Readable Formalization of Euler's Partition Theorem in Mizar*

Karol Pąk

Institute of Computer Science,
University of Bialystok, Poland
pakkarol@uwb.edu.pl

Abstract. We present a case study on formalization of a textbook theorem in a form that is as close to the original textbook presentation as possible. Euler's partition theorem, listed as #45 at Freek Wiedijk's list of "Top 100 mathematical theorems", is taken as the subject of the study. As a result new formal concepts including informal flexary (i.e. flexible arity) addition are created and existing ones are extended to go around existing limitations of the Mizar system, without modification of its core. Such developments bring more flexibility of informal language reasoning into the Mizar system and make it useful for wider audience.

Key words: Operations on languages, Legibility of proofs, Euler's partition

1 Introduction

Famous mathematical theorems rarely occur with only one proof in informal mathematical practice. In the mathematical literature we can often find several formulations or even conceptually different proofs of the same theorem. However, the reader can easily compare proofs that have the same main idea. Such situations are not so popular in repositories of formal mathematical knowledge. Usually, one version of a theorem with one proof only is stored there. Additionally, comparing proofs created in different formal proof systems is not so trivial. Even, if we consider two declarative environments or two procedural ones, this problem does not seem much easier.

It comes as no surprise that the main idea of the formal proof is often different from all known informal proof variants of the theorem, even if the author tried to create a formal equivalent of a particular informal development. The experience of big proof formalization developments shows that proof script authors can often, given the set of definitions and theorems collected in the Mizar Mathematical Library (MML) [3], obtain a new, so far unknown, and sometimes simpler, proof of a particular statement [8, 9]. Therefore, many authors compare informal proof variants to check the possible use of collected resources before starting their formalization effort.

* The paper has been financed by the resources of the Polish National Science Center granted by decision n°DEC-2012/07/N/ST6/02147.

To illustrate such situations we can consider the development of Brouwer's fixed-point theorem [5]. This theorem has been used in the formalization of Jordan curve theorem in the Mizar [7]. But for this purpose, the 2-dimensional case was enough. Therefore, this statement has been provided by A. Korniłowicz, only for this case using basic arguments concerning the fundamental groups of the respective spaces [13]. Note that this approach for higher-dimensional cases requires incomparably more difficult facts about these groups. The same theorem was proved in a combinatorial way in the HOL Light by J. Harrison for n-dimensional case, based on Sperner's lemma [10]. The original approach to prove this lemma is based on intuitively clear facts about the standard n-dimensional simplex and its arbitrarily small subdivision. However, these facts are not so easy if we consider them formally. Therefore, he chose an alternative justification of this lemma, wherein the simplex structure is replaced by the cubic one. Note that simplex structure is explored in one of the approaches to prove Brouwer's invariance of the domain theorem that was selected to formalize in Mizar by K. Pąk [18]. Therefore, having a large collection of facts about simplices, Brouwer's fixed-point theorem has been redeveloped to the general case based on Sperner's lemma in the original approach.

In this paper, we present the results of an experiment where we formalize Euler's partition theorem in the original approach [1, 6, 22]. Obviously, we can obtain a very slick proof using definitions and theorems collected in the MML that looks more or less similar to the original proof. However, the point of this exercise was not to obtain "a formalization", but to see how a natural language proof can be expressed in the Mizar format. Therefore our aim was to recreate the main idea and steps of reasoning as closely as possible, sometimes work around the system's limitations, however without a modification of its core, to obtain the result that looks almost the same as the informal one. Furthermore, as a measure of "closeness", we consider also the sketch of the proof that is generated automatically from the Mizar proof scripts and is published in the journal *Formalized Mathematics*.

Structure of the paper In Section 2 we discuss several conceptually different proofs of Euler's partition theorem. We focus our attention on three approaches: the original one that was presented by L. Euler [6], the *Euler's bijective proof* that was presented by G.E. Andrews [1], and the approach basing on Sylvester's bijection created by J.J. Sylvester, and choose one. In Section 3 we analyze informal mathematical constructions that are used in the selected approach, and we propose an adaptation method of this construction to the formal language in a way that the obtained visual effect is as closely as possible to the informal one. In Section 4 we present a formalization of the proof in the selected approach written in the Mizar system. Finally, in Section 5 we conclude the paper and we discuss future work. Note additionally that each fragment of the Mizar proof scripts contained in this paper comes from Mizar theory files `FLEXARY1.miz`, `EULRPART.miz` available in the Mizar distribution.

2 Informal proofs of Euler's partition theorem

Generally, a partition of a natural number n is a way of writing n as a sum of positive integers where the arrangement of the addends does not need to be determined. Denote by $\mathcal{O}_n$ the set of partitions of n into odd parts and similarly denote by $\mathcal{D}_n$ the set of partitions of n into distinct parts. Then Euler's partition theorem states that the cardinality of $\mathcal{O}_n$ is equal to the cardinality of $\mathcal{D}_n$ for all natural n. Euler presented a very slick proof in 1748 [6] by generating functions that can be sketched as follows:

$$\begin{aligned} 1+\sum_{n=1}^{\infty}|\mathcal{D}_n|x^n &= (1+x)\cdot(1+x^2)\cdot(1+x^3)\cdot(1+x^4)\cdot(1+x^5)\cdot\ldots \\ &= \frac{1-x^2}{1-x}\cdot\frac{1-x^4}{1-x^2}\cdot\frac{1-x^6}{1-x^3}\cdot\frac{1-x^8}{1-x^4}\cdot\frac{1-x^{10}}{1-x^5}\cdot\ldots \\ &= \frac{1}{1-x}\cdot\frac{1}{1-x^3}\cdot\frac{1}{1-x^5}\cdot\ldots \\ &= (1+x+x^2+x^3+\ldots)\cdot(1+x^3+(x^3)^2+(x^3)^3+\ldots) \\ &\qquad\cdot(1+x^5+(x^5)^2+(x^5)^3+\ldots)\cdot\ldots \\ &= 1+\sum_{n=1}^{\infty}|\mathcal{O}_n|x^n \end{aligned} \tag{1}$$

