Automated improving of proof legibility in the Mizar system[⋆]

Karol Pąk

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

Abstract. Both easily readable and obscure proof scripts can be found in the bodies of formalisations around formal proof checking environments such as Mizar. The communities that use this system try to encourage writing legible texts by making available various solutions, e.g., by introduction of phrases and constructs that make formal deductions look closer to the informal ones. Still, many authors do not want to invest additional efforts in enhancing readability of their scripts and assume this can be handled automatically for them. Therefore, it is desirable to create a tool that can automatically improve legibility of proofs. It turns out that this goal is non-trivial since improving features of text that enhance legibility is in general NP-complete.
The successful application of SMT technology to solving computationally difficult problems suggests that available SMT solvers can give progress in legibility enhancement. In this paper we present the first experimental results obtained with automated legibility improving tools for the Mizar system that use Z3 solver in the backend.

Key words: Operations on languages, Legibility of proofs, Proof assistants, SMT solvers

1 Introduction

1.1 Motivations

Analysing examples of declarative natural deduction proof scripts, especially long and complicated ones, it can be observed that the proofs are often formulated in a chaotic way. During the analysis of formalised proofs, we can obviously find that some authors of proof scripts spend a lot of time on their readability, but there are other ones who tend to create deductions that are correct for computers, neglecting the legibility of their proof scripts. They believe that no one, with the exception of a proof checker, will want to analyse them. The experience of big proof development efforts [8] shows that adapting or modifying the existing proofs is unavoidable and requires reading of proof scripts [10]. Additionally, any

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

attempt to analyse the details of the proof scripts created in this way, according to the opinion of some proof writers, is extremely difficult or even impossible.

The problem of formal deductions illegibility, which at first sight does not seem to be difficult, led to the abandonment or collapse of many formalisation projects such as these carried by Bourbaki or Whitehead and Russell [28]. The formalisation has become feasible only after the availability of computers [29] increased, but even now it is still quite difficult. Therefore, it comes as no surprise that communities around formal proof checking environments such as Mizar [18] and Isabelle/Isar [27] attempt to resolve this problem and often implement various solutions aimed at raising the legibility of proof scripts. Adaptation of informal mathematical language constructs to formal ones is the most natural direction for research. Many of these implemented solutions [14, 26] are also known from the programming language editor frameworks, where they proven their usefulness (B. A. Myers shows that syntax highlighting or hint system in programming languages can save up to 35% of time spent for search in the code that is supposed to be modified [13]). These efforts are also focused on the visualisation of proof scripts in the linked HTML form [25].

1.2 Proposed approach

In this paper we focus on another, still underdeveloped approach that concentrates on modification of order between independent steps written in a proof script. Based on a result of a long experience with the Mizar system and development of Mizar Mathematical Library (MML) [19], an analysis of different opinions shared by users of MML as well as models of human cognitive perception of read material, we can conclude that there is a close relationship between the reasoning legibility and grouping of steps into linear, directly connected fragments. Naturally, legibility has different meanings for different people, but in general the results obtained in this research complies with the expectations. This research precisely demonstrates how important for human being "local deductions" are in the analysing process of mathematical proofs.

This can be summarised with the Behaghel's First Law that *elements that belong close together intellectually should be placed close together* [1]. This law is also recognised in modern scientific literature concerning human perception [15]. Note that every information used in justification of a step had to be derived before in the proof, but often we can manipulate the location of these premises in deduction. With Behaghel's law in mind, we assume that a step where at least part of required information is available in close neighborhood of the step is more intelligible than a step in which all used information is far away in the proof scripts. Obviously, close neighborhood can be defined in different ways, therefore we parametrise this notion with a fixed number n that counts the number of preceding steps that are considered to be our close neighborhood.

An important case emerges for $n = 1$. In this case the general method of referencing by means of labels can be replaced in the Mizar system by the `then` construction. Additionally, in case where every reference to a step can be replaced by the `then` construction, the label attached to this step in unnecessary and can be removed from proof scripts. In consequence, the number of labels in the proof script can be brought smaller. Note that the human short-term memory is the capacity for holding a small amount of information in mind in an active state, and its capacity is 7 ± 2 elements [3, 17]. However, this capacity is prone to training [5], and in the case of research on Mizar it has been shown that it is in the range 5-10 [16]. Therefore, it comes as no surprise that we want to choose an ordering of steps that maximises the number of references in close neighbourhoods of reference targets, where we understand by a close neighbourhood proof steps that are supposed to be directly available from within the short-term memory.

In this paper we present the first experimental results obtained with automated legibility improving tools for Mizar that aim at realisation of the above-described legibility criteria. Since improving legibility in most cases in NP-hard [22], we also present the results obtained with the application of a SMT-solver Z3 [4]. The impact of these criteria and the other ones has been already recognised by the scientific community of people who write proof scripts in Mizar [20, 21] and in other systems [2, 12, 24].

In Section 2 we introduce the notion of an abstract model of proofs and its linearisation. In Section 3 we discuss selected methods to improve legibility and we show the complexity of two so far open cases, transforming to them known NP-complete problems. In Section 4 we discuss translations of legibility problems to the Z3-solver and application of its responses to proof scripts. Then in Section 5 we report the statistical results obtained for the MML database. Finally, Section 6 concludes the paper and discusses the future work.

2 Graph representation of proofs

Mizar [19] is a mathematical language and a proof checker for the language that are based on natural deduction created by S. Jaśkowski and F. B. Fitch [6, 11]. In this paper, we do not concentrate on a full explanation of Mizar. We will just focus on a few easy-to-observe Mizar proof structure features presented in an example (Fig. 1), where the statement *a square matrix is invertible if and only if it has non-zero determinant* is proved. This example is contained in an article by Pąk and Trybulec [23].

