Proceedings of the Federated Conference on Computer Science and Information Systems pp. 245–248 DOI: 10.15439/2017F368 ISSN 2300-5963 ACSIS, Vol. 11

Introducing Euclidean Relations to Mizar

Adam Naumowicz, Artur Korniłowicz
Institute of Informatics, University of Białystok
ul. Ciołkowskiego 1 M, 15-245 Białystok, Poland
Email: {adamn, arturk}@math.uwb.edu.pl

***Abstract*—In this paper we present the methodology of implementing a new enhancement of the Mizar proof checker based on enabling special processing of Euclidean predicates, i.e. binary predicates which fulfill a specific variant of transitivity postulated by Euclid. Typically, every proof step in formal mathematical reasoning is associated with a formula to be proved and a list of references used to justify the formula. With the proposed enhancement, the Euclidean property of given relations can be registered during their definition, and so the verification of some proof steps related to these relations can be automated to avoid explicit referencing.**

I. Introduction

THE Mizar system [1], [2], [3] is a computer proof-assistant system used for encoding formal mathematical data (definitions and theorems) and verifying mathematical proofs. The system comprises a dedicated formal computer language – the Mizar language, a collection of command-line tools including the VERIFIER and a repository of formal texts – Mizar Mathematical Library (MML) that have been written in the Mizar language and machine-verified for their logical correctness by the VERIFIER.

The Mizar language preserves many features of natural language mathematical writing. It is designed to write declarative-style documents both readable for humans and effectively processed by computers.[1] The feature-rich language enables producing rigorous and semantically unambiguous texts. Apart from rules for writing traditional mathematical items (e.g. definitions, lemmas, theorems, proof steps, etc.) it also provides syntactic constructions to launch specialized algorithms for processing particular mechanisms (e.g. term identifications, term reductions [7], flexary connectives [8]) increasing computational power of VERIFIER (e.g. equational calculus [9], [10]). There have also been experiments on using specialized external systems to increase computational power of the Mizar system in selected domains [11], [12], [13].

Mizar allows registering various properties of predicates and functors [14] when defining new notions. Let us briefly recall that the set of currently implemented properties includes `involutiveness` and `projectivity` for unary operations, as well as `commutativity` and `idempotence` for binary operations. As far as binary predicates are concerned, which is directly related to the topic of this paper, the current version of the Mizar system supports registering `reflexivity`, `irreflexivity`, `symmetry`, `asymmetry` and `connectedness`. The `transitivity` property has been analyzed and implemented most recently [15].

[1]The legibility of proofs is a subject of ongoing research [4], [5], [6].

Table I presents how many registrations of predicate properties are used in the MML. The data has been collected with Mizar Version 8.1.05 working with the MML Version 5.37.1275.[2]

Property	Occurrences	Articles
`reflexivity`	138	91
`irreflexivity`	11	10
`symmetry`	122	82
`asymmetry`	6	6
`connectedness`	4	4
total	281	119

Table I
PREDICATE PROPERTIES OCCURRENCES

On the other hand, Table II shows the impact of registering predicate properties for proofs stored in the library as the number of errors occurring after removing registrations of the properties from texts, and the numbers of articles with such errors.

Property	Errors	Affected articles
`reflexivity`	356	44
`irreflexivity`	9	2
`symmetry`	498	47
`asymmetry`	6	4
`connectedness`	65	4
total	934	73

Table II
PREDICATE PROPERTIES IMPACT

In this paper we propose strengthening the Mizar system by implementing the representation of two new properties – `rightEuclidean` and `leftEuclidean`.[3] The properties are described in Section II. In Section III we present some examples of Euclidean properties found in the current MML. In Section IV we indicate several directions of further development of processing properties in Mizar.

II. The Euclidean Property

An example of a property which does not have its corresponding representation in Mizar yet is the Euclidean property.

[2]Computations were carried out at the Computer Center of University of Białystok http://uco.uwb.edu.pl

[3]An experimental version of the VERIFIER executable file for the Linux platform implementing the new features as well as other supplementary resources required for it to work with the current MML are available at http://alioth.uwb.edu.pl/~artur/euclidean/euclidean_ver.zip.