However, in such an analytical proof, information that describes the relationship between relevant partitions of n is implicit and hard to grasp. Obviously, without this information the proof is complete, but there are many people who prefer to compare the cardinality of sets based on an explicit mapping that associates their elements. Such a bijective proof has also been described by Euler. It has been given by G.E. Andrews [1, pp. 149-150], and also by H.S. Wilf [22, p. 10] in the following form:

Euler's bijective proof: A partition into distinct parts can be written as

$$n = d_1 + d_2 + \ldots + d_k. \tag{3}$$

Each integer d_i can be uniquely expressed as a power of 2 times an odd number. Thus, $n = 2^{a_1}O_1 + 2^{a_2}O_2 + 2^{a_3}O_3 + \ldots + 2^{a_k}O_k$ where each O_i is an odd number. If we now group together the odd numbers we get an expression like:

$$\begin{aligned} n &= (2^{\alpha_1}+2^{\alpha_2}+\ldots)\cdot 1 + (2^{\beta_1}+2^{\beta_2}+\ldots)\cdot 3 + (2^{\gamma_1}+2^{\gamma_2}+\ldots)\cdot 5 + \ldots \\ &= \mu_1\cdot 1 + \mu_3\cdot 3 + \mu_5\cdot 5 + \ldots \end{aligned}$$

In each series $(2^{\alpha_1}+2^{\alpha_2}+\ldots)$, the α_i's are distinct (why?). Thus the sum is the binary expansion of some μ_j. We now see the partition of n into odd parts that corresponds, under this bijection, to the given partition (3) into distinct parts. It is the partition that contains μ_1 1's, μ_3 3's, etc. □

Fig. 1. A bijective proof of Euler's partition theorem that is used in [22].

In this constructive proof only several simple facts are used (implicite): the existence and the uniqueness of conversion between a natural number and its

binary equivalent; a positive number decomposition as the product of a power of two and an odd number.

It is also possible to prove this result by another partition transformation, where correspondence is visually apparent by some modification of the Ferrers diagram, called a *bent graph*. The construction of this graph is based on the observation that we can represent every odd number $2i+1$ as one central point, and a column and a row that are built with i points. Then we may arrange the configurations of points in such a way that the central points are inserted into a diagonal line. In this way we obtain what we call the bent graph (see Fig. 2). Additionally, the symmetrical parts of this graph, located above and below the diagonal line, are called *regular graphs*. Sylvester's bijection that is defined on bent graphs has been presented in [20, pp. 287–288] and is formulated as follows: *Each of these graphs will be bounded by lines inclined to each other at an angle one-half of that contained between the original bounding lines, and each may be regarded as made up of bends fitting into one another.* The original proof also contains a sketch of a justification that the resulting partition consists of pairwise different numbers (see to the example presented in Fig. 2 as an illustration). Obviously, the justification presented in this form shows in a simple way the

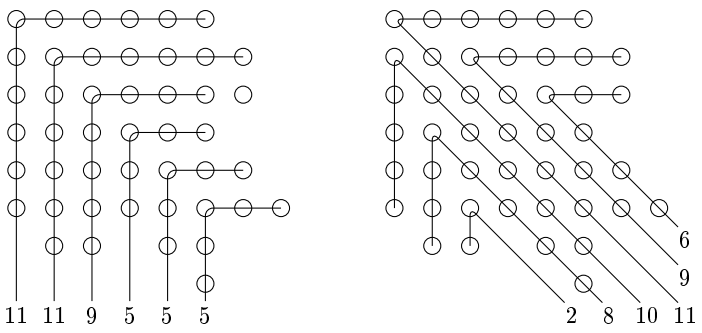

Fig. 2. An example that illustrates Sylvester's bijection, which maps a partition of 46 into odd numbers $(5,5,5,9,11,11)$ to a partition of 46 into pairwise different numbers $(2,6,8,9,10,11)$, used in [20].

proof idea, but only the idea. Therefore, a formalization of such a reasoning could not reflect the original proof at a very high level of similarity.

Analysis of well-known justifications of Euler's partition theorem for the purposes of formal transcription shows that reflecting the original proof is possible in two mentioned approaches. However, the second justification, presented in Fig. 1, contains a more interesting informal construction, namely flexary (i.e. flexible arity, see [12]) addition with visible lower and upper bounds of summation, but also with only a lower bound available. Additionally, the formalization

of this theorem in the HOL system imitates also the main idea of the second approach[1].

3 Formal introduction of informal notations

To reflect the informal reasoning in the Mizar language, we discuss in this section definitions and notations used in the reasoning presented in Fig. 1. We divide this discusion into three subsections. In Subsection 3.1 we choose concepts of a partition definition. Next, in Subsection 3.2 we define operators that reflect in the Mizar system the informal flexary plus in both cases, finite and infinite. At the end, in Subsection 3.3 we define a special kind of a matrix generalization to realize the rearrangement of the values of a finite sequence into a sequence of sequences that is used in the considered reasoning.

