Robbins Algebras vs. Boolean Algebras

Adam Grabowski*
Institute of Mathematics
University of Białystok, Poland
adam@math.uwb.edu.pl

Abstract

In the early 1930s, Huntington proposed several axiom systems for Boolean algebras. Robbins slightly changed one of them and asked if the resulted system is still a basis for variety of Boolean algebras. Partial solution has been given by Winker in 1992 and in 1996 McCune with the help of theorem prover gave the positive answer. Some simplified and restructurized versions of this proof are known. We describe in this paper some of the issues concerned with full mechanically checked proof of the fact that all Robbins algebras are Boolean.

1 Introduction

In 1933 E. V. Huntington [12] presented the following basis for Boolean algebras:

$$a + b = b + a,$$

$$(a + b) + c = a + (b + c),$$

$$(a' + b')' + (a' + b)' = a;$$

where a, b, c are arbitrary elements of an abstract algebra $(L, +, ')$.

In this *fourth set of independent postulates for the algebra of logic* the last one is called *Huntington equation.* Robbins proposed another axiom as a replacement of Huntington equation, namely:

$$((a + b)' + (a + b')')' = a,$$

however his hypothesis that commutativity, associativity together with *Robbins equation* is sufficient to characterize Boolean algebras remained open for over 60 years. In 1992 Winker gave the partial solution of the problem by proposing several another conditions which imply Huntington axiom. One of them, named *Winker's second condition* played a crucial role in the solution of Robbins problem:

*This work has been partially supported by CALCULEMUS Grant HPRN-CT-2000-00102.

A Robbins algebra satisfying $\exists x \exists y\ (x+y)' = y'$ *is a Boolean algebra.*

Given by W. McCune with the help of automated theorem prover EQP/OTTER proof (October 1996) that all Robbins algebras satisfy also Winker condition started a series of papers devoted to methods of proving/analysing that fact in different systems ([4], Mathematica [6], ILF [5]).

Shortly thereafter, the problem has been formalized also in MIZAR language. However, we did not decide to include the article to the Mizar Mathematical Library (MML for short). In fact, in the very first form we proved only that algebras satisfying Robbins equation satisfy also Winker's second condition. One of the aims of my work was 'an attempt to translate EQP/OTTER proof into more human-readable form' [14].

In this earliest formalization of Robbins algebras we used another structure which was inappropriate for work with formal apparatus for Boolean lattices yet defined in the MML.

In our opinion, state of the MIZAR database at that time (1997, MIZAR version 5.3.07, MML version 3.5.544) posed us far from the convenient way of formalization. Numerous revisions of the MML urged us to revise also our proof to keep it correct with respect to the MML and enabled us to find a way of more writeable construction of the proof.

In [8] we formalize full proof of the fact that all Robbins algebras are Boolean and, as far as we know, it is first mechanically checked proof of Robbins conjecture which takes into account Huntington's [11] paper as well. Furthermore, extensive use of notions and theorems for lattice theory yet defined in the Mizar Mathematical Library points us its possible improvements.

The structure of this paper is as follows. In Section 2 used structures are described, the next section contains basic notions for Boolean algebras which were available in the Mizar Mathematical Library. Section 4 deals with main mechanisms and techniques which were of major importance during our work with adopting other proofs into MIZAR. In the fifth section position of MIZAR proof among the other machine proofs is shown while in the last section we draw some concluding remarks.

We cite definitions straight from corresponding MIZAR abstract files, so correctness conditions are not given (as well as justifications for theorems and clusters).

We will use $'$ as a symbol of complement but in MIZAR text ` is used instead.

2 Hierarchy of structures

According to the classical definition, Boolean algebra is a 6-tuple $\langle B, \vee, \wedge, {}', 0, 1\rangle$, where $\vee, \wedge$ are binary operations in B, $'$ is function from B into B and $0, 1$ are two distinct elements of B. The axiomatics concerned with it is well known.

