The Methods of Improving and Reorganizing Natural Deduction Proofs

Karol Pąk

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

Abstract. The interaction between a proof assistant system and its user can be improved not only by developing new features of the front-end applications, but also by offering the user enriched readability of the underlying formal language. Encoding of new mathematics is greatly simplified if the user has access to more legible texts in the library of already formalized theories. Although a rich language enables writing readable texts, it does not mean that all texts produced by authors sufficiently exploit the language's capabilities. It can be observed analyzing nontrivial examples of natural deduction proofs, either declarative or procedural, that authors tend to create deductions which are correct for computers, but hardly readable for humans, as they believe that finding and removing inessential reasoning fragments, or shortening the proofs is not so important as long as the computer accepts the proof script. To resolve this problem we created auxiliary applications to improve the quality of formal texts and we present them in this article.

1 Introduction

The databases of the formalized mathematics are constantly enlarging and achieve the considerable sizes because of adding to it more and more articles. Unfortunately, this enlargement of the databases in not always accompanied by the improvement of the quality of the formalization of the articles. Obviously, the legibility is a subjective notion and an individual matter for different authors.

There are some reasons of database's illegibility. Firstly, the articles are written not only by the users who know the content of the database and are able to use it, but also by inexperienced users. These new users do not know the whole database and because of that they prove unintentionally theorems which have been already proved. Moreover, new users learn from simple and not sophisticated articles which were written in the older versions of the system. Because of this fact, inexperienced users are often repeating the old proof strategies and do not use all possibilities of the database and of the system for verifying correctness of proofs. The second reason of illegibility consists of the fact that the majority of authors who develop and revise subsequent versions of a proof often add statements that might have been useful at some point of a revision, but are not really necessary for the final version of the proof. Finally, lack of the legibility of the database results from the fact that some proofs are excessively large, often because of manual revisions which unwittingly introduce unnecessary items.

To avoid this problem must be elaborated the uniform criteria of the proofs' legibility and then the tools which will cause that these criteria will be fulfilled. There are two possibilities to do it. Firstly, all articles can be improved manually, what requires thousands of hours of work. Secondly, it can be done with innovative programs which automatically shorten the considerable part of this manual and arduous work. It is possible because these auxiliary applications automatically standardize and shorten certain steps of deduction. The user of this application can establish the hierarchy of the criteria which must be used during the reorganization of the proof by this program. These criteria are described in the our paper are illustrated with Mizar system [7] in order to improve the quality of the Mizar Mathematical Library (MML) [13][1]. However, these techniques are useful in every declarative system basing on the natural deduction, created by S. Jaśkowski and F. B. Fitch [3, 4, 6].

The structure of this article is as follows. In Section 2 we present the abstract representation of proofs in the form of a graph, based on natural deduction of Jaśkowski. Subsequently, Section 3 presents selected types of excessive steps of deduction, which can appear in declarative formal proofs. This section also contains the description of problems occurring during the elimination of these steps of deduction. The knowledge of the Mizar system is not required to understand the problem even if the proofs which are described are represented in the Mizar style. In Section 4 we present the algorithms for reorganization and elimination, and also we report the statistical results obtained with the MML database.

2 The Proof Graph

When proofs written in a formal system are considered, graphs and directed trees are often employed to express the intuitions connected with the reasoning. In this section, on the basis of the natural deduction system, we will try to give a definition of the proof graph, which will express the most general approach to this question.

For analyzing the relations between consecutive steps in formal proofs in the system of S. Jaśkowski, the interpretation of the proof graph as a directed graph (digraph) of the reference $\mathfrak{R}$ is often used, where the individual steps of reasoning are the vertices and the directed edges define the relations between an expression and a previously justified fact used as the justification for that expression.

More precisely, the expressions α and β are connected with a directed edge pointing directly from α to β, if and only if there exist a set $\mathcal{R}$, such that $\alpha \in \mathcal{R}$ and the computer system can verify correctness of β using premises from $\mathcal{R}$ We would like that $\mathcal{R}$ not contain conclusions from β (then such a graph does not have directed cycles) and the cardinality of $\mathcal{R}$ is as small as possible.

[1] Results on the improvement of the quality of the base MML, which contains more than 1000 articles verified the Mizar proof checker, are on the website [10]

In such interpretation, the flow of information in the proof is well presented but the structure of the proofs is not preserved. In order to represent the structure, it is necessary to extend the analyzed graph with the relations describing the dependence between the expression and its proof.

Therefore, we can consider the proof $\mathcal{D}$, that can contain nested lemmas, as a finite sequence of families of proof graphs without these nested lemmas. To construct such sequence, the first family have to consist of only one element (the proof graph of $\mathcal{D}$ with cut nested lemmas). In the second step, we create a second family which consists of proof graph nested lemmas cut from the proof graph of $\mathcal{D}$. In the third step, we cut the nested lemmas from all proof graphs from the second family and we create from it the third family. We repeat the third step until the last family will not contain any nesting. Then, to regain the lost relations between the expression and its proof, we introduce an additional set of arcs, that we will call meta-edges. A meta-edge leads from β to α if and only if β is one of the steps in the reasoning of the proof of α.

