Designing Mathematical Libraries based on Minimal Requirements for Theorems

Christoph Schwarzweller
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen, Germany
schwarzw@informatik.uni-tuebingen.de

25th August 2001

Abstract

The main observation of this paper is that theorems can be considered independent of abstract domains; they depend rather on a set of properties necessary to prove the theorems correct. This leads to more general theorems and hence to improved reuse of mathematical theorems. We discuss how this view influences the design of mathematical libraries and illustrate our approach with examples written in the Mizar language.

1 Introduction

At the end of the 19th century the interest of mathematicians turned away from concrete objects like numbers or points and lines. Since then more abstract domains like groups, rings, or vector spaces given by axioms only have been considered. The advantage is obvious: Besides abstracting away from unnecessary details, this allows to prove theorems in an abstract setting. So once the theorem is proven for an abstract domain it holds in every particular domain fulfilling the axioms of the abstract one. However, this is not the end of the line: Often a theorem proven for an abstract domain does not depend on all properties of that domain; this can be usually observed in the proof of the theorem where not all the domain's properties are actually used. So we claim that the basis of mathematical theorems is not an abstract domain, but rather a set of properties. Consequently, theorems can be considered independent of domains: They exist in their own merit, and there are different abstract domains in which a theorem holds. In the following we explain this view on theorems in more detail and discuss its impact on the design of mathematical libraries. We give examples from (commutative) algebra formulated using the Mizar system [3] which can be seen as an implementation of our approach (see also [4]).

2 Requirements for Theorems

Considering a theorem independent of an abstract domain, what does constitute such a theorem? The answer is twofold: First, there are of course carriers and operators on these carriers necessary to state the content of the theorem. Second, a number of properties of the operators is necessary to prove the theorem correct. We call the set of carriers and operators the *signature* of the theorem, the set of properties the set of *adjectives* of the theorem. Note that not every combination of a signature with a set of adjectives leads to a well-known domain such as a ring or a vector space. The main advantage of viewing theorems this way is that now theorems can be stated (and proven) using as less adjectives as possible, hence for the minimal conditions under which they hold: Such a theorem holds for every particular domain providing the necessary signature and fulfilling all the adjectives required by the theorem. This gives maximal possibility for reusing theorems.

Let us consider a straightforward example: An ideal is conventionally defined over a ring R; it is a subset S of R that is closed with respect to both addition and multiplication with arbitrary ring elements, that is for all $a, b \in S$ and all $r \in R$ we have $a + b \in S$ and both $r \cdot a \in S$ and $a \cdot r \in S$. It is rather obvious that the set $\{0\}$ is an ideal for an arbitrary R where 0 denotes the ring's zero element. However, to prove that fact it is sufficient to show

$$0 + 0 \;=\; 0 \quad \text{and} \quad a \cdot 0 \;=\; 0 \;=\; 0 \cdot a \;\; \text{for all} \;\; a \in R.$$

A closer inspection shows that this can be done using only that addition is associative, provides a right zero and a right complement and that addition and multiplication distribute. These four properties are the adjectives necessary for this theorem, the signature consists of two binary operators $+$, $*$ and the element 0. In other words the set $\{0\}$ is an ideal in much weaker structures than rings.

3 Formalizing Requirements in Mizar

In this section we illustrate how requirements for theorems can be formalized in the Mizar language [3]. To be more precise, we explain how to define signatures and adjectives in Mizar and how these can be combined to describe requirements for theorems. Again we use the theory of ideals and the theorem from the last section as an example.[1]

First, we have to deal with the signature of a theorem, that is we have to fix the carriers and the operators the theorem is about. For that purpose, the Mizar language provides a special construct, the *structure definition*. Rings, for example, are given by a carrier, two binary and two nullary binary operators, the later ones denoting the zero and the unit element. Hence, the typical signature for rings—called in Mizar `doubleLoopStr`—can be defined simply by

[1] The basics of the theory of ideals have been formalized in Mizar [1] and are part of the Mizar Mathematical Library.

```
definition
struct (LoopStr,multLoopStr_0) doubleLoopStr
   (# carrier      -> set,
      add, mult    -> BinOp of the carrier,
      unity, Zero -> Element of the carrier #);
end;
```

Please note the terms `LoopStr` and `multLoopStr_0` mentioned in the first line of the definition. They also denote Mizar structures— `LoopStr` a carrier with one binary operator `add` and a zero element `Zero`, `multLoopStr_0` a carrier with one binary operator `mult` and a unit element `unit`— which are now glued together giving `doubleLoopStr`. As a consequence `doubleLoopStr` is in particular both a `LoopStr` and a `multLoopStr_0`.