This could lead us to introduce new MIZAR structure as an ancestor of `GroupStr` and `doubleLoopStr` as defined in [13]. The latter one has been used widely e.g. while developing theory of polynomials in MIZAR, especially formalizing Fundamental Theorem of Algebra (see [15] for details). This structure was also used in our very first attempt to formalize Robbins algebras.

For two reasons we decided that we choose current formalization. One of them was to be close as possible to Huntington's original paper where the underlying structure was $\langle B, \vee, {}' \rangle$. Moreover, intensive work with formalization of [7] (nearly 60 Mizar articles and numerous revisions of MML) made lattice theory developed in Mizar more suitable to work with. In fact, the only two definitions of 'Boolean' attribute in the MML were introduced just for lattices. The first—for partially ordered sets, and the second for algebras with two binary operations. The first approach has its full exposure in [2].

Figure 1 illustrates the relationship of used structure modes.

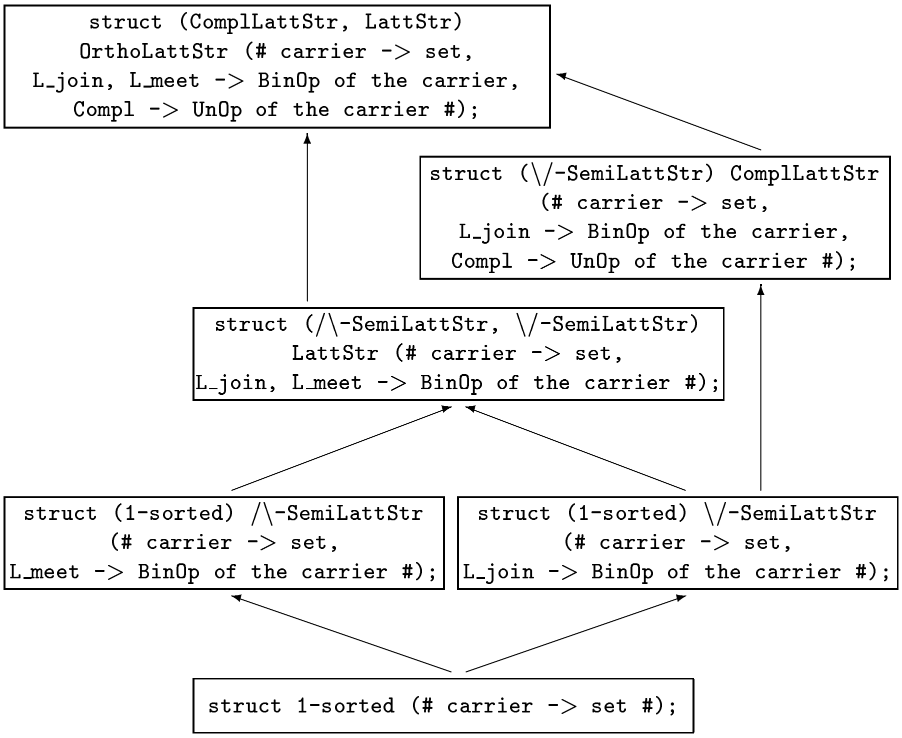

Figure 1. The net of structures used in the article.

The `ComplLattStr` has been introduced to keep defined notions and theorems at the satisfactory level of generality. The `OrthoLattStr` ensures that all the attributes (with 'Boolean' at the top) for `LattStr` can be used. In fact, full correspondence between newly introduced Robbins and yet defined in the MML Boolean algebras can be shown for lattices, not for semilattices.

3 Boolean Algebras: Formalization in MIZAR

In this section we shall describe briefly the formal apparatus for lattices introduced in the MML [18] and its correspondence to axiom systems proposed by Huntington (*the first set of independent postulates for the algebra of logic* [10, 11]).

The lattice theory in MIZAR is built on the structure mode `LattStr` as a backbone. The mode `Lattice` is defined as `Lattice-like non empty LattStr`.