The meta-edges describe relations between suitable vertices of families i and $i+1$. Obviously, the extended graph does not contain directed cycles, and for an arbitrary arc resulting from the reference connected directly from α belonging to the graph $\mathfrak{G}$ from the n-family to β belonging to the graph from the k-family we can say that

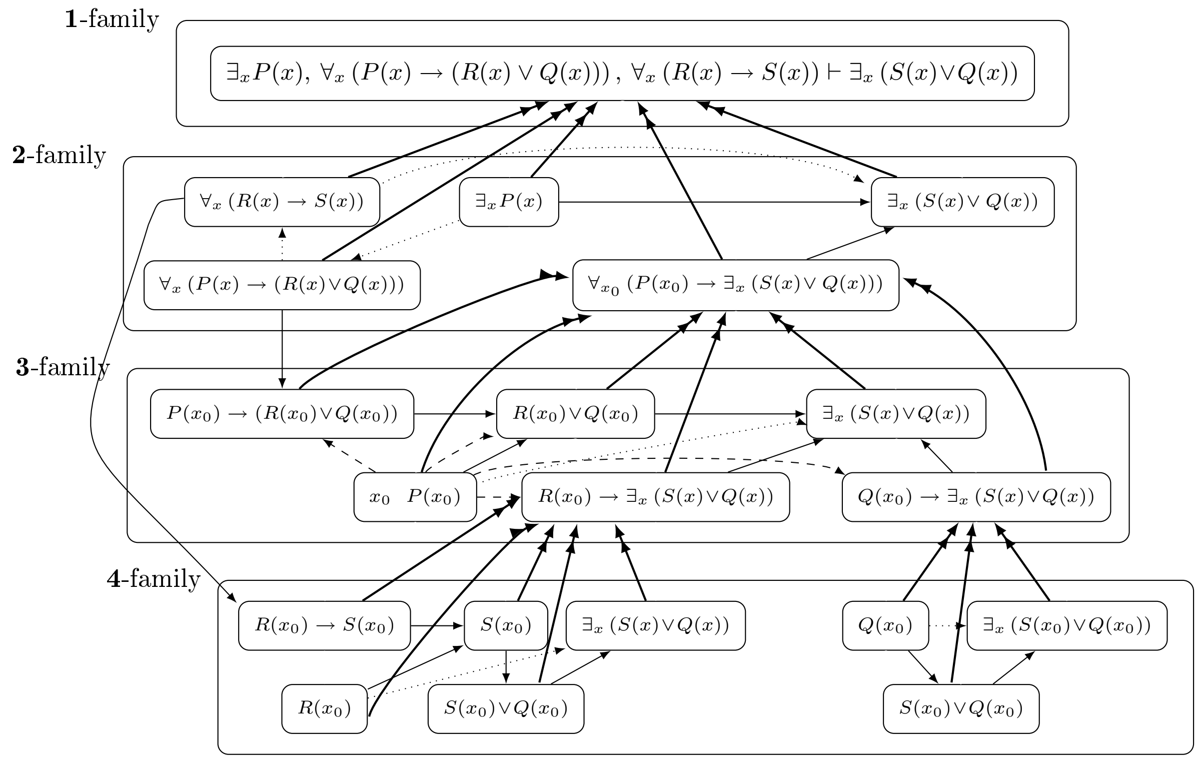
– $n \leq k$,
– there exists a path directed to meta-edges joining β and a chosen vertex in $\mathfrak{G}$.

This approach was created by Milewski [8] however it does not describe the sufficient number of relations necessary to reorganize the proof.

The following example shows a formal proof in Fitch notation and its associated graph with meta edges, which demonstrate this lack of the relations.

1		$\exists_x P(x)$			premise
2		$\forall_x\,(P(x) \rightarrow (R(x) \vee Q(x)))$			premise
3		$\forall_x\,(R(x) \rightarrow S(x))$			premise
4	x_0	$P(x_0)$			assumption
5		$P(x_0) \rightarrow (R(x_0) \vee Q(x_0))$			$\forall_x$e 2
6		$R(x_0) \vee Q(x_0)$			$\rightarrow$e 5,4
7		$R(x_0)$	assumption	$Q(x_0)$	assumption
8		$R(x_0) \rightarrow S(x_0)$	$\forall_x$e 3	$S(x_0) \vee Q(x_0)$	$\vee$i$_2$ 7
9		$S(x_0)$	$\rightarrow$e 9,8	$\exists_x\,(S(x) \vee Q(x))$	$\exists_x$i 8
10		$S(x_0) \vee Q(x_0)$	$\vee$i$_1$ 9		
11		$\exists_x\,(S(x) \vee Q(x))$	$\exists_x$i 10		
12		$\exists_x\,(S(x) \vee Q(x))$			$\vee$e 6,7-9;7-11
13		$\exists_x\,(S(x) \vee Q(x))$			$\exists_x$e 1, 4-12

The interpretation of the above-mentioned formal proof as a graph with meta-edges has the following structure, where $\rightarrow$ arrows represent the references and $\twoheadrightarrow$ arrows illustrate the meta-edges. The dashed and dotted arrows do not belong to these graph, their meaning will be described in the further part.