This property appears, most notably, in the set of axiomatic statements given at the start of Book I of Euclid's The Elements [16] as Common Notion 1, which states that: *Things which are equal to the same thing are also equal to each other.* [4] Formally speaking, a binary relation R on a set X is Euclidean (sometimes called right Euclidean) if it satisfies the following condition: $\forall a, b, c \in X\, (a\, R\, b \wedge a\, R\, c \rightarrow b\, R\, c)$.

Dually, a relation R on X may be called left Euclidean if $\forall a, b, c \in X\, (b\, R\, a \wedge c\, R\, a \rightarrow b\, R\, c)$.

The property of being Euclidean is a variant of transitivity, but the properties are indeed different. A transitive relation is Euclidean only if it is also symmetric. Only a symmetric Euclidean relation is transitive. A relation which is both Euclidean and reflexive is also symmetric and therefore it is an equivalence relation.

In the sequel we describe an enhancement of the Mizar system supporting automatic processing of Euclidean predicates. The automation involves specific computations performed during the verification process.[5]

To enable such an automation, when a new predicate is being defined, if it is Euclidean, it should be declared as such. The declaration is placed within a definitional block with the following Mizar syntax for right Euclidean:

```
definition
 let x1 be θ1, x2 be θ2, ..., xn be θn, y1,y2 be θn+1;
 pred π(y1,y2) means :ident:
  Φ(x1,x2,...,xn,y1,y2);
 rightEuclidean
 proof
  thus for a, b, c being θn+1
   st Φ(x1,x2,...,xn,a,b) & Φ(x1,x2,...,xn,a,c)
   holds Φ(x1,x2,...,xn,b,c);
 end;
end;
```

and left Euclidean, respectively:

```
definition
 let x1 be θ1, x2 be θ2, ..., xn be θn, y1,y2 be θn+1;
 pred π(y1,y2) means :ident:
  Φ(x1,x2,...,xn,y1,y2);
 leftEuclidean
 proof
  thus for a, b, c being θn+1
   st Φ(x1,x2,...,xn,b,a) & Φ(x1,x2,...,xn,c,a)
   holds Φ(x1,x2,...,xn,b,c);
 end;
end;
```

The statements of the formulas of correctness proofs shown in both the above definitions must be proved according to a special formula expressing the corresponding property of the defined predicate. For example, having such a definition, whenever VERIFIER meets a conjunction of formulas $\pi(a, b)$ and $\pi(a, c)$ within an inference, and the predicate π is known to be right Euclidean, then the set of premises in the inference is enlarged by the automatically generated formula $\pi(b, c)$, which may help to justify the proof step.

[4]https://proofwiki.org/wiki/Axiom:Euclid%27s_Common_Notions

[5]Other such automations are, for example, processing of adjectives [17] and definitional expansions [18].