3.1 The formal definition of a partition

The reasoning presented in Fig. 1 uses a partition of a natural number n (see (3)) defined as a finite sequence of positive integers that sum up to n. Note also that the order of the addends is not indicated. However, to count partitions of a number we need to opt for some type of arrangement, non increasing or non decreasing. The alternative solution is to represent a partition as a sequence of addends frequency, i.e. a sequence $(a_1, a_2, \ldots)$ that represent the partition $\{\underbrace{1, 1, \ldots, 1}_{a_1}, \underbrace{2, 2, \ldots, 2}_{a_2}, \ldots\}$. Obviously, we find this approach in the formulation *"the partition that contains μ_1 1's, μ_3 3's"* where the non-decreasing arrangement is suggested. Therefore, to simplify, we use only one method of arrangement in the definition of partition, which is formulated as a non-decreasing finite sequence of non-zero natural numbers that sum up to n, thus obtained by writing (in Mizar)

```
definition
   let n be Nat;
   mode a_partition of n -> non-zero non-decreasing natural-valued
      FinSequence means
         Sum it = n;
end;
```
(2)

Obviously, such a definition is adapted to represent simple modifications of partitions in an intuitive way, but requires a special attention in the formal approach. Note that we can neither simply include elements to a partition, nor modify its existing elements, without violating the arrangement. Such problems do not occur if we consider the *frequency representation* of partitions. In this case the realization of the mentioned operations on a partition is reduced to a simple modification that increases or reduces by one the value of one element in the frequency sequence. This approach has been used in the formalization of Euler's theorem in HOL, where the partition is defined as follows:

[1] For more details see `https://code.google.com/p/hol-light/source/browse/trunk/100/euler.ml?r=2`.

```
let partitions = store_name "partitions" new_definition
 ‘p partitions n <=> (!i. ~(p i = 0) ==> 1 <= i /\ i <= n) /\
                     nsum(1..n) (\i. p(i) * i) = n‘;;
```

3.2 The flexary plus

Analyzing the first sentence of the proof, we find the equation (3) where we face two formalization problems. Obviously, the informal mathematical operators are present here, i.e. the flexary plus. This formula is usually formalized in the equivalent form $n = \sum_{i=1}^{k} d_i$. It can be written in the Mizar language as `n = sum d`, if we additionally assume that `k` is the length of d, or it can be written as `n = sum (d|k)`, where `k` is an arbitrary natural number, `d` is a finite sequence of natural numbers, and `d|k` is the restriction of `d` to the set $\{1, 2, \ldots, k\}$. However, in our experiment we want to obtain a similar term represented as:

```
(d,2)+...+(d,k).                                                    (4)
```

Note also that the informal expression $d_2+\ldots+d_k$ contains a hidden information that the finite sequence d has a second, a third, and up to a k-th element in the domain. The method of hiding this information in a term as (4) or getting around this problem is the second formalization problem. In the Mizar language we can resolve this, e.g. by creating a definition by cases or summarizing only these values that correspond to arguments in the domain of a finite sequence. In our experiment, we use a solution that gives the greater flexibility. The solution is based on the concept developed in the MML: a permissive definition of the function value, where it is assumed that the empty set is the value of a function outside its field, and also on the Peano number approach, where the empty set equals 0. However, this solution can be applied to summation of such D-valued sequence if D contains 0. We define the flexary plus as follows:

```
definition
   let k,n;
   let f,g be complex-valued Function;
   func (f,k) +...+ (g,n) -> complex number means
      h.(0+1) = f.(0+k) & ... & h.(n-'k+1) = f.(n-'k+k)                (5)
         implies
      it = Sum (h| (n-'k+1)) if f = g & k <= n
                             otherwise it = 0;
end;
```

where `h` is a complex valued finite sequence, the operation `-'` is the limited substraction of natural numbers, i.e. $a-'b$ is equal to $\max\{a-b, 0\}$, `& ... &` is the flexary logical conjunction (for more details see [11]). Note that the Mizar system's limitations prohibit the repetition of a locus in an operator expression when it is defined. Therefore, we cannot eliminate the function `g`, even if we want to consider only the case `f = g`.

The value of the defined above flexary plus is a complex number, but in the reasoning presented in Fig. 1 only the natural valued finite sequences are used,

for which this value should be a natural number. To obtain such information about the value, the sufficient solution is to have the following registration in the environment of the Mizar article:

```
registration
   let n,k;
   let f be natural-valued FinSequence;
   cluster (f,k) +...+ (f,n) -> natural;
end;
```
(6)

Based on the flexary plus definition, we can formulate and prove the first equality presented in Fig. 1 as `n = d.1 + (d,2)+...+(d,len d)` for an arbitrary partition `d` of `n`, where `len d` is the length of `d`. Additionally, if we consider finite sequences `a`, `O` that represent the unique decomposition of `d` as a power of 2 times an odd number, we can formalize also the second equality as:

```
n = 2|^(a.1) * O.1 + 2|^(a.2) * O.2 +
                (O (#) 2|^a,3) +...+ (O (#) 2|^a,len d),
```
(7)

where `(#)` represents the product of functions and `a`, `O` have already been introduced to the reasoning in the preceding step which reads as follows:

```
consider O be odd-valued FinSequence,
         a be natural-valued FinSequence such that
   A1: len O = len d = len a & d = O (#) 2|^a and
   A2: d.1 = O.1*(2|^a.1) &...& d.len d = O.len d*(2|^a.len d);
```
(8)

Observe that the formula labeled by `A2` is an equivalent formulation of the statement `d = O (#) 2|^a`. This statement has been added only for improving the readability of dependencies occurring between `d`, `O`, and `a`. Note also that we could prove the equality (7) without the restriction on the length of `d` (that is equal to the length of `O` and `a`), e.g. for `len O` equal `0` we obtain simply that `O.1 = O.2 = 0`, since `1`, `2` do not belong to the domain of `O`, but also `n = 0`, since `0`-length sequence can be only the partition of `0`.