The present form of the definition of mode `Lattice` by attributes is relatively new. In 1989 when the article [18] was accepted into the MML inheritance for structure modes remained in MIZAR open question. Library Committee of the Association of Mizar Users introduced `1-sorted` structure in 1995 [1].[1]

In the original version of the article [18] all 6 lattice axioms were included as a definiens to the mode `Lattice`.

When the MIZAR language evolved and conception of *attribute* was implemented, the automatic revision of MML took place and many mode definitions were replaced by ground structure mode with collection (*cluster*) of adjectives. The attribute names were built by adding postfix `-like` to corresponding mode names.

Just around 2000 we decided to separate from the previous single adjective `Lattice-like` six new attributes: `join-commutative, join-associative, join-absorbing` and dual `meet-commutative, meet-associative, meet-absorbing`. The names are self-explanatory enough.

The attribute `Lattice-like` is still kept as an abbreviation for these six axioms.

It could be noticed that many theorems and even definitions are often stated with all six attributes although the proof efectively used only two–three of them. We have software which detects assumptions and proof steps which are not used at all but we lack mechanisms how to find more general required type or conditions under which the proposition remains true. Anyway, this problem doesn't seem to be solvable fully automatically.

Obviously the problem of keeping notions at the possibly high level of generality is particularly important in articles which form the foundations of a given field of mathematics. It can be explained better using the definition of a complement from [18]:

```
definition let L be non empty LattStr,
               a, b be Element of the carrier of L;
  pred a is_a_complement_of b means
:: LATTICES:def 18
   a\/b =⊤L & b\/a = ⊤L & a/\b = ⊥L & b/\a = ⊥L;
end;
```

together with corresponding adjective definition:

[1] In fact implementation of a concept of ancestor for structure is older (about 1993), in 1995 only a part of built-ins was separated into regular article [1].

```
definition let IT be non empty LattStr;
  attr IT is complemented means
:: LATTICES:def 19
   for b being Element of the carrier of IT
    ex a being Element of the carrier of IT st a is_a_complement_of b;
end;
```

MIZAR functors[2] $\top$ and $\bot$ has been defined previously for bounded lattices. So if one wanted to use the attribute `complemented`, one had to prove that the underlying object has all these properties what was obviously a big disadvantage. We added two conjuncts to the first definiens to get rid of attribute `join-commutative` for L.

The $a \vee b$ is a shorter notation for `(the L_join of L).[a,b]`, dually is $a \wedge b$. This is usual way—right after the selector functor is introduced.

What is worth noticing at this point is that the classical axiomatics proposed e.g. by Grätzer [9] or Birkhoff [3] consists of 8 axioms and the idempotency laws can be derived from the other axioms (namely absorption).

For any element a of structure in which the both absorption laws hold we have that $\big(a \wedge (a \vee a)\big) \vee a$ is equal to $a \vee a$. On the other hand it is equal to a. By similar reasoning we have $a \wedge a = a$.

Huntington's definition of Boolean algebra agrees with that of [18], with one exception: according to the definition in [18], the carrier of a lattice may be a singleton. This contradicts with Huntington's *the first set of postulates for the algebra of logic*, where we have as an axiom:

VI. There are at least two distinct elements in the class K.

'Boolean' stands in MML for 'bounded complemented distributive' Lattice.

A few explanations are due to the definition of adjective 'bounded'. On the one hand, bounded lattice can be seen as a structure with additional **0** and **1**. This leads to introducing three additional structures to make inheritance issues implemented for MIZAR structures working. On the second hand, lower and upper bound can be defined separately, not as selectors in underlying structures, but using adjectives. The latter way has been chosen in the MIZAR library.