1-family
$\exists_x P(x),\ \forall_x\,(P(x) \rightarrow (R(x) \vee Q(x)))\,,\ \forall_x\,(R(x) \rightarrow S(x)) \vdash \exists_x\,(S(x) \vee Q(x))$
2-family
$\forall_x\,(R(x) \rightarrow S(x))$
$\exists_x P(x)$
$\exists_x\,(S(x) \vee Q(x))$
$\forall_x\,(P(x) \rightarrow (R(x) \vee Q(x)))$
$\forall_{x_0}\,(P(x_0) \rightarrow \exists_x\,(S(x) \vee Q(x)))$
3-family
$P(x_0) \rightarrow (R(x_0) \vee Q(x_0))$
$R(x_0) \vee Q(x_0)$
$\exists_x\,(S(x) \vee Q(x))$
$x_0\ \ P(x_0)$
$R(x_0) \rightarrow \exists_x\,(S(x) \vee Q(x))$
$Q(x_0) \rightarrow \exists_x\,(S(x) \vee Q(x))$
4-family
$R(x_0) \rightarrow S(x_0)$
$S(x_0)$
$\exists_x\,(S(x) \vee Q(x))$
$Q(x_0)$
$\exists_x\,(S(x_0) \vee Q(x_0))$
$R(x_0)$
$S(x_0) \vee Q(x_0)$
$S(x_0) \vee Q(x_0)$

The structure defined in this way enables the use of known facts and algorithms from graph theory. In order to use this structure during the reorganization of reasoning we have to extend it with the omitted dependencies between steps of the type "the natural deduction way of reasoning" called "skeleton steps" in further parts of this article (represented by dotted arrows):

– the universal quantifier introduction and the introduction of the existential quantifier,
– the implication introduction and the indication of the thesis,
– the introduction of reasoning by cases.

There are also omitted relations between the place of introduction, use and redefinition of type of variables (e.g. between $x_0\ \ P(x_0)$ and $P(x_0) \rightarrow (R(x_0) \vee Q(x_0)))$ (represented by dashed arrows) and other dependencies characteristic to a particular system (e.g. the Mizar system). The extension of the definitions to other above-mentioned dependencies would limit the legibility of the definition, and it would require the reader's intuitive understanding of the dependencies which can appear in the system of natural deduction. However, it is possible to generalize the definition of a proof structure, which is separated from the notion of the proof. It contains only three conditions.

Let us take a non-empty set V, and disjoint families M, E of ordered pairs from V.

The Proof Graph The structure $\mathfrak{P} \coloneqq \langle V, M, E\rangle$ will be called the proof graph if and only if

1. $\mathfrak{M} \coloneqq \langle V, M\rangle$ is a forest, i.e. a disjoint union of trees, in which every connected maximal tree is an arborescence, i.e a rooted tree with inverted direction (all arcs go in the direction from leaves to the root) [5].
2. An arbitrary arc (u, v) in the directed graph $\mathfrak{E} \coloneqq \langle V, E\rangle$ fulfills the condition: every nearest successor of u is a predecessor of v in the forest $\mathfrak{M}$.
3. The directed graph $\mathfrak{G} \coloneqq \langle V, M \cup E\rangle$ is acyclic.

The effect of inversion of direction consists in swapping the notions of child and parent (or predecessor and successor) in comparison to the natural trees.

To explain why we use the definition of the forest, in which every connected maximal tree is an arborescence with the root and inverted directions, let us define the auxiliary function $\mathfrak{l} : V \to \mathbb{N}$. The value of $\mathfrak{l}(v)$ is equal to the length of the directed path increased by one from v to the root in the connected maximal tree of $\mathfrak{M}$, which contains v. Then $\mathfrak{l}(v)$ we can interpret as a index of family from the sequence of families of proof graphs without nested lemmas, which contains v.

Additionally, graphs from individual families are defined by the notion of the nearest successor (inverted relation of siblings). Let us also notice that such introduction of the sequence of the families of graphs on the basis of forest $\mathfrak{M}$, causes that meta-edges connect statements of the family $i + 1$ with statements of the family i.

In graph theory the two conditions describe limitations imposed on the formal proof. Namely, the second condition says that the statements of reasoning of the provided theorem are invisible beyond that proof. Whereas, the third condition rejects existence of directed cycles which is equivalent to the application of provided theorem or conclusion from that theorem inside the proof.

Inside the family of arcs E we can distinguish three important subfamilies:

- $\mathcal{R}$ — arcs resulting from the use of facts which have been already proved in the justification of another rule;
- $\mathcal{V}$ — arcs defining the dependence between the introduction of the variable and its use;
- $\mathcal{S}$ — arcs defining the order of skeleton steps.