Analyzing the next part of the reasoning presented in Fig. 1 we can observe that the flexary plus is used also in an unbounded form, without the upper bound of summation. Generally, this operation is used in the informal mathematical practice to speak conveniently about the sum of the terms of a sequence, where basing on several first terms we can precisely predict the others elements by analogy. Additionally, according to a popular informal convention, the information about the convergence of a sequence is often assumed *a priori*. However, this issue does not concern the reasoning presented in Fig. 1, where the unbounded flexary plus is used only in the context of finite sequences. For such kind of sequence, we can define this operator as the flexary plus with an upper bound, where the upper bound is greater than or equal to the maximum of the domain of the sequence. It is obtained by writing:

```
definition
   let n;
   let f be complex-valued Function;
   assume dom f /\ NAT is finite;
   func (f,n)+... -> complex number means
      for k st for i st i in dom f holds i <= k holds
         it = (f,n) +...+ (f,k);
end;
```
(9)

Note that in the considered reasoning, only finite sequences are used, where the intersection of the domain and $\mathbb{N}$ is finite. Therefore, the introduction of such a definition to our experiment seems to be redundant. However, without this assumption, we cannot use this operator in the Mizar system if we need to substitute a term that is *a priori* a finite sequence on `f`, but we have not proved this statement yet. Obviously, such a possibility is very useful, if we want to formulate similarly a formal equivalents of an informal term. Moreover, we can reinforce the Mizar checker in such a way that the equality

```
(f,n)+... = (f,n) +...+ (f,len f);
```
(10)

is automatically generated and added to every justification of a step, where the expression `+...` is used and a term substituted for `f` is a finite sequence. For this purpose we create the following redefinition:

```
definition
   let n;
   let f be complex-valued FinSequence;
   redefine func (f,n)+... -> complex number equals
      (f,n) +...+ (f,len f);
end;
```
(11)

We are aware that a definition that can generate automatically the summed sequence based only on the terms on endpoints (in a finite case) or two consecutive terms (in an infinite case) is a more interesting solution. However, such a solution requires a modification of the core Mizar system, as it has been done in the case of flexary logical operators for generalized conjunction and alternative (for more details see [11]). Such a solution goes beyond the point of this study.

3.3 Regrouping the values of sequence

In the considered reasoning, we come across another interesting informal procedure that consists of grouping the odd number. Obviously, partitioning of a set into non-empty subsets, according to some properties of its members is nothing new. However, to improve the readability of the defined partition, the proof authors add to the reasoning some exemplifications or even write elements of several members in such a way that the reader can easily find out other members by analogy. Note that such kind of exemplification is important for humans, but generally is unnecessary for the Mizar checker, except from the case where the existence of some kind of partition is proved.

Therefore, specially for our experiment, we define a specific kind of a finite sequence of finite sequences (a matrix generalization) over an odd valued finite sequence O, denoted in the Mizar language by odd_organization **of** O. This map is defined in such a way that the first sequence contains all arguments of O for which the value equal 1, the second sequence contains all arguments of O for which the value equal 3, etc., where the number of finite sequences is sufficient to cover the domain of O. Obviously, odd_organization **of** O is not uniquely determined by O, but we can use the global choice [14] (for more details see [7]).

Note that we can define odd_organization "directly", i.e. without subtypes and attributes. However, if another user of the MML will need to regroup the values in a different way, then probably he would have to provide some analogous properties. To avoid such situations in the MML, new types are defined as a restricted version of existing, more general types, if only the last ones exist. Therefore, we define the odd_organization in the following way. First we note that individual finite sequences in odd_organization have to be injective and determined, disjoint sets of values. Hence we introduce the following attribute:

```
definition
   let F be Function-yielding Function;
   attr F is double-one-to-one means
      for x1,x2,y1,y2 be object st
            x1 in dom F & y1 in dom (F.x1) &
            x2 in dom F & y2 in dom (F.x2) & F_(x1,y1)=F_(x2,y2)
      holds x1 = x2 & y1 = y2;
end;
```
(12)

and a mode that reorganizes a finite set D into a finite sequence of finite sequences:

```
definition
   let D be finite set;
   mode DoubleReorganization of D -> double-one-to-one FinSequence of D*
      means Values it = D;
end;
```
(13)

Then we define a type where we have that in every individual finite sequence, elements are mapped to the same value, and such values in different finite sequences are different:

```
definition
   let f be finite Function;
   mode valued_reorganization of f -> DoubleReorganization of dom f means
   (for n ex x st
      x = f.it_(n,1) & ... & x = f.it_(n,len (it.n))) &
   for n1,n2,i1,i2 be Nat st
      i1 in dom (it.n1) & i2 in dom (it.n2) &
      f.it_(n1,i1) = f.it_(n2,i2)
   holds n1 = n2;
end;
```
(14)

and finally we define odd_organization as follows:

```
definition
   let f be odd-valued FinSequence;
   mode odd_organization of f -> valued_reorganization of f means
   2*n-1 = f.it_(n,1) & ... & 2*n-1 = f.it_(n,len (it.n));
end;
```
(15)