Let us recall here introduced concepts for upper bound.

```
definition let IT be non empty \/-SemiLattStr;
  attr IT is upper-bounded means
:: LATTICES:def 14
   ex c being Element of the carrier of IT st
    for a being Element of the carrier of IT holds c\/a = c & a\/c = c;
end;
```

This adjective is used in the assumption of the following definition:

[2]As of now, Library Committee plans to fully eliminate extended ASCII characters from MML articles; these two symbols will be replaced by `Top` and `Bottom` respectively. We hope they succeed soon.

```
definition let L be non empty \/-SemiLattStr;
 assume  L is upper-bounded;
  func ⊤L -> Element of L means
:: LATTICES:def 17
   for a being Element of the carrier of L holds it\/a = it & a\/it = it;
end;
```

The definition of upper bound for arbitrary upper semilattice structure is permissive: if the semilattice L is not necessarily upper-bounded, all we can say about $\top L$ is that is an element of L (it expands to the type element of the carrier of L), but we do not know which one.

If the semilattice L is upper-bounded according to the former definition, i.e. if suitable element of the carrier of L exists, it plays a role of the upper bound for L. Obviously, if one wants to use definiens of $\top L$, one has to prove that L is upper-bounded.

Due to permissiveness of the bounds in the previously cited predicate definition `is_a_complement_of` the attribute `bounded` for one of the loci can be dropped.

Introducing bounds in this manner allows for writing cluster like this:

```
definition
  cluster finite -> bounded Lattice;
end;
```

with all the automatics it provides. This will be described in the next section, as the concept of clusters is crucial for our formalization.

4 Robbins Algebras Are Boolean

4.1 Attributes and Clusters

To be close to the Huntington formalization of Boolean algebras we could introduce new Mizar modes, e.g. `Boolean_Algebra` which satisfies commutativity, associativity and Huntington equation and `Robbins_Algebra` with commutativity, associativity and Robbins axiom. However introducing individual axioms by adjectives is more elegant and provide some automatics.

There is 944 adjective symbols in all MML vocabularies and 1388 definitions and redefinitions of attributes in the whole Mizar library (MML version 3.23.701). We work mainly with seven attributes for `OrthoLattStr`, that is `join-associative`, `join-commutative`, `Boolean`, `Huntington`, `Robbins`, `de_Morgan`, `well-complemented`. While first three were described in the previous section, let us cite the definitions of two others, which are introduced for `ComplLattStr` and are fundamental for our formalization:

```
definition let L be non empty ComplLattStr;
  attr L is Robbins means
:: ROBBINS1:def 5
   for x, y being Element of the carrier of L holds
```

```
    ((x + y)` + (x + y`)`)` = x;

  attr L is Huntington means
:: ROBBINS1:def 6
   for x, y being Element of the carrier of L holds
     (x` + y`)` + (x` + y)` = x;
end;
```

Obviously, to have the notion of Robbins algebra according to the literature, we have to define it as Robbins join-commutative join-associative non empty ComplLattStr. Because we cannot use with these attributes adjective `Boolean` (we have only one binary operation defined), we have to conclude our formalization of Huntington's proof with the following cluster:

```
definition
 cluster Robbins -> Huntington (join-associative join-commutative
   (non empty ComplLattStr));
end;
```

which can be read as: *every join-associative join-commutative non empty ComplLattStr which is Robbins is also Huntington.*[3]

There is 4579 registrations of clusters in the MML (there are three types of them: existential, functorial and conditional, see [16] for precise description).

To make use of all formal apparatus from [18], we introduced two adjectives of rather technical nature: if we read `OrthoLattStr` as $(L, \vee, \wedge, \,')$, then attribute `de_Morgan` means that $\wedge$ is properly defined in terms of $\vee$ and $'$. In MIZAR text:

```
definition
  let L be non empty OrthoLattStr;
  attr L is de_Morgan means
   for x, y being Element of the carrier of L holds
    x /\ y = (x` \/ y`)`;
end;
```

The name `de_Morgan` can be slightly confusing; De Morgan Algebras are characterized in literature by (two) de Morgan identities but in class of algebras which are described here (the complement operation is involution) it is equivalent definition.