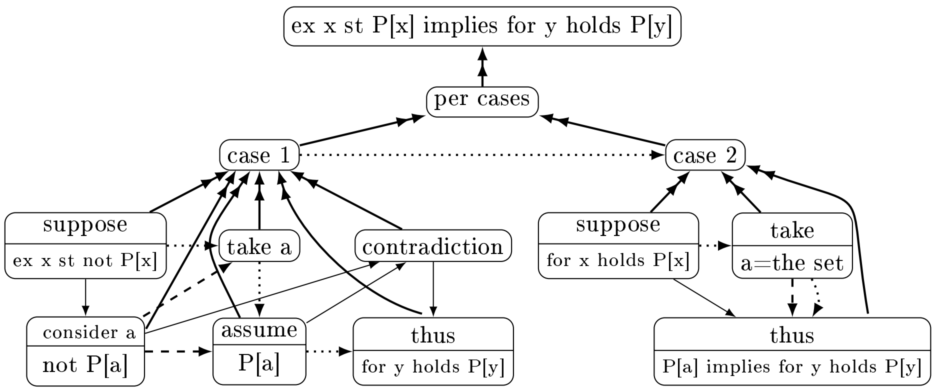
Obviously, the above-mentioned families are not disjoint and they do not exhaust the set E.

To illustrate the above-mentioned definition, let us consider the following example. We will represent the drinker's principle described in the article [15]. It states that "in every group of people one can point to one person in the group such that if that person drinks, then all the people in the group drink". The quoted proof is not indispensable for the Mizar system (an empty "semicolon" justification suffices to have it accepted by the checker), but a proof graph based on this reasoning illustrates well the subfamilies of the family E.

The reasoning in the Mizar style:

```
ex x st P[x] implies for y holds P[y]
  proof
    per cases;
      suppose A0: ex x st not P[x];
         consider a such that A1: not P[a] by A0;
         take a;
         assume A2: P[a];
         A3: contradiction by A1,A2;
         thus A4: for y holds P[y] by A3;
      end;
      suppose A5: for x holds P[x];
         take a=the set;
         thus P[a] implies for y holds P[y] by A5;
      end;
   end;
```

The proof graph looks like:



where ↠ arrows are subordinated to meta-edges and the continuous, dashed and dotted arrows are subordinated to suitable families $\mathcal{R}$, $\mathcal{V}$, $\mathcal{S}$, respectively.

The large number of arcs reduces the legibility of the graph, but enables to reconstruct the dependencies in reasoning. In the above-mentioned example, if we close transitively the graphs of individual cases and ignore the orientation of the arrows, we have a complete graph, what one can easily prove, impose an unambiguous order of steps. However such a graph generally does not enable an unambiguous reconstruction of the order of reasoning steps.

3 The Chosen Forms of Inessential Steps of Deduction

The presentation of unnecessary types of deduction in declarative formal proofs is hard to explain, but easy to illustrate. The basic types of redundant steps of deduction, such as:

– unnecessary references used in the justification of an expression,

– references which can be replaced by all references used to justify statements they point to,
– steps of deduction which are not used in any proof leading to the fact that some thesis or steps of deduction have no references pointing to them (vertices for which every thesis is not a successor in the proof graph),
– steps of deduction which can be totally replaced by external references, which can be found in justifications of these steps of deduction.

are already resolved (see [9]) and in this aim the necessary auxiliary applications (e.g. Relprem, Relinfer, Inacc or Trivdemo) have already been created.

These auxiliary applications have enabled to discover the next important problem that was not resolved. We can qualify it as "covered by &". To describe this problem let us consider two deductions merged by the author into one reasoning.

```
$\alpha_1$ implies $\alpha_n$   $\beta_1$ implies $\beta_m$    $\alpha_1$ & $\beta_1$ implies $\alpha_n$ & $\beta_m$
proof                proof                 proof
  assume $\alpha_1$;         assume $\beta_1$;          assume $\alpha_1$ & $\beta_1$;
  then $\alpha_2$;           then $\beta_2$;            then $\alpha_2$ & $\beta_2$;
  ⋮                    ⋮                     ⋮
  hence $\alpha_n$;          then $\beta_n$;            then $\alpha_n$ & $\beta_n$;
end;                   then $\beta_{n+1}$;        then $\alpha_n$ & $\beta_{n+1}$;
                       ⋮                     ⋮
                       hence $\beta_m$;           hence $\alpha_n$ & $\beta_m$;
                     end;                  end;
```

Such a merge does not cause mistakes in the reasoning, but can contain some repeated expressions (not only the expression α_n). Let us also notice that in such a proof, e.g. to prove the rule α_{i+1} an unnecessary step, β_i, is used. Moreover, the rule α_i & β_i can be necessary in reasoning despite the fact that one of the steps α_i or β_i is not necessary.