It should be noted that the form of definitions and a corresponding correctness proof can in general be more complicated when the definition has several cases. Below is a formal representation of the correctness proof structure in the case of a definition with three explicit cases ($\Gamma_1(x_1, x_2, \ldots, x_n, y_1, y_2)$, $\Gamma_2(x_1, x_2, \ldots, x_n, y_1, y_2)$ and $\Gamma_3(x_1, x_2, \ldots, x_n, y_1, y_2)$) as well as the default case (introduced by **otherwise**):

```
definition
 let x1 be θ1, x2 be θ2, ..., xn be θn, y1,y2 be θn+1;
 pred π(y1,y2) means :ident:
  Φ1(x1,x2,...,xn,y1,y2) if Γ1(x1,x2,...,xn,y1,y2),
  Φ2(x1,x2,...,xn,y1,y2) if Γ2(x1,x2,...,xn,y1,y2),
  Φ3(x1,x2,...,xn,y1,y2) if Γ3(x1,x2,...,xn,y1,y2)
  otherwise Φn(x1,x2,...,xn,y1,y2);
 consistency;
 rightEuclidean
 proof
  thus for a, b, c being θn+1 st
   (
    (Γ1(x1,x2,...,xn,a,b) implies Φ1(x1,x2,...,xn,a,b)) &
    (Γ2(x1,x2,...,xn,a,b) implies Φ2(x1,x2,...,xn,a,b)) &
    (Γ3(x1,x2,...,xn,a,b) implies Φ3(x1,x2,...,xn,a,b)) &
    (not Γ1(x1,x2,...,xn,a,b) &
     not Γ2(x1,x2,...,xn,a,b) &
     not Γ3(x1,x2,...,xn,a,b) implies
       Φn(x1,x2,...,xn,a,b)) &
    (Γ1(x1,x2,...,xn,a,c) implies Φ1(x1,x2,...,xn,a,c)) &
    (Γ2(x1,x2,...,xn,a,c) implies Φ2(x1,x2,...,xn,a,c)) &
    (Γ3(x1,x2,...,xn,a,c) implies Φ3(x1,x2,...,xn,a,c)) &
    (not Γ1(x1,x2,...,xn,a,c) &
     not Γ2(x1,x2,...,xn,a,c) &
     not Γ3(x1,x2,...,xn,a,c) implies
       Φn(x1,x2,...,xn,a,c))
   ) holds
   (
    (Γ1(x1,x2,...,xn,b,c) implies Φ1(x1,x2,...,xn,b,c)) &
    (Γ2(x1,x2,...,xn,b,c) implies Φ2(x1,x2,...,xn,b,c)) &
    (Γ3(x1,x2,...,xn,b,c) implies Φ3(x1,x2,...,xn,b,c)) &
    (not Γ1(x1,x2,...,xn,b,c) &
     not Γ2(x1,x2,...,xn,b,c) &
     not Γ3(x1,x2,...,xn,b,c) implies
       Φn(x1,x2,...,xn,b,c))
   );
 end;
end;
```

The proof for a left Euclidean predicate with an analogous set of conditions would look like this:

```
definition
 let x1 be θ1, x2 be θ2, ..., xn be θn, y1,y2 be θn+1;
 pred π(y1,y2) means :ident:
  Φ1(x1,x2,...,xn,y1,y2) if Γ1(x1,x2,...,xn,y1,y2),
  Φ2(x1,x2,...,xn,y1,y2) if Γ2(x1,x2,...,xn,y1,y2),
  Φ3(x1,x2,...,xn,y1,y2) if Γ3(x1,x2,...,xn,y1,y2)
  otherwise Φn(x1,x2,...,xn,y1,y2);
 consistency;
 leftEuclidean
 proof
  thus for a, b, c being θn+1 st
   (
    (Γ1(x1,x2,...,xn,b,a) implies Φ1(x1,x2,...,xn,b,a)) &
    (Γ2(x1,x2,...,xn,b,a) implies Φ2(x1,x2,...,xn,b,a)) &
    (Γ3(x1,x2,...,xn,b,a) implies Φ3(x1,x2,...,xn,b,a)) &
    (not Γ1(x1,x2,...,xn,b,a) &
     not Γ2(x1,x2,...,xn,b,a) &
     not Γ3(x1,x2,...,xn,b,a) implies
       Φn(x1,x2,...,xn,b,a)) &
    (Γ1(x1,x2,...,xn,c,a) implies Φ1(x1,x2,...,xn,c,a)) &
    (Γ2(x1,x2,...,xn,c,a) implies Φ2(x1,x2,...,xn,c,a)) &
    (Γ3(x1,x2,...,xn,c,a) implies Φ3(x1,x2,...,xn,c,a)) &
```

```
      (not $\Gamma_1(x_1,x_2,\dots,x_n,c,a)$ &
       not $\Gamma_2(x_1,x_2,\dots,x_n,c,a)$ &
       not $\Gamma_3(x_1,x_2,\dots,x_n,c,a)$ implies
          $\Phi_n(x_1,x_2,\dots,x_n,c,a)$)
    ) holds
    (
      ($\Gamma_1(x_1,x_2,\dots,x_n,b,c)$ implies $\Phi_1(x_1,x_2,\dots,x_n,b,c)$) &
      ($\Gamma_2(x_1,x_2,\dots,x_n,b,c)$ implies $\Phi_2(x_1,x_2,\dots,x_n,b,c)$) &
      ($\Gamma_3(x_1,x_2,\dots,x_n,b,c)$ implies $\Phi_3(x_1,x_2,\dots,x_n,b,c)$) &
      (not $\Gamma_1(x_1,x_2,\dots,x_n,b,c)$ &
       not $\Gamma_2(x_1,x_2,\dots,x_n,b,c)$ &
       not $\Gamma_3(x_1,x_2,\dots,x_n,b,c)$ implies
          $\Phi_n(x_1,x_2,\dots,x_n,b,c)$)
    );
  end;
end;
```