Based on this approach, we can prove in a more general form the properties of odd_organization that are needed to justify steps in the considered theorem. Obviously this approach is much more difficult, but is consistent with the popularized direction of the development of the MML. According to this direction, the legible formulation and proving of a theorem is an important and challenging aim, when proof scripts are created for further development of the MML. However, no less important in this direction is extraction of definitions, creation of auxiliary theorems and notations in such a way that MML users will be able to adapt this knowledge for their own purposes.

4 The theorem formalization

In the reasoning presented in Fig. 1 the first and also the biggest part is the description of the transformation that maps a partition of a number into odd parts to a partition of the number into distinct parts. To adapt this fragment in a Mizar proof script we define this transformation as follows:

```
definition
   let n be Nat;
   let p be one-to-one a_partition of n;
   func Euler_transformation p -> odd-valued a_partition of n means
```
(16)

where the value denoted by it can be determined by the condition:

```
for O be odd-valued FinSequence,a be natural-valued FinSequence,
    sort be odd_organization of O st
      len O = len p = len a & p = O (#) 2|^a
for j holds card Coim(it,j*2-1) = ((2|^a)*.sort.j,1)+...
```
(17)

However, we decided on a more descriptive definition, where several formulas occur that describe some "exemplifications", only to improve the legibility of obtained condition. Note that the equivalence of such extended definition, presented below, with above ones is provided [15, (12)].

```
for j be Nat,O1 be odd-valued FinSequence,a1 be natural-valued FinSequence st
   len O1 = len d = len a1 & d = O1 (#) 2|^a1
for sort1 be DoubleReorganization of dom d st
   (1 = O1.sort1_(1,1) & ... & 1 = O1.sort1_(1,len (sort1.1))) &
   (3 = O1.sort1_(2,1) & ... & 3 = O1.sort1_(2,len (sort1.2))) &
   (3 = O1.sort1_(3,1) & ... & 5 = O1.sort1_(3,len (sort1.3))) &
for i holds
   2*i-1 = O1.sort1_(i,1) & ... & 2*i-1 = O1.sort1_(i,len (sort1.i))
holds
   card Coim(it,1) = (2|^a1).sort1_(1,1)+((2|^a1)*.sort1.1,2)+... &
   card Coim(it,3) = (2|^a1).sort1_(2,1)+((2|^a1)*.sort1.2,2)+... &
   card Coim(it,5) = (2|^a1).sort1_(3,1)+((2|^a1)*.sort1.3,2)+... &
   card Coim(it,j*2-1) = (2|^a1).sort1_(j,1)+((2|^a1)*.sort1.j,2)+...
```
(18)

Obviously to prove the correctness of this definition in the Mizar system, we have to justify that such a value exists and is unique. We selected the proof of the first conditions [15, (11)] to formally represent the informal description of Euler's transformation. For this aim, we constructed a reasoning, wherein all steps that are located on the first level of nesting (for more details see [7]) correspond to the selected fragments of the informal proof. Additionally, as a measure of correspondence we can analyze the generated automatically sketch of this reasoning presented in Fig. 3. Note that the full proof contains about 300 lines (for the full description see [15]), therefore we hide all nested reasonings and every list of statements that is used in a justification.