Now adjectives can be introduced in Mizar by defining *attributes* over structures providing the necessary signature for this adjective. Note that in the following definition associativity of addition[2] is introduced for `LoopStr` only, just because multiplication is not necessary here. Nevertheless, the attribute `add-associative` will be available for `doubleLoopStr` also, as `LoopStr` is an ancestor of `doubleLoopStr`.

```
definition
let R be non empty LoopStr;
attr R is add-associative
  for x,y,z being Element of R holds x+(y+z) = (x+y)+z;
end;
```

Further properties can be introduced analogously, for example `right_zeroed`, `right_complementable` and `distributive`, which are the adjectives necessary to state the example theorem of section 2. Note, that the attributes `right_zeroed` and `right_complementable` can be defined also for `LoopStr` only, whereas `distributive` requires a `doubleLoopStr`, namely addition and multiplication.

In the same way adjectives necessary for a subset S of R to be an ideal over R can be introduced using attributes. Again, in the first definition `R` is a `LoopStr` that is `R` provides no multiplication. Similarly, the other two adjectives are defined for `HGrStr`—an ancestor of `multLoopStr_0`— giving a carrier and multiplication only.

```
definition
let R be non empty LoopStr, S being Subset of R;
attr S is add-closed means
  for x,y being Element of R st x in S & y in S holds x+y in S;
end;

definition
let R be non empty HGrStr, S being Subset of R;
attr S is left-ideal means
  for a,x being Element of the carrier of R st x in S holds a*x in S;
```

[2] The operator $+$ is just a shorthand for `add` using hidden arguments, so that the carrier R of the operator $+$ need not be explicitely stated.

```
attr S is right-ideal means
  for a,x being Element of the carrier of R st x in S holds x*a in S;
end;
```

Combining these three adjectives we get the notion of an ideal. Here we need the signature given by `doubleLoopStr`, as now both addition and multiplication must be provided. It may be worth mentioning that using our approach we implicitely indroduced the notion of an ideal for much weaker domains than for a ring; in fact we just used the signature of rings, but no further adjectives for the given operators.

```
definition
let R be non empty doubleLoopStr;
mode Ideal of R is add-closed left-ideal right-ideal (non empty Subset of R);
end;
```

Also, we like to mention that this definition is not necessary as it is just syntactic sugar, but it improves readability; in fact `Ideal of R` is just a synonym for `add-closed left-ideal right-ideal (non empty Subset of R)`. However, independent of which name we choose, Mizar expects an existence proof before such an attributed structure can be used. In our case we have to show that for an arbitrary `R` that is a `doubleLoopStr` there indeed exists a `add-closed left-ideal right-ideal (non empty Subset of R)`.

Now we can state the theorem mentioned in section 2 by just combining the signature, i.e. the appropriate Mizar structure, with the required adjectives, i.e. the required attributes:

```
theorem
for R being add-associative right_zeroed right_complementable
            distributive (non empty doubleLoopStr)
holds {0.R} is Ideal of R;
```

Note that in this theorem `R` provides exactly the adjectives necessary to prove the theorem correct and nothing more.[3] So this theorem is valid not only in rings, but also in much more general domains. In addition the theorem is easier applicable for a particular domain being a ring: We have to show only four adjectives to apply it, whereas it takes to prove eight adjectives to show that a domain is a ring. We will come back to that in section 5. Here we close by mentioning that, similar to the situation above where we defined ideals, the existence of an `add-associative right_zeroed right_complementable distributive (non empty doubleLoopStr)` has to be shown before the theorem can be stated the way we did.