III. Examples

To search for examples of predicates that fulfill the Euclidean properties one needs to use a parser capable of processing MML articles. For example, there is a free customizable parser[6] [19] implemented using the popular open-source GNU parser generator suite: `flex` and `bison` that gets in line with the free license on which the Mizar Mathematical Library is distributed [20]. It should be noted that the parser works with the "Weakly Strict Mizar" (WS-Mizar) representation of Mizar texts. With the current Mizar system one can generate a WS-Mizar document from a Mizar article using the `wsmparser` tool. Customized parsing actions allow filtering out theorems stated as implications resembling the Euclidean conditions, i.e. in which the antecedent contains a conjunction of two instances of predicative formulas and the consequent is also the same predicate (with appropriately instantiated variables). As a result, such theorems could be eliminated from the MML in order to avoid redundancy [21].

Our first example of a definition which naturally possesses the Euclidean property comes from a Mizar article about Tarski's classes and ranks [22].

```
definition
 let F,G be Relation;
 pred F,G are_fiberwise_equipotent means
 for x being object holds
  card Coim(F,x) = card Coim(G,x);
 reflexivity;
 symmetry;
end;
```

It should be noted that the properties of `reflexivity` and `symmetry` have already been identified and proved for this definition.

The Euclidean property of this definition is expressed as theorem `CLASSES1:76`:

```
theorem
 for F,G,H being Function
  st F,G are_fiberwise_equipotent &
   F,H are_fiberwise_equipotent holds
   G,H are_fiberwise_equipotent
```

[6]It can be downloaded from a dedicated Git repository https://github.com/MizarProject/wsm-tools. The distribution includes a simple Makefile for use with GNU make, which contains instructions to generate both the lexer and parser source code and build an executable called `wsm-parser` with customized syntactic actions.

The above theorem is quite popular (in total it is referenced 54 times in 12 various Mizar articles). In order to be possibly as general as in the original definition, i.e. the equipotence should be provable for arbitrary relations. This theorem can obviously be generated without any problems to this form:

```
theorem
 for F,G,H being Relation
  st F,G are_fiberwise_equipotent &
   F,H are_fiberwise_equipotent holds
   G,H are_fiberwise_equipotent
```

Another example of a theorem stating the right Euclidean property for a binary predicate is located in an article about midpoint algebras, in particular its theorem `MIDSP_1:21` [23]:

```
theorem Th21:
 p ## q & p ## r implies q ## r
```

This theorem does not have any references in the library yet.

Another example comes from a Mizar article devoted to parallelity spaces [24].

```
definition
 let PS be non empty ParStr;
 let a,b,c,d be Element of PS;
 pred a,b '||' c,d means
 [[a,b],[c,d]] in the CONGR of PS;
end;
```

Defining a new shortcut type for nonempty parallelity spaces and reserving the type for free variables including `a`, `b`, `c` and `d` to be used in the sequel

```
definition
 mode ParSp is ParSp-like non empty ParStr;
end;

reserve PS for ParSp,
 a,b,c,d for Element of PS;
```

the article provides the following theorem `PARSP_1:35`:

```
theorem Th35:
 a,b '||' a,c & a,b '||' a,d implies
  a,b '||' c,d
```

This theorem is referenced twice, only in the article introducing the notion. The above example is particularly interesting, because it shows that the Euclidean property can also be interpreted for predicates with more than two parameters. Namely, in this case the `'||'` predicate has four nominal arguments, but if we fix the first pair and consider only the last two, then the theorem clearly states a variant of the right Euclidean property.