(11) Let us consider a one-to-one partition d of n. Then there exists an odd-valued partition e of n such that for every natural number j for every odd-valued finite sequence O_1 for every natural-valued finite sequence a_1 such that $\operatorname{len} O_1 = \operatorname{len} d = \operatorname{len} a_1$ and $d = O_1 \cdot 2^{a_1}$ for every double reorganization τ of $\operatorname{dom} d$ such that $1 = O_1(\tau_{1,1})$ and ... and $1 = O_1(\tau_{1,\operatorname{len}(\tau(1))})$ and $3 = O_1(\tau_{2,1})$ and ... and $3 = O_1(\tau_{2,\operatorname{len}(\tau(2))})$ and $5 = O_1(\tau_{3,1})$ and ... and $5 = O_1(\tau_{3,\operatorname{len}(\tau(3))})$ and for every i, $2 \cdot i - 1 = O_1(\tau_{i,1})$ and ... and $2 \cdot i - 1 = O_1(\tau_{i,\operatorname{len}(\tau(i))})$ holds $\overline{\overline{\operatorname{Coim}(e,1)}} = 2^{a_1}(\tau_{1,1}) + ((2^{a_1} \odot \tau)(1), 2) + \ldots$ and $\overline{\overline{\operatorname{Coim}(e,3)}} = 2^{a_1}(\tau_{2,1}) + ((2^{a_1} \odot \tau)(2), 2) + \ldots$ and $\overline{\overline{\operatorname{Coim}(e,5)}} = 2^{a_1}(\tau_{3,1}) + ((2^{a_1} \odot \tau)(3), 2) + \ldots$ and $\overline{\overline{\operatorname{Coim}(e, j \cdot 2 - 1)}} = 2^{a_1}(\tau_{j,1}) + ((2^{a_1} \odot \tau)(j), 2) + \ldots$.
PROOF: $n = d(1) + ((d,2) + \ldots + (d, \operatorname{len} d))$ by [16, (22)]. Consider O being an odd-valued finite sequence, a being a natural-valued finite sequence such that $\operatorname{len} O = \operatorname{len} d = \operatorname{len} a$ and $d = O \cdot 2^a$ and $d(1) = O(1) \cdot 2^{a(1)}$ and ... and $d(\operatorname{len} d) = O(\operatorname{len} d) \cdot 2^{a(\operatorname{len} d)}$. $n = 2^{a(1)} \cdot O(1) + 2^{a(2)} \cdot O(2) + ((O \cdot 2^a, 3) + \ldots + (O \cdot 2^a, \operatorname{len} d))$ by [16, (20)], [21, (25)]. Reconsider $\sigma =$ the odd organization of O as a double reorganization of $\operatorname{dom} 2^a$. Consider μ being a $(2 \cdot \operatorname{len} \sigma)$-element finite sequence of elements of $\mathbb{N}$ such that for every j, $\mu(2 \cdot j) = 0$ and $\mu(2 \cdot j - 1) = 2^a(\sigma_{j,1}) + ((2^a \odot \sigma)(j), 2) + \ldots$. Set $\alpha = a \cdot \sigma(1)$. Set $\beta = a \cdot \sigma(2)$. Set $\gamma = a \cdot \sigma(3)$. $n = (2^\alpha(1) + (2^\alpha, 2) + \ldots) \cdot 1 + (2^\beta(1) + (2^\beta, 2) + \ldots) \cdot 3 + (2^\gamma(1) + (2^\gamma, 2) + \ldots) \cdot 5 + ((\mathrm{id}_{\operatorname{dom}\mu} \cdot \mu), 7) + \ldots$ by [21, (29)], [16, (41)], [21, (25)], [4, (12)]. $n = \mu(1) \cdot 1 + \mu(3) \cdot 3 + \mu(5) \cdot 5 + ((\mathrm{id}_{\operatorname{dom}\mu} \cdot \mu), 7) + \ldots$ by [16, (42), (41), (25)]. Consider e being an odd-valued finite sequence such that e is non-decreasing and for every i, $\overline{\overline{\operatorname{Coim}(e,i)}} = \mu(i)$. $n = \overline{\overline{\operatorname{Coim}(e,1)}} \cdot 1 + \overline{\overline{\operatorname{Coim}(e,3)}} \cdot 3 + \overline{\overline{\operatorname{Coim}(e,5)}} \cdot 5 + ((\mathrm{id}_{\operatorname{dom}\mu} \cdot \mu), 7) + \ldots$. $n = \sum C$ by [16, (20)], (9). For every j such that $1 \leq j \leq \operatorname{len} d$ holds $O(j) = O_1(j)$ and $a(j) = a_1(j)$ by [21, (25)], [19, (9)], [2, (4)]. For every j, $\overline{\overline{\operatorname{Coim}(e, j \cdot 2 - 1)}} = 2^{a_1}(\tau_{j,1}) + ((2^{a_1} \odot \tau)(j), 2) + \ldots$ by [16, (42)], [21, (29)], [4, (72)], [16, (22)]. □

Fig. 3. The sketch generated automatically of proof [15, (11)] that justifies the existence of `Euler_transformation`. The content of the sketch has not been changed with the exception of the order of bibliography items.

At the beginning, we introduce a partition `d` of `n` into distinct parts and represent that its elements sum up to `n`:

```
let d be one-to-one a_partition of n;
n = d.1 + (d,2)+...+(d,len d) proof...
```
(19)

Then we introduce two finite sequences that describe a decomposition of every element of partition `p` as the product of a power of two and an odd number. We

formalize also the informal connection between these finite sequences and the number n that occur in the considered reasoning.

```
consider O be odd-valued FinSequence,a be natural-valued FinSequence such that
   len O = len d = len a & d = O (#) 2|^a and
   d.1 = O.1 * (2|^a.1) & ... & d.len d = O.len d * (2|^a.len d) by...
n = 2|^(a.1) * O.1 + 2|^(a.2) * O.2 + (O (#) 2|^a,3) +...+(O (#) 2|^a,len d)
   proof...
```
(20)

As it has been mentioned in Section 3.3 to select a method that groups together values of O we use the global choice: **the** odd_organization **of** O. Note that the same method is used to reorganize the values of 2|^a. It can be done since lengths of 2|^a and O are equal. However, we have to include this information in the type of this reorganization. In the Mizar system, such modification of the type can be realized as follows:

```
len (2|^a)=len O by...
then reconsider sort = the odd_organization of O as
   DoubleReorganization of dom (2|^a) by...
```
(21)

To formalize the unlabelled equality presented in Fig. 1 we have to introduce a finite sequence and three sets:

```
consider mu be (2*len sort)-element FinSequence of NAT such that
   for j holds mu.(2*j) = 0 &
               mu.(2*j-1) = (2|^a). sort_(j,1) + ((2|^a)*.sort.j,2)+... by...
set alpha = a*(sort.1), beta = a*(sort.2), gamma = a*(sort.3);
```
(22)

Then this equality can be formally formulated as follows:

```
n = ((2|^alpha).1+ (2|^alpha,2)+...) * 1 + ((2|^beta).1+ (2|^beta,2)+...) * 3 +
   ((2|^gamma).1+ (2|^gamma,2)+...) * 5 + ((id dom mu)(#)mu,7)+... proof...
n = mu.1 * 1 + mu.3 * 3 + mu.5 * 5 + ((id dom mu)(#)mu,7) +... proof...
```
(23)