Note that symbols `0.K`, `1.K` represent the additive identity element and the multiplicative identity element of a field `K`, respectively. Additionally, the Mizar system uses only ASCII characters, therefore operations such as $^{-1}$, $\cdot$, $(\cdot)^T$ are represented in Mizar as `"`, `*`, `@` respectively, the labels `MATRIX_6`, `MATRIX_7`, `MATRIX11` are identifiers of Mizar articles. An abstract model of the proof scripts (see Fig. 2) was considered in detail in [20], but for our purposes we detail only a sketch of its construction, focusing mainly on one-level deductions that ignore nested lemmas. Generally we call a DAG $P = \langle V, O \cup M \rangle$ *abstract proof graph* if O, M are disjoint families of arcs, called *ordered arcs* and *meta-edges* respectively, and O contains a distinguished set of arcs $\mathfrak{R}(P) \subseteq O$, the elements of which are called *references*. The vertices of P represent reasoning steps, ordered arcs represent all kinds of additional constraints that force one step to precede another

```
 1:    theorem Th34:
         for n be Nat, M be Matrix of n,K st n >= 1 holds
           M is invertible iff Det M <> $0_K$
         proof
 2:        let n be Nat, M be Matrix of n,K such that
     A1:     n >= 1;
 3:        thus M is invertible implies Det M <> $0_K$
           proof
 4:          assume M is invertible;
 5:          then consider Minv be Matrix of n,K such that
     A2:       M is_reverse_of Minv by MATRIX_6:def 3;
 6:  A3:     M · Minv = 1.(K,n) by A2,MATRIX_6:def 2;
 7:          Det 1.(K,n) = $1_K$ by A1,MATRIX_7:16;
 8:          then Det M · Det Minv = $1_K$ by A1,A3,MATRIX11:62;
 9:          thus then Det M <> $0_K$;
           end;
10:        assume
     A4:     Det M <> $0_K$;
11:        then
     A5:     M · ( (Det M)$^{-1}$ · (Matrix_of_Cofactor M)$^T$ ) = 1.(K,n) by Th30;
12:        (Det M)$^{-1}$ · (Matrix_of_Cofactor M)$^T$ · M = 1.(K,n) by A4,Th33;
13:        then M is_reverse_of (Det M)$^{-1}$ · (Matrix_of_Cofactor M)
             by A5,MATRIX_6:def 2;
14:        thus then thesis by MATRIX_6:def 3;
         end;
```

Fig. 1. The example of proof script written in the Mizar style.

one, and meta-edges represent dependence between a step that as a justification contains nested reasoning and each step of this reasoning. In Fig. 2 arrows $\twoheadrightarrow$ represent meta-edges. Elements of $\mathfrak{R}(P) \subseteq O$ correspond to solid arrows and represent the information flow between a step (the head of the arc, e.g., the vertex 8) that use in the justification premises formulated in a previously justified step (the tail of the arc, e.g., vertices 2, 6, 7). Other ordered arcs correspond to dashed arrows and represent, e.g., the dependence between a step that introduces a variable into the reasoning and a step that uses this variable in an expression (e.g., the arc $\langle 5, 6\rangle$ that corresponds to the use of the variable `Minv`). Ordered arcs represent also the order of special kind reasoning steps in the Jaśkowski-style natural-deduction proofs (for more detail see [9]) such as steps that introduce quantifier or implication; or indicate a conclusion (e.g., arcs $\langle 2, 3\rangle$, $\langle 4, 9\rangle$). Clearly, every abstract proof graph does not contain labeled vertices and multiple arcs. Labels and multiple arcs visible in Fig. 2 have been used here only to aid the reader and simplify their identification.

It is easily seen that digraph $\langle V, M\rangle$ is a forest, i.e., a disjointed union of trees, in which every connected maximal tree is an arborescence (i.e., a rooted tree where all arcs are directed from leaves to the root). Additionally, using the notion of meta-edges we can define formal equivalent of one-level deductions.

Definition 1. *Let $P = \langle V, O \cup M\rangle$ be an abstract proof graph and D be a subgraph of P induced by a set of vertices. We call D* one-level deduction*, if D is induced by the set of all roots in the forest $\langle V, M\rangle$ or it is induced by $N^{-}_{\langle V,M\rangle}(v)$ for some vertex $v \in V$.*

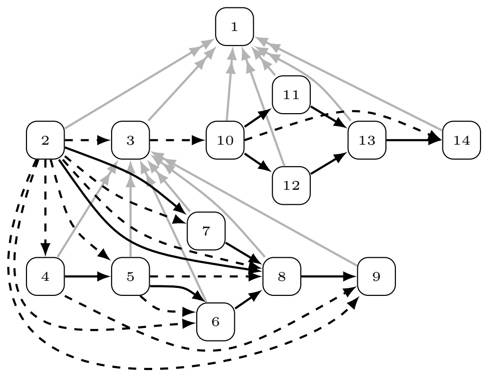

Fig. 2. The abstract proof graph illustrating the structure of reasoning presented in Fig. 1.

Naturally, one-level deductions do not have meta-edge. Consequently, focusing only on such deductions results in a simplified model of proof graphs. Negative consequence of this simplification is that hidden dependencies between steps in one-level deductions may occur. Therefore, we have to carefully add such dependencies to these digraphs. These dependencies was considered in detail in [22]. Here we remind only that hidden dependency occur between vertices v, u if

(i) there exists a common one-level deduction that contains v and u,
(ii) the step that corresponds to u is justified by a nesting lemma,
(iii) a step s of this nesting lemma uses in the expression a variable that is introduced in a step $\overline{v}$ or the justification of s refers to the statement formulated in $\overline{v}$, where $\overline{v}$ corresponds to v.

Additionally in the case, where the justification of s refers to the statement formulated in $\overline{v}$ we call $\langle u, v\rangle$ *extended reference*.

Let $D = \langle V_D, O_D\rangle$ be a one-level deduction of P. To simplify we assume that the set of ordered arcs of D contains also every hidden dependency between vertices of V_D, and $\mathfrak{R}(D)$ contains only original references of P that connect vertices of V_D. The set of reference and extended reference arcs of G is denoted by $\overline{\mathfrak{R}}(D)$.

To study the general case, without the Mizar context, we assume only relation between distinguished sets of arcs in D that $\mathfrak{R}(D) \subseteq \overline{\mathfrak{R}}(D) \subseteq O_D$. Therefore, in the following considerations as a one-level deduction we take a DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}\rangle$, with two distinguished sets $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$ that correspond to $\mathfrak{R}(\mathcal{D})$ and $\overline{\mathfrak{R}}(\mathcal{D})$, respectively. For simplicity, we assume also that $\mathcal{A}_1$-references can be replaced in the Mizar system by the `then` construction.