4 Mathematical Libraries

In practice, domains in mathematical libraries are defined by introducing carriers and operators. Then further properties about these operators are stated

[3]The proof of the example theorem has been carried out and can be found in [1].

using some kind of axioms. So, here we also observe a distinction between signature and adjectives, because axioms can be considered as a special way of writing adjectives. However, this distinction beween signature and adjectives is not made when considering theorems: New theorems are stated (and proven) for a particular domain, that is for a fixed signature and a fixed set of adjectives. Whether or not all these operators and adjectives are necessary for the theorem's proof is seldom taken into consideration.[4] Consequently, theorems are closely bound to particular domains.

In contrast, how would a mathematical library be organized dealing with theorems based on this distinction between signatures and adjectives? First of course there are theorems of the form explained above: They are not bound to a particular domain, but exist with respect to a signature and a set of adjectives. As we have seen in section 2, such theorems hold for all particular domains providing at least the same signature and fulfilling at least the adjectives linked to the theorem. Domains in the usual sense, e.g. rings, modules or the (ring of) integers, are also part of such a library. However, again the perspective is somewhat different: A domain also consists of a signature giving the operators of the domain and in addition of a number of adjectives the domain's operators fulfill. As already mentioned, this perspective is quite obvious for abstract domains, but also concrete domains in the classical sense can be considered this way. For example, the signature of the integers would provide at least two binary operators $+$ and $*$ and at least two constants 0 and 1 with their usual meaning. Adjectives for the integers would be among others associativity and commutativity of $+$ and $*$ or that 0 is a unit with respect to $+$.

This organization of mathematical libraries enables straightforward application of theorems: Having the necessary signature and adjectives for both theorems and domains of the library, a given theorem holds for a given domain if both the signature of the theorem is included in the signature of the domain and the adjectives of the theorem are included in the (so far proven) adjectives holding for the domain.

5 Reusing Theorems in Mizar

In the following we illustrate how Mizar can serve as a mathematical library realizing the approach presented in the last section. In section 3 we already saw how theorems based on the distinction between signature and adjectives can be formalized (and proven) using the Mizar Language. Now we explain how particular domains can be contstructed and how general theorems can be applied to such domains. For abstract domain this is straightforward. An abstract ring, for example, can be defined the same way as ideals in section 2: We just combine the necessary structure with the necessary attributes:

[4] Of course it is used, for example, that theorems about groups also holds for the addition of rings. This in fact is shows that theorems are in some sense independent of particular domains.

```
definition
mode Ring is Abelian add-associative right_zeroed right_complementable
  associative left_unital right_unital distributive (non empty doubleLoopStr);
end;
```

Of course this again requires an existence proof. Note that the example theorem from section 3 is true for `Ring`: The structures of `Ring` and the theorem are the same and the theorem's attributes are a subset of the attributes a `Ring` fulfills. And in fact the Mizar checker accepts the theorem

```
theorem
for R being Ring holds {0.R} is Ideal of R;
```

by just referencing the general theorem from section 3. Please note again, that it was sufficient to define the notion of an ideal for the structure `doubleLoopStr` in section 3; no attributes were used. The simple fact that the definition of `Ring` is based on a `doubleLoopStr` guarantees that the notion of an ideal is also availble for `Ring`.

Constructing concrete domains in the classical sense requires a bit more work. Let us consider the integers as an example. First we have to introduce the signature of the integers. Here we consider the integers as a ring, that is the signature is at least a `doubleLoopStr`. Further components of the integers can be defined, e.g. to formalize an ordering, but for our purpose a `doubleLoopStr` is sufficient. First we have to define the set of integers `INT` as well as addition and multiplication of the integers denoted by `addint` and `multint`. Then we can construct the ring of integers `INT.Ring` as a functor yielding a `non empty doubleLoopStr`, in which the components are identified with the just mentioned carrier and operators.[5]

```
definition
func INT.Ring -> non empty doubleLoopStr equals
  doubleLoopStr(#INT,addint,multint,1 in INT,0 in INT#);
end;
```