Finally, to finish the informal contraction of value of p in Euler's transformation, we use the following 5 steps:

```
consider e be odd-valued FinSequence such that
   e is non-decreasing & for i holds card Coim(e,i) = mu.i by...
n = card Coim(e,1) * 1 + card Coim(e,3) * 3 +
    card Coim(e,5) * 5 + ((id dom mu)(#)mu,7) +... proof...
n = Sum e proof...
then reconsider e as odd-valued a_partition of n by...
take e;
```
(24)

However, these steps are not sufficient to finish a formal proof of the existence. For this purpose we need also to provide that the choice of e does not depend on a, O and sort. In the considered reasoning a part of this information is mentioned as *"d_i can be uniquely expressed as a power of 2 times an odd number"*. We recreate this information proving that for every pair of finite sequences a1, O1 that satisfies d = O1 (#) 2|^a1 holds **for** j **st** 1 <= j & j <= len d **holds**

`O.j = O1.j & a.j = a1.j`. Whereas the influence of `sort` selection for obtained value `e` is omitted. Therefore, to finish the proof, we provide this influence as an external auxiliary theorem [15, (5)] and only refering to this information from nesting reasoning that justifies the following proof step:

```
for j holds
   card Coim(e,j*2-1) = (2|^a1).sort1_(j,1) + ((2|^a1)*.sort1.j,2)+...
```
(25)

where `sort1` is an arbitrary `odd_organization` **of** `O1`.

Obviously, in the situation described above, where only several steps at the first level of nesting do not have their informal counterparts in the text book proof, it has a negative consequence, i.e. the size of the full proof. There are three main reasons for that. Firstly, note that to obtain such a proof we encapsulate less important fragments of reasoning at the deeper levels of nesting as subreasonings. Additionally, if steps in two different nesting subreasonings refer to the same auxiliary fact, then to avoid duplication, generally, we try to locate this fact in a common top level of nesting in such a way that this fact is available from these subreasonings (for more details see [17]). However, this solution is inconsistent with our goal, where we try to remove auxiliary facts form the first level of nesting. The second reason is the consequence of faithfully reproducing the informal term without any restrictions that are *a priori* assumed in the informal context. Note that to resolve the problem of *a priori* assumptions we have to provide several facts for different cases, that are completely redundant if we resign from the mirroring. Let us focus on the term $2^{\alpha_1} + 2^{\alpha_2} + \ldots$ that occurs in Fig. 1 and can be formalized simply as `sum (2|^alpha)`. But according to our purpose, we would like to obtain the term `2|^(alpha.1) + (2|^alpha,2)+...` that unfortunately is equal to `1` (= `2|^0`), if `alpha` is the empty finite sequence, where at the same time `sum (2|^alpha)=0`. Therefore, to resolve this problem we use `(2|^alpha).1` instead of `2|^(alpha.1)` in (23). The third reason is related to the previous one. Note that premises where we extract first few terms of summations are easily readable for a human, but difficult to use as premises. Often to use such premises we have to insert back these extracted terms and consider again the above-mentioned redundant cases.

This shows that the adaptation of the main idea from an informal proof to a formal one is not so trivial if we want to recreate it in a very precise way.

Since we defined `Euler_transformation`, we can prove that this transformation is a bijection. Note that the textbook proof contains only very sketchy justification of this property, that is formulated as follows: *In each series* $(2^{\alpha_1} + 2^{\alpha_2} + \ldots)$, *the* α_i*'s are distinct (why?).* We provide this fact in the following form [15, (13)]:

```
for O be odd-valued FinSequence, a be natural-valued FinSequence,
   s be odd_organization of O st len O = len a & O (#) 2|^a is one-to-one
holds a*.s.i is one-to-one
```
(26)

We remind that `s.i` is the finite sequence of all elements of the domain of `O`, for which the value under `O` is equals to `2*i-1`. Then `a*.s.i` is the image of the

elements of s.i under a. Based on this fact and the uniqueness of the binary number representation we provide in [15, (14)] that Euler_transformation is an injection:

```
for p1,p2 be one-to-one a_partition of n st
   Euler_transformation p1 = Euler_transformation p2
holds p1 = p2
```
(27)

Obviously, we provide also in [15, (15)] that Euler_transformation is a surjection, based on the existences of the binary number representation.

```
for e be odd-valued a_partition of n
   ex p be one-to-one a_partition of n st
      e = Euler_transformation p
```
(28)

Based on the two above-described statements we can "easily" prove that the set of all natural valued finite sequences that are partitions of n into odd parts has the same size as the set of all natural-valued finite sequences that are partitions of n into distinct parts. This statement can be represented as follows:

```
card {p where p is Element of NAT*:p is odd-valued a_partition of n}
  = card {p where p is Element of NAT*:p is one-to-one a_partition of n}
```
(29)

However, if we register in the Mizar environment, that there exists a set of all a_partition of n (for more details see [7]), we can represent (29) in more elegant form, used in [15, (16)], presented below:

```
card the set of all p where p is odd-valued a_partition of n
  = card the set of all p where p is one-to-one a_partition of n
```
(30)