If L is of the type `non empty OrthoLattStr`, all we can say about a' for arbitrary element a of the carrier of L is that a' is again an element of the carrier of L. In other words, $'$ is only a unary operation on the carrier of L. The second attribute guarantees that $'$ operation has properties of complement as defined in [18].

```
definition
  let L be non empty OrthoLattStr;
  attr L is well-complemented means
```

[3]This form appears in the Journal of Formalized Mathematics which is machine translation of MIZAR articles.

```
    for a being Element of the carrier of L holds
     a` is_a_complement_of a;
end;
```

One of the most important questions which arose during our work was how to establish a connection between $\langle L, \vee, {}' \rangle$ and $\langle L, \vee, \wedge, {}' \rangle$ (or rather $\langle L, \vee, \wedge \rangle$). We wanted the underlying structures to inherit the carrier, the join and the complement operation and add one more binary operation of meet which is defined in terms of $\vee$ and ${}'$.

We decided to define MIZAR functor which transforms semilattice into the corresponding lattice structure:

```
definition let L be non empty ComplLattStr;
  func CLatt L -> strict OrthoLattStr means
:: ROBBINS1:def 11
    the carrier of it = the carrier of L &
    the L_join of it = the L_join of L &
    the Compl of it = the Compl of L &
    for a, b being Element of the carrier of L holds
      (the L_meet of it).(a,b) = a * b;
end;
```

The auxiliary function $a * b$ is just equal to $(a' \vee b')'$.

```
definition let L be join-commutative join-associative Huntington
    (non empty ComplLattStr);
  cluster CLatt L -> complemented distributive bounded;
end;
```

Obviously then, Clatt L is `de_Morgan` and, under certain assumptions, Boolean.

Existential registrations of clusters are significant part of MIZAR article. Nevertheless, to prove that there exists a structure in which Huntington and Robbins equations are satisfied, we use trivial (i.e. with singleton as its carrier) `OrthoLattStr`.

The main theorem stating that all Robbins algebras are Boolean is expressed as conditional cluster registration:

```
definition
  cluster Robbins de_Morgan -> Boolean preOrthoLattice;
  cluster Boolean -> Robbins (well-complemented preOrthoLattice);
end;
```

where `preOrthoLattice` is `OrthoLattStr` with non empty carrier, its $\vee$ and $\wedge$ operations are commutative, associative and satisfy the absorption laws.

All described constructions force the MIZAR checker to accept without justification the following:

```
for L being Robbins de_Morgan preOrthoLattice holds
  CLatt L is Boolean;
```

4.2 Properties

There are five properties for predicate definitions implemented (reflexivity, irreflexivity, symmetry, antisymmetry, connectedness) and four for functor definitions (commutativity, idempotence, involutiveness, projectivity).

By introducing properties for selected functor, e.g. as follows ('`+`' stands as a synonym to '`\/`'):

```
definition let L be join-commutative (non empty ComplLattStr),
               x, y be Element of the carrier of L;
  redefine func x + y;
  commutativity;
end;
```

we force the checker to accept $x + y = y + x$ without reference to the definition of the attribute `join-commutative`.

After deleting `commutativity` properties for join and meet operations 186 errors occur, so using properties we can significantly shorten the MIZAR article.

Another properties could also be useful here, e.g. `involutiveness` for complement operation (i.e. $x'' = x$ for arbitrary element x of the carrier of L) or `idempotence` for join operation ($x + x = x$). These properties have been implemented quite recently, unfortunately they cannot be introduced in redefinitional block, like `commutativity` does. We hope that it will be corrected in the future versions of MIZAR verifier.

5 Comparison of the formalizations

Our approach took into account different formalizations of Robbins algebras. In fact, the crucial result was that all Robbins algebras satisfy also Winker second condition and this proof was given by McCune.

