivial
         commutative associative (non empty doubleLoopStr);
let x be Function of n,L;
cluster Polynom-Evaluation(n,L,x) -> RingHomomorphism;
end;
```

This may be due to the lack of better proofs avoiding the natural use of this property, that is, the attribute Abelian. However, this does not seem likely at the moment and we decided to restrict ourselves to polynomials with one variable to see whether or not in this special case fewer properties are necessary. This also touches on the question mentioned in [8]: how much generalization is best?

## 2. Polynomials with One Variable

The Mizar construction of polynomials with one variable based on finite support formal power series was presented in [3]. With only one variable, it is possible to define a formal power series as a function from the natural numbers into a structure L. This is done using the Mizar mode sequence (see [7]). Note that L only has to provide a carrier, but no operations.

```
definition
let L be 1-sorted;
mode sequence of L means
  it is Function of NAT, the carrier of L;
end;
```

Of course, if we want to work seriously with this definition of formal power series, the structure L has to fulfill further properties. For example, to define the addition of two formal power series, the carrier of L has to be non empty and, furthermore, it has to provide an addition itself. So L has to be a LoopStr (see [12]).

```
definition
let L be non empty LoopStr;
let p,q be sequence of L;
func p+q -> sequence of L means
  for n be Nat holds it.n = p.n + q.n;
end;
```

The negation –p and the difference p-q of two formal power series p and q are defined the same way. The definition of the product p*'q turned out to be a bit more complicated. For this, it was necessary to introduce a helper sequence for collecting the intermediate products of p and q at certain terms (compare [8,9]). The coefficient of the product series p*'q at a certain term is then given as the sum of the elements of this helper sequence. However, a special functor decomp for splitting up bags used in the general approach is not necessary here. Of course the underlying structure L now has to provide a multiplication also. Due to the use of addition and multiplication, it must be a doubleLoopStr.

```
definition
let L be non empty doubleLoopStr;
let p,q be sequence of L;
func p*'q -> sequence of L means
  for i be Nat
  ex r be FinSequence of the carrier of L st
    len r = i+1 &
    it.i = Sum r &
     for k be Nat st k in dom r holds r.k = p.(k-'1) * q.(i+1-'k);
end;
```

In the above definition –' (see [6]) denotes subtraction restricted to the natural numbers, that is, *k-'l* equals zero if *l* is greater than *k* for natural numbers *k* and *l*. Although this slightly complicates the definition, it was necessary merely because the argument of a sequence is not allowed to be less than zero. We also defined the zero series 0_.(*L*) and the unit series 1_.(*L*) that are necessary later to construct the ring of polynomials. A polynomial is a formal power series *p* where non-zero values appear only in a finite starting segment of *p*. Again this property is called *finite-Support* although technically it does not exactly correspond to the finite-Support property for polynomials with an arbitrary number of variables (see [4]). It should be clear now that to define polynomials in this way, the structure *L* must have a non empty carrier as well as a zero element 0.*L* and hence it must be a ZeroStr.

```
definition
let L be non empty ZeroStr;
mode Polynomial of L is finite-Support sequence of L;
end;
```

Due to the inheritance mechanism of the Mizar system, all operators defined for formal power series may also be applied to polynomials. However, the result will be only a formal power series and not a polynomial. This can be achieved by using term adjective registrations which say, for example, that the sum of two finite-Support sequences again has the finite-Support attribute; hence, it is not only a formal power series, but in fact a polynomial. Of course proving such registrations sometimes requires further properties concerning the underlying structure *L*: to prove the just mentioned registration for the addition of polynomials, for example, it was necessary for *L* to be *right_zeroed*, that is, for all elements *x* of *L* holds *x* + 0.*L* = *x* (and not necessarily 0.*L* + *x* = *x*). To prove analogous results for *–p* and *p-q*, it turned out that *L* had to be *add-associative*, *right_zeroed*, and *right_complementable* (see [12]), whereas for the multiplication *p*'q* of polynomials, a fourth attribute, *distributive* (see [2]), was necessary.