We identify here a modification of independent steps order to improve proof legibility with a modification of a topological sorting of $\mathcal{D}$. By *topological sorting*, called also *linearisation*, we mean a one-to-one function $\sigma : \mathcal{V} \to \{1, 2, \ldots |\mathcal{V}|\}$ such that $\sigma(u) < \sigma(v)$ for each arc $\langle u, v\rangle \in \mathcal{A}$. We denote by $TS(\mathcal{D})$ the set of all topological sortings. Let us consider $\sigma \in TS(\mathcal{D})$. We call a vertex v of $\mathcal{V}$ a $\mathtt{then}^{\mathcal{A}_1}(\sigma)$*–step* if v corresponds to a vertex that is linked by $\mathcal{A}_1$ to the directly preceding step in the linearisation σ (e.g., steps 5, 8, 9 in Fig. 1) or more precisely:

$$v \in \mathtt{then}^{\mathcal{A}_1}(\sigma) \iff \sigma(v) \neq 1 \wedge \langle \sigma^{-1}(\sigma(v)-1), v\rangle \in \mathcal{A}_1. \tag{1}$$

We call a directed path $p = \langle p_0, p_1, \ldots p_n\rangle$ of $\mathcal{D}$ a $\sigma^{\mathcal{A}_1}$*–linear reasoning* if p_k is a $\mathtt{then}^{\mathcal{A}_1}(\sigma)$–step (i.e., $\sigma(p_k) = 1 + \sigma(p_{k-1})$ and $\langle p_{k-1}, p_k\rangle \in \mathcal{A}_1$) for each $k = 1, 2, \ldots, n$ (p_0 does not have to be a $\mathtt{then}^{\mathcal{A}_1}(\sigma)$–step). A $\sigma^{\mathcal{A}_1}$–linear reasoning P is *maximal* if it is not a subsequence of any other $\sigma^{\mathcal{A}_1}$–linear reasoning. In our considerations we also use a function that maps vertices of consecutive maximal $\sigma^{\mathcal{A}_1}$–linear reasoning in linearisation σ to consecutive natural numbers. Let $\mathfrak{Then} : TS(\mathcal{D}) \times 2^{\mathcal{A}} \times \mathcal{V} \to \{1, 2, \ldots, |\mathcal{V}|\}$ be given by the following recursive definition:

$$\mathfrak{Then}_\sigma^{\mathcal{A}_1}(v) = \begin{cases} 1 & \text{if } \sigma(v) = 1, \\ \mathfrak{Then}_\sigma^{\mathcal{A}_1}(\sigma^{-1}(\sigma(v)-1)) & \text{if } v \in \mathtt{then}^{\mathcal{A}_1}(\sigma), \\ \mathfrak{Then}_\sigma^{\mathcal{A}_1}(\sigma^{-1}(\sigma(v)-1)) + 1 & \text{if } v \notin \mathtt{then}^{\mathcal{A}_1}(\sigma), \end{cases} \tag{2}$$

for an arbitrary $v \in \mathcal{V}$, $\mathcal{A}_1 \subseteq \mathcal{A}$, and $\sigma \in TS(\mathcal{D})$.

Now we define a formal equivalent of a label in proof graph formalism. We will say that a vertex v of $\mathcal{V}$ has to have a label or is a *labeled vertex* in linearisation σ if v is the tail of at most one $\mathcal{A}_2 \setminus \mathcal{A}_1$-arc (e.g., steps 2 in Fig. 1) or is the tail of at most one $\mathcal{A}_1$-arc that corresponds to a link which does not connect two steps located directly one after the other in the linearisation σ (e.g., step 6 in Fig. 1). We write $\mathtt{lab}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$ for the set of all labeled vertices given by

$$v \in \mathtt{lab}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma) \iff \underset{u\in\mathcal{V}}{\exists} \big(\langle v, u\rangle \in \mathcal{A}_2 \setminus \mathcal{A}_1 \vee \ (\ \langle v, u\rangle \in \mathcal{A}_1 \wedge \sigma(v) + 1 \neq \sigma(u)\ )\big) \tag{3}$$

where $v \in \mathcal{V}$. We define also a function $\mathfrak{Lab} : TS(\mathcal{D}) \times 2^{\mathcal{A}} \times 2^{\mathcal{A}} \times \mathcal{V} \to \{1, 2, \ldots, |\mathcal{V}|\}$ that associates the number of all labeled vertices u such that $\sigma(u) < \sigma(v)$, with every vertex $v \in \mathcal{V}$, defined as:

$$\mathfrak{Lab}_\sigma^{\mathcal{A}_1,\mathcal{A}_2}(v) = \begin{cases} 0 & \text{if } \sigma(v) = 1, \\ \mathfrak{Lab}_\sigma^{\mathcal{A}_1,\mathcal{A}_2}(\sigma^{-1}(\sigma(v)-1)) & \text{if } \sigma(v) \neq 1 \wedge v \notin \mathtt{lab}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma), \\ \mathfrak{Lab}_\sigma^{\mathcal{A}_1,\mathcal{A}_2}(\sigma^{-1}(\sigma(v)-1)) + 1 & \text{if } \sigma(v) \neq 1 \wedge v \in \mathtt{lab}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma). \end{cases} \tag{4}$$

At last we define a metric $\mathrm{d}_\sigma : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{N}$ for a linearisation σ, which is called σ–distance and is determined by $\mathrm{d}_\sigma(v, u) = |\sigma(v) - \sigma(u)|$ for each $v, u \in \mathcal{V}$.

3 Methods of Improving Legibility

According to the approach presented in Section 1.2 we can formally define criteria of reasoning linearisation that quantify the legibility of obtained proof scripts.

Since steps that refer to the preceding reasoning statements are perceived as more intelligible, we expect that the set $\mathtt{then}^{\mathcal{A}_1}(\sigma)$ has the largest cardinality for the selected linearisation σ (1st MIL). Obviously, the length of σ–linear reasoning is also important for proof readers, therefore the average length of σ–linear reasoning should also have the maximal value. Note that optimisation of this determinant is realised by 1st MIL defined below, since such average length is equal to $\frac{|V_G|}{|V_G|-|\mathtt{then}^{\mathcal{A}_1}(\sigma)|}$.