Let us list here main facts encoded in [8]:

- Fourth set of independent postulates is equivalent to the first set [11];
- Robbins algebras satisfying Winker second condition are Huntington, originally proved by Winker [14],
- Robbins algebras satisfy Winker second condition [5];
- Boolean algebras are Robbins [6];
- Several conditions which make near-boolean algebras boolean [14]

Fourth and first set of postulates are axiom systems for Boolean algebras, as described in detail in Section 1 and 3, respectively.

It should be pointed out here that during translation of Huntington's proofs into MIZAR we discovered a few, perhaps typographical, mistakes in original text.

The name *de Bruijn factor* has been proposed by Freek Wiedijk [17] to describe 'loss factor' between the size of an ordinary mathematical exposition and its full formal translation inside a computer. However in Wiedijk's considerations and examples it is equal to four, in our case two is a good upper approximation.

It is comparatively low for the part which is a formalization of the original Huntington proof. For 11 lemmas with proofs ([11], p. 282) the quotient of the size of a formalization of a mathematical text in MIZAR and the size of its informal original is equal to 1.26 if we compare gzipped MIZAR and TEX version.

For other parts it is bigger: unfortunately `associativity` property still remains not implemented in MIZAR. This was one of major disadvantages of my formalization, e.g. if one wants to infer that $a + (b + c + d) = a + b + c + d$, one should in each step refer to the definition of the `join-associative` attribute.

The de Bruijn factor grows up also dramatically if we take into account the proof of Lemma 4.29 or 4.33, p. 283 ('*By similar reasoning, we find* $(A + B + C)(D + E + F + G + H) = z$*, which proves the theorem.*').

We did not try to find the de Bruijn factor for the parts which are MIZAR formalization of OTTER proofs.

It should be noticed here that the output from OTTER is relatively easier to transform automatically into MIZAR formalism than proof objects converted by ILF.

Let us compare here the line of EQP/OTTER's proof:

```
152   n(n(D+n(C))+n(C+n(D+n(C))))=D            [20→ 3,simp:2]
```

and corresponding MIZAR proof:

```
  set L = -(D + -C);
A152:
now
  thus -(L + -(C + L)) = -(-(D + -(C + L)) + -((D + C) + L)) by A2, A20
                 .= -(-(D + -(C + L)) + -(D + (C + L))) by LATTICES:def 5
                 .= D by A3;
end;
```

OTTER, unlike MIZAR uses AC unification. That is why one of the steps had to be written explicitly (`LATTICES:def 5` is just the definition of attribute `join-associativity`). The symbol '`.=`' denotes here the so called *iterated equality*, the constant L was introduced to shorten the notation. Obviously, all the terms occuring in MIZAR proof were also generated by OTTER. One can implement interface between OTTER and MIZAR transforming all lines of proof into MIZAR. Such proofs can be shorten using enhancing mechanisms available in the MIZAR system, e.g. removing unnecessary lines of proof.

Unfortunately internal MIZAR utilities do not enable restructurization of proof in such degree as ILF does.

The readibility of ILF output is significantly higher (the formulas and the proofs in general are shorter) and is is more comfortable to follow it during making MIZAR proof by hand. What is especially important, following ILF proof we introduced some auxiliary notions and rules which can be find useful by other authors. This effects in better reusability of theorems.

For some concepts in [5] however the cost of formalization exceeds its current usefulness. For this reason, in fact we did not introduce e.g. functor nx and, consequently, a_n for arbitrary natural n, as in [5]. However, it may be corrected in future versions of [8].

6 Conclusions

The Mizar article [8] stems from at least three papers covering different steps of proof that all Robbins algebras are Boolean: in the first one [11] proofs are done by hand, the second [14] presents proofs as an output straight from EQP/Otter theorem prover, in the last one [5] automatically derived proofs have been simplified by introducing some auxiliary functions and concepts.