Another problem which can occur in the merged parallel deduction consist of removing by the author some part of the thesis (e.g. β_m) without changing the assumptions and proof. Such change limits, in an important way, the statement; however the proof is still correct. None of the above-mentioned auxiliary applications can detect such cases. It is easy to notice that the creation of algorithm that could automatically find such cases is incomparably more difficult than the creation of the above-mentioned auxiliary application. Such an algorithm should consider all deductions and not only one step as it is done by the auxiliary applications created until now.

Our auxiliary application solves much more complicated problems, such as the following example illustrates:

```
α1 & β1 implies α4
proof
  assume A1:α1 & β1;
  α4 & β4
   proof
    α2 & β2 by A1;
    then α3 & β3;
    hence α4 & β4;
   end;
  hence α4;
end;
```

Our auxiliary application has found numerous cases of such problem not only in lemmas but also in 541 theorems in MML (it means, on average, one in every 100 theorems), as in the exemple [12]:

```
theorem :: NAGATA_1:20
   for T being non empty TopSpace st T is regular & T is T_1 &
     ex Bn being FamilySequence of T st Bn is Basis_sigma_locally_finite
  holds T is normal;
```

in which the assumption `T is T_1` is unnecessary, although the theorem was formalized on the base of the book [2] (in the book, the attributes T_3 and "regular" are synonymous, while in the Mizar system every T_3–space is a regular T_1–space, but the attributes T_1 and "regular" are not the same).

The solution presented in Section 4 breaks conjunctions and removes unnecessary steps that were created in this way, in the whole base MML. Is to easy to notice that the majority of the steps of deduction which contain conjunctions has become illegible. The analyzed proof of the step α_1 `&` β_1 `implies` α_n `&` β_m can look as follows:

```
α1 & β1 implies αn & βm
proof
  assume that A1:α1 and A2:β1;
  A3: α2 by A1;
  A4: β2 by A2;
  A5: α3 by A3;
  A6: β3 by A4;
  A7: α4 by A5;
  ⋮
```

In order to present this problem precisely, let us consider a simple Mizar-style proof, whose proof graph well illustrates typical situations met during the reorganization of the proof.

```
theorem
  i in Seg n implies i+m in Seg(n+m)
proof
  assume i in Seg n;
  then 1 <= i & i <= n & i <= i+m by NAT_1:11,FINSEQ_1:3;
  then 1 <= i+m & i+m <= n+m by XREAL_1:9,XXREAL_0:2;
  hence thesis by FINSEQ_1:3;
end;
```

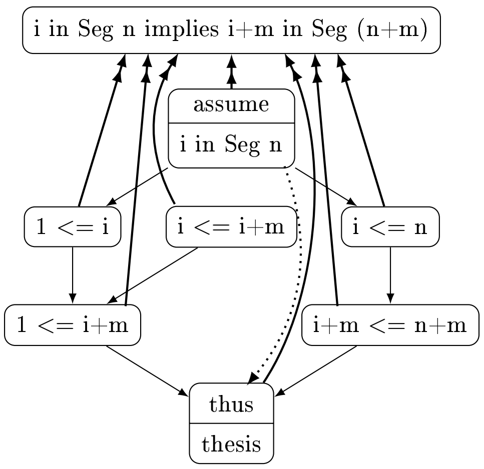
where `i`, `n`, `m` $\in \mathbb{N}$ and `Seg n={1,...,n}`. We can also analyze the above-mentioned reasoning in a different system, e.g. in the Isabelle style [11]:

```
lemma
  fixes n i m :: nat
  assumes a: "i \<in> {k :: nat . 1 <= k & k <= n}"
  shows "i + m \<in> {k :: nat . 1 <= k & k <= n + m}"
proof -
  have "1 <= i & i <= n & i <= i+m" using a by auto
  then have "1 <= i+m & i+m <= n+m" by auto
  then show ?thesis by auto
qed
```

Having broken all conjunctions of the reasoning and having simplified the lists of the reference, we obtain:

```
theorem
  i in Seg n implies i+m in Seg (n+m)
proof
  assume A1:i in Seg n;
  then A2:1 <= i by FINSEQ_1:3;
  A3:i <= n by A1,FINSEQ_1:3;
  i <= i+m by NAT_1:11;
  then A4:1 <= i+m by A2,XXREAL_0:2;
  i+m <= n+m by A3,XREAL_1:9;
  hence thesis by A4,FINSEQ_1:3;
end;
```

and proof graph:



The above-mentioned proof's graph enable simple understanding of the next steps of reasoning and finding references used in consecutive steps. Such an equally legible way in the system of natural deduction is a crucial idea of the proof's reorganization. This essential point consists of determination of criteria which can improve the legibility of formal proofs in the system of natural deduction. Having analyzed the different opinions of users of database we propose the following four criteria of legibility of deduction:

1. maximization of the length of the paths where every consecutive justification should refer to a previous line (reference with words "then", "hence" without introducing new labels) and, if it is possible, to a minimal number of different labels which does not belong to this path,
2. minimization of the quantity of introduced labels,
3. minimization of the total length of jumps to distant, previously justified facts,
4. presentation, in the coherent entirety of the proof, of reasoning steps which in the proof graph locally create the sub-deduction.