We can compare the obtained formulation of Euler's partitions theorem in the Mizar system with the formulation used in the HOL system:

```
let EULER_PARTITION_THEOREM = prove
 (‘FINITE {p | p partitions n /\ !i. p(i) <= 1} /\
   FINITE {p | p partitions n /\ !i. ~(p(i) = 0) ==> ODD i} /\
   CARD {p | p partitions n /\ !i. p(i) <= 1} =
   CARD {p | p partitions n /\ !i. ~(p(i) = 0) ==> ODD i}‘
```

5 Conclusion

In this paper we presented a formalization of a textbook theorem where we not only proved this theorem, but primarily tried to reflect the main idea of the informal proof with expressions that are available in a formal environment. We have created more expressive definitions and we extended existing ones to mirror informal mathematical language constructions in formal terms, working around the Mizar system's limitations, without modification of the core Mizar system. We have showed that an accurate formal reflection of informal terms obliges not only to introduce new concepts in our library, but also to conduct reasoning in a more difficult way. Our studies highlighted the differences between the way of

conducting human legible reasoning and reasoning that is acceptable for a proof checker.

We have showed that appropriate use of flexary operators in formal reasonings can increase the legibility of obtained proof scripts, in effectively the same way as in mathematical textbook. However, based on the same example we have also showed a negative consequence of this method, namely the growth of the reasoning length. Obviously, going beyond the point of this study and modifying the core of the Mizar system, we can obtain a more natural definition of flexary operators, where we do not have to explicitly use the summed sequence.

Our effort allowed us to formulate similarly formal equivalents of the great majority of informal terms. Still a number of corner cases resisted our efforts. In particular handling sequences of length zero was problematic and we had to fall back to non-uniform treatment of the zero-length case for the sequence $2^{\alpha_1} + 2^{\alpha_2} + \ldots$

We believe that this study brings us closer to the situation that informal reasonings can be conducted in systems such as Mizar.

References

1. G.E. Andrews. *Number Theory*. W. B. Saunders Company, Philadelphia, Dover edition, 1971.
2. G. Bancerek. Countable Sets and Hessenberg's Theorem. *Formalized Mathematics*, 2(1):65–69, 1991.
3. G. Bancerek and P. Rudnicki. Information Retrieval in MML. In A. Asperti (eds.), *Proc. of Mathematical Knowledge Management 2003*, volume 2594, page 119–131. Springer, Heidelberg, 2003.
4. C. Byliński. Functions and Their Basic Properties. *Formalized Mathematics*, 1(1):55–65, 1990.
5. R. Engelking. *General Topology*. PWN - Polish Scientific Publishers, Warsaw, 1977.
6. L. Euler. *Introduction to the Analysis of the Infinite Book I Translated by John D. Blanton*. Springer-Verlag, 1988.
7. A. Grabowski, A. Korniłowicz, and A. Naumowicz. Mizar in a Nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
8. A. Grabowski and C. Schwarzweller. On Duplication in Mathematical Repositories. In D. Hutchison (eds.), *Intelligent Computer Mathematics, Lecture Notes in Computer Science, vol. 6167*, pages 300–314. Springer-Verlag, 2010.
9. A. Grabowski and Ch. Schwarzweller. Improving Representation of Knowledge within the Mizar Library. *Studies in Logic, Grammar and Rhetoric*, 18(31):35–50, 2009.
10. J. Harrison. A HOL Theory of Euclidean Space. In J. Hurd (eds.), *8th International Conference on Theorem Proving and Higher-Order Logic, Lecture Notes in Computer Science, vol. 3603*, pages 114–129. Springer-Verlag, 2005.
11. A. Korniłowicz. Tentative Experiments with Ellipsis in Mizar. In J. Jeuring (eds.), *Intelligent Computer Mathematics 11th International Conference*, volume 7362 of *Lecture Notes in Artificial Intelligence*, pages 453–457. Springer-Verlag, 2012.

12. F. Horozal, F. Rabe, and M. Kohlhase. Flexary Operators for Formalized Mathematics. In S.M. Watt, J.H. Davenport, A.P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics - International Conference, Lecture Notes in Computer Science, vol. 8543*, pages 312–327. Springer-Verlag, 2014.
13. A. Korniłowicz and Y. Shidama. Brouwer Fixed Point Theorem for Disks on the Plane. *Formalized Mathematics*, 13(2):333–336, 2005.
14. A.C. Leisenring. *Mathematical Logic and Hilbert's ε-Symbol*. Gordon and Breach, New York, 1969.
15. K. Pąk. Euler's Partition Theorem. *Formalized Mathematics*, 23(2):91–98, 2015, doi: 10.2478/forma-2015-0009.
16. K. Pąk. Flexary operations. *Formalized Mathematics*, 23(2):79–90, 2015, doi: 10.2478/forma-2015-0008.
17. K. Pąk. Methods of Lemma Extraction in Natural Deduction Proofs. *Journal of Automated Reasoning*, 50(2):217–228, 2013.
18. K. Pąk. Topological Manifolds. *Formalized Mathematics*, 22(2):179–186, 2014.
19. P. Rudnicki and Andrzej A. Trybulec. Abian's Fixed Point Theorem. *Formalized Mathematics*, 6(3):335–338, 1997.
20. J.J. Sylvester and F. Franklin. A Constructive Theory of Partitions, Arranged in Three Acts, an Interact and an Exodion. *Amer. J. Math.*, 5:251–330, 1882.
21. W.A. Trybulec. Non-contiguous Substrings and One-to-one Finite Sequences. *Formalized Mathematics*, 1(3):569–573, 1990.
22. H.S. Wilf. Lectures on Integer Partitions. 2000.