```
definition
let L be add-associative right_zeroed right_complementable
         distributive (non empty doubleLoopStr);
let p,q be Polynomial of L;
cluster p*'q -> finite-Support;
end;
```

Now we can define the ring of polynomials with one variable over *L*. Of course here *L* has to fulfill all the above attributes which are necessary to prove the registrations about finite-Support, that is, the attributes necessary for ensuring that operations are closed with respect to polynomials. However, according to the following definition, the type of *Polynom-Ring L* is just *strict non empty doubleLoopStr* (see [2]); further registrations introduce properties necessary for *Polynom-Ring L* to indeed be a ring. The carrier of *Polynom-Ring L* corresponds to the set of all polynomials, the sum and product to the sum and product of two polynomials, the zero element is the series 0_.(*L*) and the unity is the series 1_.(*L*) mentioned above.

```
definition
let L be add-associative right_zeroed right_complementable
         distributive(non empty doubleLoopStr);
func Polynom-Ring L -> strict non empty doubleLoopStr means
  (for x be set holds
   x in the carrier of it iff x is Polynomial of L) &
   (for x,y be Element of the carrier of it,  p,q be sequence of L
     st x = p & y = q holds x+y = p+q) &
  (for x,y be Element of the carrier of it, p,q be sequence of L
     st x = p & y = q holds x*y = p*'q) &
  0.it = 0_.(L) &
  1_(it) = 1_.(L);
end;
```

Further properties of *Polynom-Ring L* are shown again using term adjective registrations; attributes included here are *Abelian*, *commutative*, *add-associative*, *associative*, *right_zeroed*, *right_complementable*, *right_unital*, and *distributive*, which means that *Polynom-Ring L* is a commutative ring. Of course, to prove these registrations there are again further requirements on *L*. For example, to show the distributive property, we have to assume in addition that *L* is Abelian (see [12]).

```
definition
let L be Abelian add-associative right_zeroed right_complementable
         distributive (non empty doubleLoopStr);
cluster Polynom-Ring L -> distributive;
end;
```

Using registrations here has the advantage that in order to prove that a particular object is a polynomial ring over a structure L, all one has to show is that this object is a *strict non empty doubleLoopStr* (and of course that the definiens, i.e., the statements after the “means”, is fulfilled). The other properties of a ring are given for free, if the coefficient domain L has all the properties required by the registrations.

## 3. Evaluation of Polynomials with One Variable

In this section we present the Mizar formalization of the evaluation of polynomials with one variable. For that we define a functor *eval* (compare [4]) returning the value of a given polynomial at a given Point $x \in L$. Like in the definition of the product of two formal power series, we employ a finite helper sequence to collect intermediate results, which are then summated using the *Sum* functor defined for finite sequences. This is the usual technique in Mizar when the sum or product of quite a number of elements has to be made explicit. We note that for the definition of the *eval* functor, the underlying structure *L* has to be only *non-empty* and *unital* (see [13]).

```
definition
let L be unital (non empty doubleLoopStr);
let p be Polynomial of L;
let x be Element of the carrier of L;
func eval(p,x) -> Element of L means
  ex F be FinSequence of the carrier of L st
    it = Sum F &
    len F = len p &
   for n be Nat st n in dom F holds
        F.n = p.(n-'1) * (power L).(x,n-'1);
end;
```

We would like to mention that the definiens of the functor *eval* could also have been equivalently written as:

```
for F be FinSequence of the carrier of L
st len F = len p &
   for n be Nat st n in dom F holds F.n = p.(n-'1)*(power L).(x,n-'1)
holds it = Sum F;
```

However, in this case, it would be necessary to explicitly construct such a finite sequence *F* each time we want to use the definition: in contrast to the first definition, this one does not guarantee the existence of *F*. On the other hand, if one wants to show that a given element $a \in L$ equals *eval(p,x)*, the second one would be better because in this case the first definition requires the construction of *F* whereas the second only needs the fact that a equals *Sum F* for a sequence *F* fulfilling the given preconditions. However, this does not happen often in our proofs, so we decided to use the first definition.

Again, to prove further properties of *eval*, for example, that it is compatible with the addition of polynomials as in:

```
theorem
for L be Abelian add-associative right_zeroed right_complementable
        unital left-distributive (non empty doubleLoopStr)
for p,q be Polynomial of L
for x be Element of the carrier of L holds
  eval(p+q,x) = eval(p,x) + eval(q,x);
```

some additional properties of *L* are necessary. Though these properties are obvious, proving them turned out to be tedious. In particular, the proof concerning multiplication of polynomials, that is, *eval(p*'q,x) = eval(p,x)*eval(q,x)*, was quite technical as we had to do a double induction. Of course there are other, and maybe better, ways to prove these properties, but for us it was more important to use as few properties of the structure *L* as possible. In fact, in the first version of [4], the fact that *L* is a field was used, which by changing the proof could be improved to:

```
theorem
for L be add-associative right_zeroed right_complementable
        Abelian left_unital distributive commutative associative
        non trivial (non empty doubleLoopStr)
for p,q be Polynomial of L
for x be Element of the carrier of L holds
  eval(p*'q,x) = eval(p,x) * eval(q,x);
```

Now we define the evaluation of a polynomial over *L* at a given term *x* as a function *eval* from *Polynom-Ring L* into *L*. To be more precise, given a structure *L* and an element *x ÎL*, the function *eval* returns the value *eval(p,x)* for every polynomial *p*. That this function is in fact a homomorphism of rings is stated analogously to the definition of *Polynom-Ring L*, using term adjective registrations.

```
definition
let L be add-associative right_zeroed right_complementable
        distributive unital (non empty doubleLoopStr);
let x be Element of the carrier of L;
func Polynom-Evaluation(L,x) -> map of Polynom-Ring L,L means
  for p be Polynomial of L holds it.p = eval(p,x);
end;
```

Note that in this definition it is sufficient that *L* is an *add-associative right_zeroed right_complementable distributive unital* (*non empty doubleLoopStr*); this holds because we required the type of *Polynom-Evaluation(L,x)* to be a function only, not a homomorphism already. Again these properties are stated using term adjective registrations. As should be clear by now, further properties of the coefficient domain *L* become necessary to prove these registrations. Specifically, for *Polynom-Evaluation(L,x)* to be a homomorphism of rings, we need the fact that the structure *L* is a commutative ring with $1\neq 0$. We conclude with the following:

```
definition
let L be add-associative right_zeroed right_complementable
         Abelian left_unital distributive commutative
         associative non degenerated (non empty doubleLoopStr);
let x be Element of the carrier of L;
cluster Polynom-Evaluation(L,x) -> RingHomomorphism;
end;
```

## 4. Comparison of the Constructions

Our hope was that by having the restriction to one variable, the construction of (the ring of) polynomials could be done with fewer attributes of the underlying coefficient domain *L* than in the general case. Unfortunately, when defining the ring of polynomials over *L* and proving that it is indeed a ring, the only attribute concerning *L* that vanished was non trivial. However, in terms of certain properties, some improvements could be made: in contrast to the general case, to prove the attribute distributive (for formal power series) for the restricted case, it was not necessary for multiplication of *L* to be associative. Also, the existence of a unit could be proven without the assumption that *L* is Abelian. Furthermore, defining polynomials as mathematical objects only, rather than as a full ring, requires much fewer attributes for *L*. For example, to define the unit polynomial we need only the attribute non empty whereas in the general approach six further attributes, namely, *non trivial*, *add-associative*, *right_zeroed*, *right_complementable*, *unital*, and *distributive* were used. Also, in proving that the addition of polynomials commutes for the one-variable approach, *right_zeroed* was not necessary. So the main lesson is that when defining the objects in the beginning, it can indeed be done with fewer attributes for polynomials with one variable. However, when proving more and more properties of polynomials, one gets closer to the level of the general case. This results in polynomial evaluation where again the definition of the functor *eval* needed much fewer attributes in the restricted case, but to prove that it is indeed a homomorphism of rings in both approaches the same attributes concerning L were necessary.

Of course, which attributes are necessary is determined by the proof of a theorem. For example, as mentioned in Section 3, the fact that the evaluation of one-variable polynomials respects multiplication was first proven using the attribute *Field-like*; by a slight modification of the original proof this attribute is now avoided. Hence, there is the possibility that there are other proofs which would allow a decrease in the number of necessary attributes in one of the constructions. It is hard to estimate whether a set of attributes is indeed minimal for a theorem (independent of the proof technique used). In fact, there are theorems in which such a minimal set is not unique (see [10]).

The construction, including the proofs, was of course much easier in the restricted case of one variable. For example, as mentioned in Section 3, to define multiplication of power series, the somewhat technical functor *decomp* for splitting bags into what is needed for multiplication of polynomials was not necessary. So it seems reasonable to have both concepts defined because if we know that all we need are polynomials with one variable, we can choose the easier restricted approach. In fact we plan to give a formal Mizar proof that both approaches - *Polynom-Ring(1,L)* and *Polynom-Ring L* - are isomorphic, so that switching between them will become easier. Again it would be interesting to see which algebraic properties of *L* are necessary to construct such a proof.

## 5. Conclusion

We presented an alternative Mizar construction of polynomials with one variable and compared it with the construction of polynomials with an arbitrary number of variables. In particular, we focused on properties of the underlying structure *L* allowing for such a construction. This also contributes to the area of non-commutative analysis [1,14] as, if possible, we in particular have proven our theorems without assuming commutativity of multiplication. We observed that although fewer properties of *L* were necessary to define the ring of polynomials with one variable, to prove that polynomial evaluation is a ring homomorphism required the fact that *L* is a commutative ring with $1\neq 0$ in both cases. However, polynomial theories not using the evaluation homomorphism can be formalized with much fewer requirements on the underlying structure if we restrict ourselves to one variable polynomials.

## References

[1] N. Bourbaki, *Elements of Mathematics, Algebra section 4*, Hermann and Addison-Wesley, 1973.

[2] Eugeniusz Kusak, Wojciech Leonczuk and Michal Muzalewski, *Abelian Groups, Fields and Vector Spaces*, Formalized Mathematics, Vol. 1, 1989, http://www.mizar.org/JFM/Vol1/vectsp_1.html.

[3] Robert Milewski, *The Ring of Polynomials*, Formalized Mathematics, Vol. 12, 2000, http://www.mizar.org/JFM/Vol12/polynom3.html.

[4] Robert Milewski, *Evaluation of Polynomials*, Formalized Mathematics, Vol. 12, 2000, http://www.mizar.org/JFM/Vol12/polynom4.html.

[5] Michal Muzalewski and Leslaw W. Szczerba, *Construction of Finite Sequence over Ring and Left-, Right-, and Bi-Modules over a Ring*, Formalized Mathematics, Vol. 2, 1990, http://www.mizar.org/JFM/Vol2/algseq_1.html.

[6] Takaya Nishiyama and Yasuho Mizuhara, *Binary Arithmetics*, Formalized Mathematics, Vol. 5, 1993, http://www.mizar.org/JFM/Vol5/binarith.html.

[7] Jan Popiolek, *Real Normed Space*, Formalized Mathematics, Vol. 2, 1990, http://www.mizar.org/JFM/Vol2/normsp_1.html

[8] Piotr Rudnicki, Christoph Schwarzweller and Andrzej Trybulec, *Defining Power Series and Polynomials in Mizar*, in: M. Kerber and M. Kohlhase (eds.), Symbolic Computation and Automated Reasoning: The Calculemus-2000 Symposium, A K Peters, 2000, pp. 191-204.

[9] Piotr Rudnicki and Andrzej Trybulec, *Multivariate polynomials with arbitrary number of variables*, Formalized Mathematics (to appear), Vol. 11, 1999, http://www.mizar.org/JFM/Vol11/polynom1.html.

[10] Christoph Schwarzweller, *The Binomial Theorem for Algebraic Structures*, Formalized Mathematics, Vol. 11, 1999, http://www.mizar.org/JFM/Vol11/polynom1.html.

[11] Christoph Schwarzweller and Andrzej Trybulec, *Evaluation of Multivariate Polynomials*, Formalized Mathematics, Vol. 12, 2000, http://www.mizar.org/JFM/Vol12/polynom2.html.

[12] Wojciech A. Trybulec, *Vectors in Real Linear Space*, Formalized Mathematics, Vol. 1, 1989, http://www.mizar.org/JFM/Vol1/rlvect_1.html.

[13] Wojciech A. Trybulec, *Groups*, Formalized Mathematics, Vol. 2, 1990, http://www.mizar.org/JFM/Vol2/group_1.html.

[14] B. Yousefi, *Unicellularity of the multiplication operator on Banach spaces of formal power series*, Studia Math. 147(3), 2001, pp. 201-209.