As we mentioned in the introduction, establishing of the degree of importance of particular criteria is a controversial matter. So we created a flexible application which can be used with different parameters which enable modification of the criteria's hierarchy. The users chose the most often two first criteria or the third one as the predominant. If we compare the results obtained for these two different situations, we get two various figures, which are presented below:

```
1  theorem
2   i in Seg n implies i+m in Seg(n+m)
3  proof
4   A1: i<=i+m by NAT_1:11;
5   assume A2: i in Seg n;
6   then i<=n by FINSEQ_1:3;
7   then A3: i+m<=n+m by XREAL_1:9;
8   1<=i by A2,FINSEQ_1:3;
9   then 1<=i+m by A1,XXREAL_0:2;
10  hence thesis by A3,FINSEQ_1:3;
11 end;
```

```
1  theorem
2   i in Seg n implies i+m in Seg(n+m)
3  proof
4   assume A1: i in Seg n;
5   then A2: 1<=i by FINSEQ_1:3;
6   i<=n by A1,FINSEQ_1:3;
7   then A3: i+m<=n+m by XREAL_1:9;
8   i<=i+m by NAT_1:11;
9   then 1<=i+m by A2,XXREAL_0:2;
10  hence thesis by A3,FINSEQ_1:3;
11 end;
```

In the first case, we find two chains of three steps (lines 5, 6, 7; 8, 9, 10), whereas in the second case there is only one chain of three steps (lines 8, 9, 10) and two chains containing two steps (lines 4, 5; 6, 7). The first criterion does not define whether it is more natural to create a maximal chain with many, often one-step chains, or to formulate several "average" ones.

Having analyzed the total distance of jumps between a label and its use, we observe that creating longer chains enlarged the total distance of jumps of two steps.

If we count the number of labels in the above-mentioned reasoning, we see that there are three of them in both cases. We can prove that this is the minimal number of labels for this proof. Moreover, a maximal anti-chain in the transitive closed proof graph of this reasoning has at most three elements. This dependence is often a loose relationship. The number of labels can be just a little smaller, e.g. in graphs of references which look as in the first example, or many times larger (the second example).

$$\begin{tikzcd}
 & \alpha_1 \arrow[dl] & \\
\beta_1 \arrow[dr] & & \alpha_2 \arrow[dl] \\
 & \beta_2 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & \beta_1 \arrow[dr] & & \beta_2 \arrow[dr] & & & \beta_n \arrow[dr] & \\
\alpha_1 \arrow[ur] \arrow[dr] & & \alpha_2 \arrow[ur] \arrow[dr] & & \alpha_3 \arrow[r, dotted, no head] & \alpha_n \arrow[ur] \arrow[dr] & & \delta \\
 & \gamma_1 \arrow[ur] & & \gamma_2 \arrow[ur] & & & \gamma_n \arrow[ur] &
\end{tikzcd}$$

(in the second example, an arbitrary maximal anti-chain has at most two elements, but the number of labels is estimated by $2 \cdot n$).

The best way of estimating the number of labels in a reference graph consists of counting the vertices whose out–degree is at least two; and the number of the arcs (u, v) for which the in–degree of v is at least two and the vertex v does not have yet the label (the number of entering arcs without a label is at the most). If we take into consideration all arcs in the proof graph, it enlarges, in the general case, the number of labels counted in this way.

4 Auxiliary Applications and Statistical Results

The problems described in the previous section can be solved with two independent sets of programs. The aim of the first one consists of finding and removing as many as possible unnecessary steps of reasoning hidden in "&". To this end, the existing utilities have been extended with five programs, which we describe below. The aim of the second sort of programs is the reorganization of the order of proof steps. This sorting was made with the application SortItem, which preserves correctness of the reasoning and relations in the proof graph. Statistical results were obtained on the MML database version `4.121.1054` and were introduced to the version `4.127.1060` [10].

BreakBinaryAnd The application breaks in a simple way (with some not so important exceptions) all statements that contain a conjunction. The application changes the sequence of expressions joint with the conjunction into the sequences of consecutive steps of reasoning, which include the suitable elements of that sequence of expressions and the identical list of the reference used in the reasoning. Breaking all of the above-mentioned conjunctions has enabled finding in MML of 767139 unnecessary references in the justifications, which caused a transmission of unnecessary sequences of conjunctions. Moreover, we found 39745 steps in deductions which were unnecessary for their correctness and we could remove them.

DelBlock The application significantly breaks the proofs of conjunction statements which do not contain implicit universal quantifiers. Moreover, the deductions cannot contain the elimination of a universal or existential quantifier and the introduction of reasoning per cases.