The 1st Method of Improving Legibility (1st MIL):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, a subset $\mathcal{A}_1$ of $\mathcal{A}$, a positive integer $K \leq |\mathcal{V}|$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which $|\mathtt{then}^{\mathcal{A}_1}(\sigma)| \geq K$?

In addition we consider a parameterised version of the above-mentioned problem, since a close neighborhood of a step in proof scripts can be extended from the directly preceding step to last n preceding steps.

The 1st Method of Improving Legibility for n (1st MIL$_n$):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, a subset $\mathcal{A}_2$ of $\mathcal{A}$, a positive integer $K \leq |\mathcal{A}_2|$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which

$$\mathtt{then}^{\mathcal{A}_2}_{\leq n}(\sigma) := \{\langle v, u \rangle \in \mathcal{A}_2 : \mathrm{d}_\sigma(v, u) \leq n\}$$

has size at most K?

It is also desirable that fragments of reasoning that are maximal σ–linear subsequences should be pieces of reasoning with dense information flow.

Therefore, this flow has to be maximal in legible proof scripts or equivalently, the information flow between maximal σ–linear reasonings has to be minimal. This can be formulated as follows:

The 2st Method of Improving Legibility (2st MIL):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, subsets $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$, a positive integer $K \leq |\mathcal{A}_2|$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which

$$\{\langle v, u \rangle \in \mathcal{A}_2 : \mathfrak{Then}^{\mathcal{A}_1}_\sigma(v) \neq \mathfrak{Then}^{\mathcal{A}_1}_\sigma(u)\}$$

has size at most K?

In a similar way we obtain that the number of labeled vertices in the selected linearisation σ should be the smallest (3rd MIL), the same as sum of all σ–distances between vertices linked by $\mathcal{A}_2$-arcs (4th MIL).

The 3rd Method of Improving Legibility (3nd MIL):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, subsets $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$, a positive integer $K \leq |V|$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which $|\mathtt{lab}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma)| \leq K$?

The 4th Method of Improving Legibility (4th MIL):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, a subset $\mathcal{A}_2$ of $\mathcal{A}$, a positive integer $K \leq \binom{|\mathcal{V}|+1}{3}$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which

$$\sum_{\langle u,v \rangle \in \mathcal{A}_2} \mathrm{d}_\sigma(v, u) \leq K?$$

Referring to the short-term memory limitation of humans we formulated the last method that maximises the number of more intelligible references. We rely here on the assumption that a reference to a labeled vertex is more intelligible, if the number of statements that correspond to labeled vertices between linked by this reference vertices can be remembered by a reader.

The 5th Method of Improving Legibility for n (5th MIL$_n$):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, subsets $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$, a positive integer $K \leq |\mathcal{A}_2|$.
QUESTION: Does there exist a topological sorting σ of $\mathcal{D}$ for which

$$\mathfrak{R}^{\mathcal{A}_1,\mathcal{A}_2}(\sigma) := \{\langle v, u \rangle \in \mathcal{A}_2 : \mathfrak{Lab}_\sigma^{\mathcal{A}_1,\mathcal{A}_2}(u) - \mathfrak{Lab}_\sigma^{\mathcal{A}_1,\mathcal{A}_2}(v) \leq n\}$$

has size at last K?

It is easy to see that the 4th MIL is NP-complete, since it generalises a known NP-complete problem Directed Optimal Linear Arrangement (see GT43 in [7] for $\mathcal{A}_2 = \mathcal{A}$). NP-completeness has been shown also for problems 1st, 2nd MIL, even for restricted instances $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{A}$ [22]. Additionally, for every instance of these problems in this case, there exists a Mizar proof script that contains a one-level deduction whose structure is that instance [22]. Therefore, realisation of 1st, 2nd, and 4th methods for one-level deductions potentially occurring in MML is NP-hard. In consequence, realisations of such methods for abstract proof graphs are also NP-hard, as one-level deductions are its substructures.

The problem 3rd MIL is also NP-complete in the general case, but for instances limited to ones that can actually occur in MML it is solvable in polynomial time [22]. Consequently, it is possible to effectively minimise the number of labels in Mizar proof scripts. These limits are a consequence of an additional syntax restriction of Mizar to use the `then` construct.

We show in Theorems 1, 2 below that problems 1st MIL$_n$ and 5th MIL$_n$ are also NP-complete. However, we do not focus on details of these proofs and we present only sketches.

Theorem 1. *The* 1st MIL *problem is reducible to the* 1st MIL$_n$ *problem for each natural number n.*

Proof. It is easily seen that the 1st MIL and the 1st MIL$_1$ are equivalent, if we take $\mathcal{A}_1 = \mathcal{A}_2$. Therefore, we can clearly assume that $n > 1$. Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, $\mathcal{A}_1 \subseteq \mathcal{A}$, $K \leq |\mathcal{V}|$ be an instance I of the 1st MIL. We can clearly construct in logspace a digraph $\mathcal{D}' = \langle \mathcal{V} \cup \mathcal{V}', \mathcal{A} \cup \mathcal{A}' \rangle$, a subset $\mathcal{A}_2' = \mathcal{A}_1 \cup \mathcal{A}'$, and $K' = K + (n-1) \cdot |\mathcal{V}|$ as follows:

$$\mathcal{V}' := \{v_i : v \in \mathcal{V} \wedge 1 \leq i < n\}, \qquad \mathcal{A}' := \{\langle v_i, v \rangle : v \in \mathcal{V} \wedge 1 \leq i < n\} \tag{5}$$

and consider it to be an instance I' of the 1st MIL$_n$. Let us take $\sigma \in TS(\mathcal{D})$ that is a solution of the 1st MIL problem for I and define a topological sorting $\sigma' \in TS(D')$ as $\sigma'(v) := n \cdot \sigma(v)$, $\sigma'(v_i) := i + (n-1) \cdot \sigma(v)$ for each $v \in \mathcal{V}$ and $1 \leq i < n$. Obviously, $\mathcal{A}' \subseteq \mathtt{then}_{\leq n}^{\mathcal{A}_2'}(\sigma')$ since a segment that contains all vertices of the form v_i for $i = 1, 2, \dots, n-1$ directly precedes v for each $v \in \mathcal{V}$. Additionally, $d_\sigma(v, u) = 1$ if and only if $d_{\sigma'}(v, u) = n$, hence finally $|\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')| \leq K'$ and σ' is a solution of I'.