IV. Conclusions and Further Work

This work falls under the continuous development of the Mizar system aimed at a still better representation of mathematical concepts taken for granted by working mathematicians in their writings. Following the addition of some automation for such commonly used relation properties like reflexivity, irreflexivity, symmetry, asymmetry, antisymmetry and most recently transitivity, we report on the experiment of implementing the left- and right- Euclidean properties. The number of Euclidean predicates detected in the current Mizar library is not big, but enriching the set of properties automatically processed by the VERIFIER is potentially useful for developing further formalizations in a more natural way without explicit references to theorems stating 'naturally obvious' properties. The availability of various properties might in future result in devising ways of handling collections of properties which often go together, e.g. the equivalence relation being a conjunction of three properties.

Another direction of future development might be connected with the extension of predicate properties for the ones which are not necessarily binary, but can be treated as such if we fix the values of their arguments. In general, properties known for binary predicates, can be introduced for n-ary predicates, where $2 \leq n$, with $n - 2$ fixed arguments.

REFERENCES

[1] A. Trybulec, "Mizar," in *The Seventeen Provers of the World*, ser. Lecture Notes in Computer Science, F. Wiedijk, Ed., vol. 3600. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 20–23. [Online]. Available: http://dx.doi.org/10.1007/11542384_4

[2] G. Bancerek, C. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pąk, and J. Urban, "Mizar: State-of-the-art and beyond," in *Intelligent Computer Mathematics – International Conference, CICM 2015, Washington, DC, USA, July 13–17, 2015, Proceedings*, ser. Lecture Notes in Computer Science, M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, Eds., vol. 9150. Springer, 2015, pp. 261–279. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-20615-8_17

[3] A. Grabowski, A. Korniłowicz, and A. Naumowicz, "Four decades of Mizar," *Journal of Automated Reasoning*, vol. 55, no. 3, pp. 191–198, October 2015. [Online]. Available: http://dx.doi.org/10.1007/s10817-015-9345-1

[4] K. Pąk, "Improving legibility of natural deduction proofs is not trivial," *Logical Methods in Computer Science*, vol. 10, no. 3, pp. 1–30, 2014. [Online]. Available: http://dx.doi.org/10.2168/LMCS-10(3:23)2014

[5] ——, "Automated improving of proof legibility in the Mizar system," in *Intelligent Computer Mathematics – International Conference, CICM 2014, Coimbra, Portugal, July 7–11, 2014. Proceedings*, ser. Lecture Notes in Computer Science, S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, Eds., vol. 8543. Springer, 2014, pp. 373–387. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08434-3_27

[6] ——, "Improving legibility of formal proofs based on the close reference principle is NP-hard," *Journal of Automated Reasoning*, vol. 55, no. 3, pp. 295–306, October 2015. [Online]. Available: http://dx.doi.org/10.1007/s10817-015-9337-1

[7] A. Korniłowicz, "On rewriting rules in Mizar," *Journal of Automated Reasoning*, vol. 50, no. 2, pp. 203–210, February 2013. [Online]. Available: http://dx.doi.org/10.1007/s10817-012-9261-6

[8] ——, "Flexary connectives in Mizar," *Computer Languages, Systems & Structures*, vol. 44, pp. 238–250, December 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cl.2015.07.002

[9] G. Nelson and D. C. Oppen, "Fast decision procedures based on congruence closure," *J. ACM*, vol. 27, pp. 356–364, April 1980. [Online]. Available: http://doi.acm.org/10.1145/322186.322198

[10] A. Grabowski, A. Korniłowicz, and C. Schwarzweller, "Equality in computer proof-assistants," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, 2015, pp. 45–54. [Online]. Available: http://dx.doi.org/10.15439/2015F229

[11] A. Naumowicz, "Interfacing external CA systems for Gröbner bases computation in Mizar proof checking," *International Journal of Computer Mathematics*, vol. 87, no. 1, pp. 1–11, January 2010. [Online]. Available: http://dx.doi.org/10.1080/00207160701864459

[12] ——, "SAT-enhanced Mizar proof checking," in *Intelligent Computer Mathematics – International Conference, CICM 2014, Coimbra, Portugal, July 7–11, 2014. Proceedings*, ser. Lecture Notes in Computer Science, S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, Eds., vol. 8543. Springer, 2014, pp. 449–452. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08434-3_37