We can describe the transformation made in the general case in the following style:

```
Lab1: α & β
 proof
  ...                  ...
  thus α;              Thus1: α;
  ...                  ...
  thus β;              Thus2: β;
 end;                Lab1: α & β by Thus1,Thus2;
```

RenInfer The application collaborates with the auxiliary application RELINREF. The program changes selected references for which Relinfer reports the message “604” it means references which can be replaced by all references used to their justification). The program creates a list of labels sorted using three criteria with decreasing significance:

1. the number of references to a particular label without the message “604” is minimal,
2. the number of all references to a particular label is minimal,
3. the number of references with message “604”, used as justification for the statement assigned to this label, is minimal.

Then, for a label selected in this way, the program replaces some of its uses by all references used to justify the expressions joined with the label.

Obviously, every time such a label is chosen, the labels of statement whose list of justifications will be modified are ignored in the next search. The described algorithm does not remove all “604” messages. After one use of the program, on average 88,7% of messages “604” is removed, and all these messages are removed on average after one and a half (1,532) runs of this program.

TrivConsider This application found 6154 cases in which removing introduced variables was possible using construction “consider” (the incorrectness in modified reasoning occurred only in two cases). Such introduction of existential quantifiers enabled finding new unnecessary steps in deductions. It is interesting to notice that after removing unnecessary steps, the application could find another 59 cases.

MergeItems The program finds statements always used together in the reasoning, and then it tries to merge them into a conjunction. Obviously, none of the statements can be a successor of the other one in the proof graph. To avoid creating long list of references, which can lengthen the time needed to verify joint statements, in the justification of joint statements it is required that the lists of the references are compatible. The level of compatibility is determined by users.

The use of the five above-mentioned programs enabled finding in MML the theorems with unnecessary assumptions. Removing these assumptions enabled finding next unnecessary steps and statements in deduction. The statistical results are presented in the following table:

Stages	Unnecessary references	Message "604"	Unnecessary inferences	Altered articles	Altered theorems
1	755196	37304	38944	1017	461
2	1640	23	633	118	64
3	596	2	168	55	16

SortItem The program creates a proof graph for a particular article and, on the basis of it, it reorganizes the order of statements in reasoning. The algorithm of reorganization is based on a successive recurrent joining of sequences of sub-deduction chosen with the imposed criteria which do not cause the conflict in the graph (i.e. no vertex from first reasoning is the predecessor of a vertex from the second reasoning). Using a greedy algorithm, which solves the problem of making the locally optimal choice at each stage, often does not enable to find an optimal global solution, but in contrast to other algorithms, enables a coherent presentation of local sub-deductions.

Let us introduce the auxiliary functions for a proof graph, in order to describe the basic criteria in a legible way $\langle V, M, E\rangle$.

Let us take a subset A of the set E. We define three functions:

$$\begin{gathered} \textit{in–deg}_A(v) = |\{w : w \in V \ \wedge \ (w,v) \in A\}|, \\ \textit{out–deg}_A(v) = |\{w : w \in V \ \wedge \ (v,w) \in A\}|, \\ I_A(F) = |\{(i,j) : 1 \leq i,j \leq n : \wedge \, (f_i, f_j) \in A\}, \end{gathered}$$

where v is an arbitrary vertex, and $F = \{f_i\}_{i=0}^{n}$ is a sequence of vertices from the family E.

In this case the minimization of the number of labels on the base of two basis criteria with decreasing significance is important (where $F^j = \{f_i^j\}_{i=0}^{n_j}$ for $j = 1,2$, and $F^1 \frown F^2$ is the concatenation of F_1, F_2).

1. The aim of the first criterion is finding the pairs of sequences for which the following condition is fulfilled: $(f_{n_1}^1, f_1^2) \in R$, and then it chooses the pairs for which the value $\textit{out–deg}_R(f_{n_1}^1)$ is minimal, and then chooses the pairs for which the value $\left(\sum_{i=1}^{n_1} \textit{out–deg}_R(f_i^1)\right) - I_R(F^1)$ is minimal.
2. The aim of the second criterion is finding the two pairs of sequences for which the number of dysfunctions (i.e. the increase of total distance between places of introduction of the label and its uses) created after merging these sequences is minimal The number of dysfunctions is described on the basis of the relation:

$$n_1 \cdot \left(\sum_{i=1}^{n_2} \textit{in–deg}_R(f_i^2))\right) + n_2 \cdot \left(\sum_{i=1}^{n_1} \textit{out–deg}_R(f_i^1)\right) - (n_1 + n_2) \cdot I_R(F^1 \frown F^2).$$

Apart from the main criteria there are several auxiliary ones, quite difficult to describe. Thanks to all these criteria the order of writing the proof tree becomes much more unequivocal.