Now let $\sigma' \in TS(\mathcal{D}')$ be a solution of the 1st MIL_n problem for I'. Note that the maximal value of $\sigma'(\mathcal{V}')$ is obtained for a vertex of $\mathcal{V}$. Denote it is as v. Let i be the index for which $\sigma'(v_i)$ has the smallest value among each $i = 1, 2, \dots, n-1$.

We show that we can move all vertices of $\mathcal{V}'_v := \mathcal{V}' \setminus \{v, v_1, v_2, \dots, v_{n-1}\}$ that are located between v_k and v, before v_k in σ' for each $k = 1, 2, \dots, n-1$ so that the arrangement between vertices of $\mathcal{V}'_v$ is preserved and the size of $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_1)$ is not reduced, where σ'_1 is the new linearisation obtained in this way. Note that to compare size of $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')$ and $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_1)$ we can clearly compare only $\mathcal{A}'_2$-arcs in-going to v included therein, which can be at most n in both sets. Suppose, contrary to our claim, that $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')$ has more such arcs. But $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_1)$ contains $\langle v_j, v\rangle$ for each $j = 1, 2, \dots, n-1$, hence $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')$ contains exactly n such arcs.

Thus, $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')$ contains the arcs $\langle u, v\rangle$ that have to be $\mathcal{A}'_2$-arcs, where u is the last vertex of $\mathcal{V}$ before v in σ. But then $d_{\sigma'_1}(u, v) = n$ and in consequence $\langle u, v\rangle \in \mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_1)$. This contradicts our assumption that $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma')$ has more arcs in-going to v than $\mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_1)$.

In a similar way, we can arrange vertices before the second, the third, the fourth, $\dots, |\mathcal{V}|$-th up to the last vertex of $\mathcal{V}$ in $\sigma'_1, \sigma'_2, \dots, \sigma'_{|\mathcal{V}|-1}$, creating a sequence $\sigma'_2, \sigma'_3, \dots, \sigma'_{|\mathcal{V}|}$, respectively. Additionally, a topological sort $\sigma_{|\mathcal{V}|} \in TS(\mathcal{D})$ that preserves the order of vertices of $\mathcal{V}$ in $\sigma'_{|\mathcal{V}|}$ is a solution of I, since

$$K' \leq \mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma') \leq \mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma'_{|\mathcal{V}|}) = \mathtt{then}^{\mathcal{A}_1}(\sigma_{|\mathcal{V}|}) + (n-1) \cdot |\mathcal{V}|, \tag{6}$$

and the proof is completed.

Theorem 2. *The* 1st MIL_n *problem is reducible to the* 5th MIL_n *problem.*

Proof. Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}\rangle$, $\mathcal{A}_2 \subseteq \mathcal{A}$, $K \leq |\mathcal{A}_2|$ be an instance I of the 1st MIL. Let $\mathcal{D}' = \langle \mathcal{V} \cup \mathcal{V}', \mathcal{A} \cup \mathcal{A}' \cup \mathcal{A}''\rangle$, $\mathcal{A}'_1 = \emptyset$, $\mathcal{A}'_2 = \mathcal{A}_2 \cup \mathcal{A}'$, $K' = K$ defined as follows:

$$\begin{gathered} \mathcal{V}' := \{v_1, v_2, \dots, v_{n+1}\}, \qquad \mathcal{A}' := \{\langle u, v_{n+1}\rangle : u \in \mathcal{V}\}, \\ \mathcal{A}'' := \{\langle u, v_1\rangle : u \in \mathcal{V}\} \cup \{\langle v_i, v_{i+1}\rangle : 1 \leq i \leq n\}, \end{gathered} \tag{7}$$

be an instance I' of the 5th MIL_n. Let us consider $\sigma \in TS(\mathcal{D}')$. Note that $\mathcal{A}' \cap \mathfrak{R}^{\mathcal{A}'_1, \mathcal{A}'_2}(\sigma) = \emptyset$. Indeed, for every $\mathcal{A}'$-arcs $\langle u, v_{n+1}\rangle$ we have $\mathrm{d}_\sigma(u, v_n{+}1) > n$, since $\sigma(u) < \sigma(v_1) < \sigma(v_2) < \ldots < \sigma(v_{n+1})$. The main task of $\mathcal{A}'$–arcs is labeling every vertex of $\mathcal{V}$. Consequently, we obtain that $\mathfrak{Lab}^{\mathcal{A}'_1, \mathcal{A}'_2}_\sigma(w_1) - \mathfrak{Lab}^{\mathcal{A}'_1, \mathcal{A}'_2}_\sigma(w_2) = \sigma(w_1) - \sigma(w_2)$ for each $w_1, w_2 \in \mathcal{V}$. Note also that every vertex of $\mathcal{V}'$ has to be located after all vertices of $\mathcal{V}$ in σ and in unique order ($\sigma(v_i) = |\mathcal{V}| + i$ for each $i = 1, 2, \dots, n+1$). Hence $\mathfrak{R}^{\mathcal{A}'_1, \mathcal{A}'_2}(\sigma) = \mathtt{then}^{\mathcal{A}'_2}_{\leq n}(\sigma) = \mathtt{then}^{\mathcal{A}_2}_{\leq n}(\sigma_{|\mathcal{V}})$, where $\sigma_{|\mathcal{V}}$ is a topological sorting of $\mathcal{D}$ obtained by restricting σ to $\mathcal{V}$. Now the proof is immediate.