There is no guarantee (read: it is highly improbable) that all theorems are at the most general level and any of the assumptions cannot be weakened. Even in original work of Huntington one of the axioms has been dependent from the others and the corrected proof in [12] has been done without it.

One of the major problems is that all the lemmas or definitions should be stated for the minimal collection of conditions under which they are true. It is not easy if we have to reuse one's own theorems if they do not fulfill this condition. That is why Mizar articles which form a basis of the MML—like [18] had to be revised to enable more convenient formalization.

Mizar still lacks the tools which significantly restructurize proofs. It effects in low readability of some articles. Implementing interface between Otter, ILF and Mizar could be quite promising step in the direction of making proofs in the MML shorter, more readable and easier to understand. We hope that inheritance mechanisms for algebraic structures offered by Mizar could allow to make it work.

References

[1] Association of Mizar Users, Library Committee, *Preliminaries to structures,* available from `http://mizar.org/JFM/Addenda/struct_0.html`.

[2] G. Bancerek, *Development of the Theory of Continuous Lattices in Mizar,* [ed.] M. Kerber, M. Kohlhase, *Symbolic Computation and Automatic Reasoning. The Calculemus-2000 Symposium,* A K Peters, 2000, pp. 65–80.

[3] G. Birkhoff, *Lattice Theory,* 3rd ed., Amer. Math. Soc. Coll. Publ. 25, Providence, R. I., 1973.

[4] S. Burris, *An anthropomorphised version of the McCune's machine proof that Robbins algebras are Boolean,* preprint available by `http://www-unix.mcs.anl.gov/~mccune/papers/robbins/`.

[5] B. I. Dahn, *Robbins algebras are Boolean: A revision of McCune's computer-generated solution of Robbins problem,* Journal of Algebra, 208:526–532, 1998.

[6] B. Fitelson, *Using Mathematica 3.0 to understand the computer proof of the Robbins conjecture,* Math. Ed. Res. 7, No. 1, 1998.

[7] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. W. Mislove, D. S. Scott, *A Compendium of Continuous Lattices,* Springer-Verlag, Berlin, Heidelberg, 1980.

[8] A. Grabowski, *Robbins algebras vs. Boolean algebras,* Journal of Formalized Mathematics, Vol. 13, 2001, available from `http://mizar.org/JFM/Vol13/robbins1.html`.

[9] G. Grätzer, *General Lattice Theory,* Birkhäuser, 1978.

[10] E. V. Huntington, *Sets of independent postulates for the algebra of logic,* Trans. AMS, 5:288–309, 1904.

[11] E. V. Huntington, *New sets of independent postulates for the algebra of logic, with special reference to Whitehead and Russell's Principia Mathematica,* Trans. AMS, 35:274–304, 1933.

[12] E. V. Huntington, *Boolean algebra. A correction,* Trans. AMS, 35:557–558, 1933.

[13] E. Kusak, W. Leończuk, M. Muzalewski, *Abelian Groups, Fields and Vector Spaces*, Formalized Mathematics, 1(2):335–342, 1990, available also from `http://mizar.org/JFM/Vol1/vectsp_1.html`.

[14] W. McCune, *Solution of the Robbins problem,* Journal of Automated Reasoning, 19:263–276, 1997.

[15] P. Rudnicki, Ch. Schwarzweller, A. Trybulec, *Commutative Algebra in the Mizar System,* Journal of Symbolic Computation, 32(1/2):143–169, 2001.

[16] P. Rudnicki, A. Trybulec, *On equivalents of well-foundednesss. An experiment in* MIZAR, Journal of Automated Reasoning, 23:197–234, 1999.

[17] F. Wiedijk, *The de Bruijn factor,* available from `http://www.cs.kun.nl/~freek/factor/`.

[18] S. Żukowski, *Introduction to lattice theory,* Formalized Mathematics, 1(1), 215–222, 1990, available also from `http://mizar.org/JFM/Vol1/lattices.html`.