Note, that this definition only results in a collection of the set `INT` with some operations on this set, that is a concrete instance of a signature. In order to apply general theorems to the integers we now have to prove that certain attributes are fulfilled. In Mizar this is done using *cluster definitions*. Of course, the fact that e.g. the addition of `INT.Ring` is associative can also be stated (and proven) as a theorem. However, clusters has the advantage, that the attribute is linked to the structure: After the cluster definition the Mizar checker can automatically infer that `INT.Ring` fulfills the clustered attribute. The number and the order of attributes in a cluster definition is without meaning and up to the user. In our example all attributes necessary for `INT.Ring` to be a ring are put in one cluster definition:

[5]The term `in INT` just makes sure that `1` and `0` are indeed elements of the set `INT` as required by the structure definition.

```
definition
cluster INT.Ring -> Abelian add-associative right_zeroed right_complementable
                    commutative associative distributive;
end;
```

After the cluster has been registered the following integer version of the theorem from section 3 holds. As in the case of `Ring` this is due to the fact that the involved structures are both `doubleLoopStr` and the theorem's attributes are a subset of the attributes a `INT.Ring` fulfills due to (the proofs of) the clusters.

```
theorem
{0.(INT.Ring)} is Ideal of INT.Ring;
```

Again all we have to do to get this theroem accepted by the Mizar checker is to refence the general theorem. Note also, that the clusters presented provide much more adjectives of the integers than necessary to prove the theorem. To do so, it is enough to show that `INT.Ring` is `add-associative` `right_zeroed` `right_complementable` and `distributive`.

6 Conclusion and Future Work

Interestingly, there are properties of which it is not obvious whether they should belong to a particular domain or not. For example, the natural numbers provide the principle of induction. In fact the principle of induction is a major part in the definition of the natural numbers used implicitly when defining the operators $+$ and $*$. One can consider this property as an adjective, but there are no examples of other domains than the natural numbers fulfilling it. So it seems that this property indeed belongs to a particular domain rather than exists in its on merit. Consequently, the domains of our approach would be equipped with both adjectives existing on their own and properties holding only for the particular domain. On the other hand induction for natural numbers can be considered as an instantiation of a higher order theorem saying that induction can carried out on well-ordered sets. The question here is, whether it makes sense to generalize every property as far as possible or whether it is more practicable to have properties adapted for particular domains.

However, we believe that the approach we presented provides a flexible way to organize mathematical libraries, in particular with respect to reusing theorems included in such a library. Furthermore, it seems that our approach also works for algorithmic libraries: The signature of an algorithm gives the necessary carriers and operators, the adjectives of an algorithm describe minimal conditions under which the algorithm works correctly. This kind of description can be done using e.g. the concept description language Tecton [2]. Consider for example a sort algorithm working on a domain giving a set and a binary relation on this set. Then the signature of the algorithm consists of a carrier and a binary relation over this carrier. Adjectives necessary for the correctness of the algorithm are reflexivity, antisymmetricty, transitivity and totality of the binary realation. The integers with its usual order fulfill all the adjectives, hence the integers constitute a legal domain for the sort algorithm.

References

[1] Jonathan Backer, Piotr Rudnicki and Christoph Schwarzweller, *Ring Ideals.* in: Formalized Mathematics, 2000. (to appear); available by anonymous ftp from `http://mizar.org/JFM/Vol12/ideal_1.html`.

[2] D. Musser, *The Tecton Concept Description Language.* available by anonymous ftp from `http://www.cs.rpi.edu/~musser/gp/tecton`, 1998.

[3] Piotr Rudnicki and Andrzej Trybulec, *On Equivalents of Well-foundedness. An Experiment in Mizar.* in: Journal of Automated Reasoning, 23:197–234, 1999.

[4] Piotr Rudnicki, Christoph Schwarzweller and Andrzej Trybulec *Commutative Algebra in the Mizar System.* in: Journal of Symbolic Computation, vol. 32(1/2), pp. 143-169, 2001.

[5] Christoph Schwarzweller, *The Ring of Integers, Euclidean Rings and Modulo Integers.* in: Formalized Mathematics, vol.8(1), pp. 17-22, 1999. available by anonymous ftp from `http://mizar.org/JFM/Vol11/int_3.html`.