4 Automated improving of legibility as support for Mizar proof authors

When we restrict our focus to methods based on a modification of the order of independent steps, the presented techniques agree in most cases with the needs of Mizar users. However, it is controversial what the particular hierarchy of criteria should be applied, i.e., which criteria should be considered to be more important. Additionally, it can be observed in proof scripts of MML that application of a method to improve legibility can degrade the parameter optimised by another method. Therefore, we created a flexible application which can be used even by users with conflicting hierarchies of criteria for legibility. To use this application on a proof script, author needs simply to indicate a one-level deduction by typing a pragma `::$IL` $\langle strategy \rangle\ \{\langle method \rangle\}^*$ at the beginning of this deduction (e.g., directly after `proof`, see Fig. 2) and run it, where:

$$\begin{aligned}
\langle strategy \rangle &::= \texttt{Z3} : \langle time \rangle \mid \texttt{BF} : \langle number \rangle \mid \texttt{auto} : \langle number \rangle : \langle time \rangle\ ,\\
\langle method \rangle &::= \langle criterion \rangle : \langle condition \rangle\ [\, : \langle parameter \rangle\,]\ ,\\
\langle criterion \rangle &::= \texttt{Then} \mid \texttt{Flow} \mid \texttt{Lab} \mid \texttt{SumRef} \mid \texttt{LabRef}\ ,\\
\langle condition \rangle &::= \texttt{=} \mid \texttt{<} \mid \langle number \rangle,\\
\langle parameter \rangle &::= \langle number \rangle\ .
\end{aligned}$$

This application tries to find in three stages a more legible linearisation of a deduction decorated with this pragma. First, we create an abstract proof graph of this deduction. In the second stage, depending on the specified strategy, we describe the proof graph and the selected methods as a list of assertions and we check satisfiability by the Z3 solver for a given number of seconds ($\texttt{Z3} : \langle time \rangle$); use a brute-force attack with the limited number of checked linearisations ($\texttt{BF} : \langle number \rangle$); or check the number of linearisations and depending on the result we select the first (for greater than the limit) or the second (otherwise) strategy ($\texttt{auto} : \langle number \rangle : \langle time \rangle$). In the third stage, we adapt the obtained solution to the considered proof script, if we get a more legible linearisation. Criteria correspond to the 1st MIL$_n$, 2nd, 3rd, 4th MILs, 5th MIL$_n$ problems respectively. Moreover, `Then` criterion with parameter `0` corresponds to the 1st MIL problem. The condition field set to `<` instructs the tool to search for a linearisation in which the optimised parameter given as the $\langle criterion \rangle$ tag is improved when compared with the initial situation. The condition field set to `=` instructs the tool not to make the parameter worse. Finally, condition field set to $\langle number \rangle$ defines the degree of the criterion in selected hierarchy.

The choice of the brute-force method is a consequence of the fact that 96,2% of one-level deductions in MML have at most one million possible linearisations that can be checked in this way "effectively". A vast majority of such deductions is located on deeply nested levels of proofs. Note that the other 3,8% of deductions cannot be omitted in the process of improving legibility, since this small group in MML is mainly located on shallow levels of proofs theorems (46% of this group is located on the first level, 25% on the second one, 14% on the third one). Additionally, when we try to analyse the main idea of reasoning we often concentrate the focus more on the first few levels than deeper ones (the deepest level of MML, 21, is used in the Mizar article JGRAPH_6).

Our translation of proof graph structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, $A_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$ for an optimisation method strongly depends on the method. Generally, we define the search linearisation as follows:

```
1:  (declare-const n Int) (assert (= n“|𝒱|”))
2:  (declare-fun S (Int) Int) (declare-fun Sinv (Int) Int)
3:  (assert (forall ((x Int))
      (=> (and (<= 1 x) (<= x n)) (and (<= 1 (S x)) (<= (S x) n)))))
4:  (assert (forall ((x Int) (y Int))
      (! (=> (and(and (<= 1 x) (<= x n)) (= (S x) (S y))) (= x y))
      :pattern ((S x) (S y)))))
5:  (assert (forall ((x Int))
      (! (=> (and (<= 1 x ) (<= x n)) (= x (Sinv (S x)))) :pattern ((S x)))))
```

where `Sinv` corresponds to S^{-1} and is introduced only for selected MILs problems. We also assert `(<(S“x”) (S“y”))` where $\langle x, y \rangle \in \mathcal{A}$ and every path of $\mathcal{D}$ directed form x to y has length at most 1.

The choice of the interpretation of MIL problems in Z3 assertions, generally requires to carry out initial research on the impact of the translation choice on the time of solution search. We often obtain significant reduction of the searching time when we replace an assertion with a quantifier (e.g., $\forall_{1 \leq x \leq n}\, \varphi(x)$) by a list of assertions that represent individual cases (e.g., $\varphi(1), \varphi(2), \ldots, \varphi(n)$). As an illustration, let us consider the 1st interpretation of the 1st MIL defined as follows:

```
1:  (declare-fun T (Int) Bool)
2:  (assert (= (T (Sinv 1)) false))
3:  (assert (forall ((x Int)) (= (T (Sinv x))
      (ite (and(and (< 1 x) (<= x n)) (= (select A1 (Sinv (- x 1)) (Sinv x)) true))
      true false))))
4:  (declare-fun SumT (Int) Int)
5:  (assert (= (SumT 0) 0))
6:  (assert (forall ((x Int)) (= (SumT x) (ite(and(and(< 0 x) (<= x n)) (= (T x) true))
      (+ 1 (SumT (- x 1))) (SumT (- x 1))))))
```

where `A1` is the incidence matrix of the family of $\mathcal{A}_1$-arcs (`(declare-const A1 (Array Int Int Bool))`). An analysis of 285 one-level deductions contained in the proof script of [23] (e.g., Fig. 1) shows that we can speed up the search process (see Fig. 3) on average 37 times if we replace the assertion in the 6th row by a sequence of assentations like:

```
(assert (= (SumT“i+1”)
    (ite(= (T (Sinv“i+1”))true ) (+ 1 (SumT“i”)) (SumT“i”))))
```

for $i = 1, 2, \ldots, \mathtt{n}$ (the 2nd interpretation). Note that we replace a quantifier in this assertion by a list of individual cases and we count the number of *“true”* values of `T` in the order determined by S^{-1}. This result can be improved further on average 2.57 times, if we remove an intermediate function `T`, using a sequence of assentations like:

```
(assert (= (SumT“i+1”)
    (ite (= (select A1 (Sinv“i”) (Sinv“i+1”)) true) (+ 1 (SumT“i”)) (SumT“i”))))
```

(the 3rd interpretation). Note that the number of unresolved cases that remain unresolved after 10 minutes was reduced from 31 to 10 and 7 respectively. Additionally, the average time to solve newly resolved cases were 38.2 and 16.9 seconds respectively.