Statistical Results

Having measured (in tokens) the length of articles we observe that despite an important number of modifications, the length of articles did not change in an important way (the articles were reduced on average by 0,31%). On the other hand, having analyzed the new form of the articles, we observed that the modification caused a reduction of the length of formulas on average by 3,12%, but the length of the references was extended by 3,7%. The verification time of

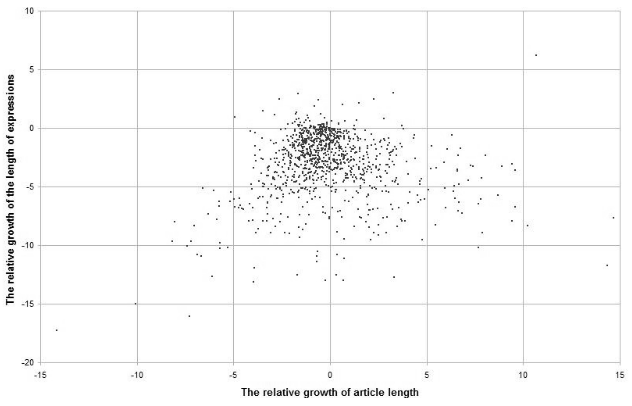


Fig. 1. The change of article length

an article (time of mizaring) is another measure which can be used to describe the modifications. This time on average was reduced by 5,13%.

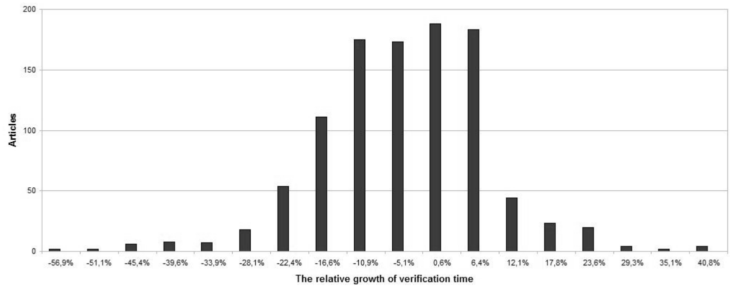


Fig. 2. The change of verification time

Final Remarks

Thanks to algorithms presented in our paper many advantages have been obtained. Firstly, our algorithms constitute in the Mizar system fully automatic ways of standardizing the reasoning's structure based on the imposed criteria in the aim of legibility's improvement (so far, it was done mainly manually). Secondly, the reasoning's structure was improved in important way and the particular steps of deduction have became more readable for users of database. In consequent, using of database has become easier. Thirdly, our algorithm of the proof's reorganization can be used independently to the introduction of the results of the different experiments which have not enabled to maintain the readable proof's structure. It is really important because databases are public and the perseverance of the high level of quality is a priority. It does not change the fact that our algorithms are the first complex tool which enable both the shortening of the database and the preservation and even improvement of its legibility. Fourthly, an average time of the verification of articles of database has been shorten. And last but not least, shortening of the number of steps of deduction (about 3%) and reduction of the number of assumptions in about 1% theorems may does not seem at a first glance an impressive result. However the time necessary to obtain such effects manually is comparable with time of manual calculation of 35 decimals of π by Ludolph van Ceulen.

References

1. E. Bonarska, *An Introduction to PC Mizar*, Fondation Ph. le Hodey, Brussels, 1990.
2. R. Engelking, *General Topology*, PWN – Polish Scientific Publishers, Warsaw, 1977.
3. F. B. Fitch. *Symbolic Logic: an Introduction.* The Ronald Press Co., New York, 1952.
4. S. Jaśkowski, *On the Rules of Supposition in Formal Logic*, Studia Logica I, 1934, Warszawa Reprinted in Polish Logic, ed. S.McCall, Clarendon Press, Oxford, 1967.
5. B. Jorgen, G. Gregory, *Digraphs: Theory, Algorithms and Applications*, Springer, ISBN 1-85233-268-9, 2000.
6. W. Marciszewski, *A Jaśkowski-Style System of Computer-Assisted Reasoning*, Philosophical Logic in Poland, Kluwer, 1993.
7. R. Matuszewski, P. Rudnicki, *MIZAR: the first 30 years*, Mechanized Mathematics and Its Applications, 4(**1**), p. 3–24, 2005.
8. R. Milewski, *Algorithms analyzing formal deduction support systems*-PhD thesis, The Computer Science Faculty of Białystok Technical University, Białystok, 2008.
9. R. Milewski, *New Auxiliary Software for MML Database Management Mechanized Mathematics and Its Applications*, ISSN 1345-8272, Vol. 5, No. 2, p. 1–10, 2006.
10. *Mizar Home Page*, `<http://mizar.uwb.edu.pl/>`.
11. L.C. Paulson, *The Isabelle Reference Manual*, 2000.
12. K. Pąk, *The Nagata–Smirnov Theorem. Part I*, Formalized Mathematics 12(3), p. 341–346, 2004.
13. P. Rudnicki, *An Overview of the Mizar Project*, Proceedings of the 1992 Workshop on Types for Proofs and Programs, Chalmers University of Technology, Bastad, 1992.
14. A. Trybulec, *Some features of the Mizar language*, Presented at a workshop in Turin, Italy, 1993.
15. F. Wiedijk, *Mizar Light for HOL Light*, Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics, p. 378–394, September 03–06, 2001.