[13] ——, "Automating Boolean set operations in Mizar proof checking with the aid of an external SAT solver," *Journal of Automated Reasoning*, vol. 55, no. 3, pp. 285–294, October 2015. [Online]. Available: http://dx.doi.org/10.1007/s10817-015-9332-6

[14] A. Naumowicz and C. Byliński, "Improving Mizar texts with properties and requirements," in *Mathematical Knowledge Management, Third International Conference, MKM 2004 Proceedings*, ser. MKM'04, Lecture Notes in Computer Science, A. Asperti, G. Bancerek, and A. Trybulec, Eds., vol. 3119, 2004, pp. 290–301. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-27818-4_21

[15] A. Korniłowicz, "Enhancement of Mizar texts with transitivity property of predicates," in *Intelligent Computer Mathematics – 9th International Conference, CICM 2016, Bialystok, Poland, July 25–29, 2016, Proceedings*, ser. Lecture Notes in Computer Science, M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., vol. 9791. Springer, 2016, pp. 157–162. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-42547-4_12

[16] Euclid, *The Elements, books I-XIII.* New York: Barnes & Noble, 2006.

[17] A. Naumowicz, "Enhanced processing of adjectives in Mizar," in *Computer Reconstruction of the Body of Mathematics*, ser. Studies in Logic, Grammar and Rhetoric, A. Grabowski and A. Naumowicz, Eds. University of Białystok, 2009, vol. 18(31), pp. 89–101.

[18] A. Korniłowicz, "Definitional expansions in Mizar," *Journal of Automated Reasoning*, vol. 55, no. 3, pp. 257–268, October 2015. [Online]. Available: http://dx.doi.org/10.1007/s10817-015-9331-7

[19] A. Naumowicz and R. Piliszek, "Accessing the Mizar library with a weakly strict Mizar parser," in *Intelligent Computer Mathematics – 9th International Conference, CICM 2016, Bialystok, Poland, July 25–29, 2016, Proceedings*, ser. Lecture Notes in Computer Science, M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., vol. 9791. Springer, 2016, pp. 77–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-42547-4_6

[20] J. Alama, M. Kohlhase, L. Mamane, A. Naumowicz, P. Rudnicki, and J. Urban, "Licensing the Mizar Mathematical Library," in *Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics*, ser. MKM'11, Lecture Notes in Computer Science, J. H. Davenport, W. M. Farmer, J. Urban, and F. Rabe, Eds., vol. 6824. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 149–163. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22673-1_11

[21] A. Grabowski and C. Schwarzweller, "On duplication in mathematical repositories," in *Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, MKM 2010, Paris, France, July 5–10, 2010. Proceedings*, ser. Lecture Notes in Computer Science, S. Autexier, J. Calmet, D. Delahaye, P. D. F. Ion, L. Rideau, R. Rioboo, and A. P. Sexton, Eds., vol. 6167. Springer, 2010, pp. 300–314. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14128-7_26

[22] G. Bancerek, "Tarski's classes and ranks," *Formalized Mathematics*, vol. 1, no. **3**, pp. 563–567, 1990. [Online]. Available: http://fm.mizar.org/1990-1/pdf1-3/classes1.pdf

[23] M. Muzalewski, "Midpoint algebras," *Formalized Mathematics*, vol. 1, no. **3**, pp. 483–488, 1990. [Online]. Available: http://fm.mizar.org/1990-1/pdf1-3/midsp_1.pdf

[24] E. Kusak, W. Leończuk, and M. Muzalewski, "Parallelity spaces," *Formalized Mathematics*, vol. 1, no. **2**, pp. 343–348, 1990. [Online]. Available: http://fm.mizar.org/1990-1/pdf1-2/parsp_1.pdf

[25] M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., *Intelligent Computer Mathematics – 9th International Conference, CICM 2016, Bialystok, Poland, July 25–29, 2016, Proceedings*, ser. Lecture Notes in Computer Science, vol. 9791. Springer, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-42547-4

[26] S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, Eds., *Intelligent Computer Mathematics – International Conference, CICM 2014, Coimbra, Portugal, July 7–11, 2014. Proceedings*, ser. Lecture Notes in Computer Science, vol. 8543. Springer, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08434-3