Similar speed up in searches was obtained with removal of quantifiers from interpretations of the other methods. It comes as no surprise since functions defined in one interpretation are used in other ones. Note that the function `SumT` is used indirectly in the interpretation of the 2nd MIL problem, since there exists a relationship between the values of `SumT` and a relation determined by belonging of two vertices to the same maximal linear reasoning.

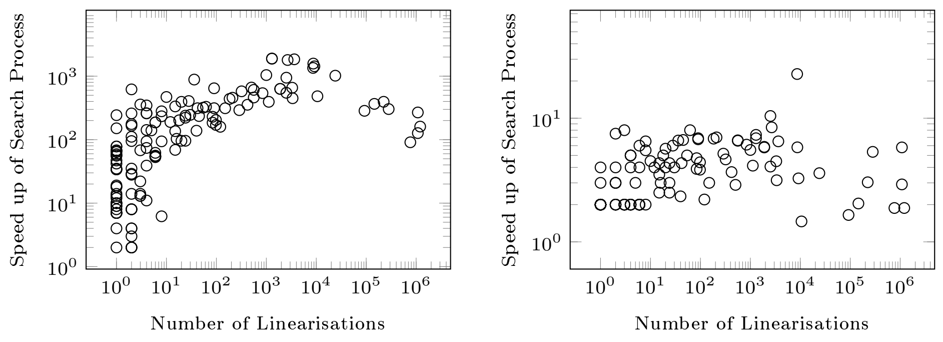

Fig. 3. The proportion between times of linearisation search with `Z3` strategy realised through 1st and 3rd interpretation (on the left side) and through 2nd and 3rd interpretation (on the right side) on one-level deductions in proof scripts of [23].

As an illustration, let us consider one-level deductions contained in the abstract proof graph presented in Fig. 2, induced by vertices 2, 3, 10–14 and 4–9 respectively. Note that the largest number of **`then`**-steps is equal to 3 in both deductions and this this value is obtained in the proof script presented in Fig. 1. However, the Z3 solver verify the possibility of increasing the number of **`then`**-steps in the 1st interpretation of the 1st MIL for 6.10 and 0.43 seconds, in the 2nd interpretation for 0.08 and 0.06 seconds, and in the 3rd interpretation for 0.02 and 0.01 seconds respectively.

5 Statistical Results

The effectiveness of `Z3` and `BF` strategies was studies on the MML database version `5.22.1191` including 208590 one-level deductions. We present here only statistical results that were obtained for two most popular criteria that correspond to the 1st and the 4th MIL problem. The results for other criteria have proved to be similar to those selected ones. This resemblance is mainly due to similar problem translation to Z3 solver. Running the optimisation program on the whole MML base with pragmas:

(1) `::$IL Z3:600 Then:<:0` , (2) `::$IL Z3:600 SumRef:<` ,
(3) `::$IL BF:1000000 Then:<:0` , (4) `::$IL BF:1000000 SumRef:<` ,

takes 62.5h, 16.4h, 2.4h, and 16 min respectively, on a platform with 16 Intel Xeon E5520 Processors and 24 GB of RAM. Using both strategies only 1.92%, 0.03% of problems were left unsolved, for `Then` and `SumRef` criterion respectively. Clearly, `BF` strategy turns out to be significantly better than `Z3`, but this holds only whole MML base is taken as the subject of the test case. When we take into consideration only deductions that possess at least ten million possible linearisations, the effectiveness in both cases becomes comparable. Additionally, the effectiveness of the strategy `Z3` is more highlighted when we analyse it in terms of the deductions length (see Fig. 4, 5). As we expected, application of SMT technology to long deductions, where the computationally hardest part of legibility improvement is concentrated, was fully justified. Additionally, this situation is evident even for deductions that have at least 25 steps.

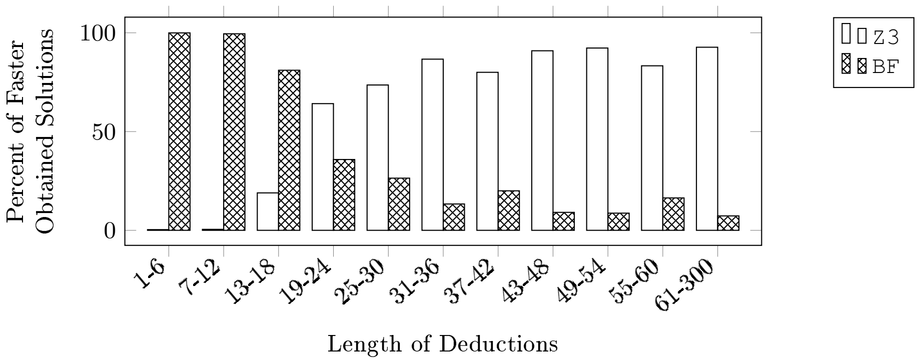

Fig. 4. The percent of one-level deductions, where the search time with `Z3:600` and `BF:10000000` strategies is compared depending on the deduction length, for criterion `SumRef:<`. Naturally, we limited results to cases, in which at least one strategy solved the problem.

6 Conclusions

In this paper we describe a next stage in the research on methods that improve proof legibility based on the modifying the order of independent deduction steps. The Mizar users need a "push-button" tool that automatically facilitates localisation of premises in deduction and highlights local subdeductions. However, creating such a tool is a non-trivial task, since the realisation of these expectations leads to NP-hard optimisation.

We presented initial results obtained with such a tool that uses the Z3 solver. This research showed that such tools can improve the legibility of deductions,

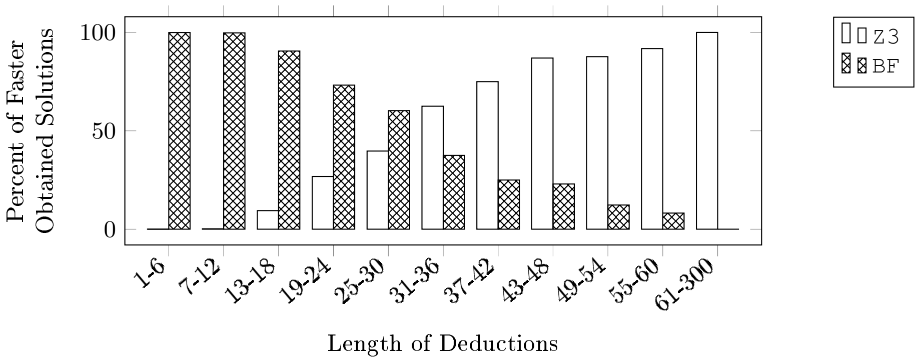

Fig. 5. The percent of one-level deductions, where the search time with `Z3:600` and `BF:10000000` strategies is compared depending on the deduction length, for criterion `Then:<:0`. Clearly, we limited results to cases, in which at least one strategy solved the problem.

even in the case they are long. Definitely, our result suggests the need of further work on the choice of interpretation of MIL problems in Z3 to speed up the search process

Note that we encounter computationally difficult problems, similarly as for MIL, when we try to improve the legibility of proof scripts based, e.g. on automated look-up of passages from long reasoning and extracting them as a lemma. Therefore, by successful application of the SMT-solver Z3 to solving MIL problems we we expect similar results to other methods of legibility enhancement.

References

1. O. Behaghel. Beziehungen zwischen Umfang und Reihenfolge von Satzgliedern. *Indogermanische Forschungen*, 25:110–142, 1909.
2. J. C. Blanchette. Redirecting Proofs by Contradiction. In *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013*, volume 14 of *EPiC Series*, pages 11–26. EasyChair, 2013.
3. N. Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–114, 2001.
4. L. de Moura and N. Bjørner. Z3: An effcient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer-Verlag, 2008.
5. K. A. Ericsson. *Analysis of memory performance in terms of memory skill*, volume 4 of *Advances in the psychology of human intelligence.* Hillsdale, NJ: Lawrence Erlbaum Associates Ins., 1988.
6. F. B. Fitch. *Symbolic Logic: an Introduction.* The Ronald Press Co., 1952.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* A Series of Books in the Mathematical Science. W. H. Freeman and Company, New York, 1979.

8. G. Gonthier. Formal Proof—The Four-Color Theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.
9. A. Grabowski, A. Korniłowicz, and A. Naumowicz. Mizar in a Nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
10. A. Grabowski and Ch. Schwarzweller. Improving Representation of Knowledge within the Mizar Library. *Studies in Logic, Grammar and Rhetoric*, 18(31):35–50, 2009.
11. S. Jaśkowski. On the Rules of Supposition in Formal Logic. *Studia Logica*, 1934. Warszawa Reprinted in Polish Logic, ed. S.McCall, Clarendon Press, Oxford 1967.
12. C. Kaliszyk and J. Urban. PRocH: Proof Reconstruction for HOL Light. In M. P. Bonacina, editor, *24th International Conference on Automated Deduction, CADE-24*, volume 7898 of *Lecture Notes in Computer Science*, pages 267–274. Springer-Verlag, 2013.
13. A. J. Ko, B. A. Myers, M. J. Coblenz, and Htet Htet Aung. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions On Software Engineering*, 32(12):971–988, 2006.
14. A. Korniłowicz. Tentative Experiments with Ellipsis in Mizar. In J. Jeuring, J. A. Campbell, J. Carette, Gabriel G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics 11th International Conference*, volume 7362 of *Lecture Notes in Artificial Intelligence*, pages 453–457. Springer-Verlag, 2012.
15. R. Levy. Expectation-based syntactic comprehension. *Cognition*, 106(2008):1126–1177, 2007.
16. R. Matuszewski. On Automatic Translation of Texts from Mizar-QC language into English. *Studies in Logic, Grammar and Rhetoric*, 4, 1984.
17. G. A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63:81–97, 1956.
18. A. Naumowicz and A. Korniłowicz. A Brief Overview of Mizar. In *TPHOLs'09, Lecture Notes in Computer Science, vol. 5674*, pages 67–72. Springer-Verlag, 2009.
19. A. Naumowicz and A. Korniłowicz. A Brief Overview of Mizar. In S. Berghofer, T. Nipkow, Ch. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science, vol. 5674*, pages 67–72. Springer-Verlag, 2009.
20. K. Pąk. The Algorithms for Improving and Reorganizing Natural Deduction Proofs. *Studies in Logic, Grammar and Rhetoric*, 22(35):95–112, 2010.
21. K. Pąk. Methods of Lemma Extraction in Natural Deduction Proofs. *Journal of Automated Reasoning*, 50(2):217–228, 2013.
22. K. Pąk. *The Algorithms for Improving Legibility of Natural Deduction Proofs*. PhD thesis, University of Warsaw, 2013.
23. K. Pąk and A. Trybulec. Laplace Expansion. *Formalized Mathematics*, 15(3):143–150, 2008.
24. S. J. Smolka and J. C. Blanchette. Robust, Semi-Intelligible Isabelle Proofs from ATP Proofs. In *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013*, volume 14 of *EPiC Series*, pages 117–132. EasyChair, 2013.
25. J. Urban. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In M. P. Bonacina, editor, *4th International Conference Mathematical Knowledge Management 2005, MKM'05*, volume 3863 of *Lecture Notes in Computer Science*, pages 346–360. Springer-Verlag, 2005.
26. J. Urban. MizarMode - An Integrated Proof Assistance Tool for the Mizar Way of Formalizing Mathematics. *Journal of Applied Logic*, 4(4):414–427, 2006.

27. M. Wenzel. *The Isabelle/Isar Reference Manual.* University of Cambridge, 2011.
28. A. N. Whitehead and B. Russell. *Principia Mathematica to *56.* Cambridge Mathematical Library. Cambridge University Press, 1910.
29. V. Zammit. *On the Readability of Machine Checkable Formal Proofs.* PhD thesis, The University of Kent at Canterbury, March 1999.