STUDIES IN LOGIC, GRAMMAR AND RHETORIC 22 (35) 2010

Karol Pąk
University of Białystok
Institute of Computer Science, Białystok, Poland

THE ALGORITHMS FOR IMPROVING AND REORGANIZING NATURAL DEDUCTION PROOFS

Abstract: It can be observed in the course of analyzing nontrivial examples of natural deduction proofs, either declarative or procedural, that the proofs are often formulated in a chaotic way. 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.
This article consists of two parts. In the first part, we present some types of unnecessary deductions and methods of reorganizing proof graphs in order to make them closer to good quality informal mathematical reasoning. In the second part, we describe tools implemented to solve the above-mentioned problems. Next, we demonstrate their usability by analysing statistical data drawn from the Mizar Mathematical Library.

1. Introduction

The databases of formalized mathematics are constantly growing and have achieved considerable sizes through the addition of numerous articles. Unfortunately, this enlargement of the databases is not always accompanied by the improvement of the quality of the formalization of the articles. Obviously, the legibility is a subjective notion and is an individual matter for different authors.

There are various 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 the proofs.

ISBN 978–83–7431–273–8 ISSN 0860-150X

The second reason of the 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 revision, but are not actually necessary for the final version of the proof. Finally, the 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 the uniform criteria of the proofs' legibility must be elaborated and then the tools which will cause that the theses criteria will be fulfilled. There are two possibilities of doing 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 the 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. The criteria described in our paper are illustrated by the Mizar system [6] in order to improve the quality of the Mizar Mathematical Library (MML) [12][1]. However, theses techniques are useful in every declarative system based on the natural deduction, created by S. Jaśkowski and F. B. Fitch [2, 3, 5].

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 the reorganization and elimination, and also we report the statistical results obtained through 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,

[1] Results of 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]

we will attempt 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 exists set $\mathcal{R}$, such that $\alpha \in \mathcal{R}$ and the computer system can verify the correctness of β using premises from $\mathcal{R}$ Obviously, we would like that $\mathcal{R}$ not contain conclusions from β (then such a graph does not have directed cycles) and the number of $\mathcal{R}$ is possibly the least.

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 $\mathcal{D}$ with possible nested lemma as a finite sequence of families of proof graphs without nested lemma. To construct such a sequence, the first family has to cosist of only one element (the proof graph of $\mathcal{D}$ with cut nested lemma). In the second step, we create a second family which consists of proof graph nested lemma cut from the proof graph of $\mathcal{D}$. In the third step, we cut the nested lemma 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, these 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

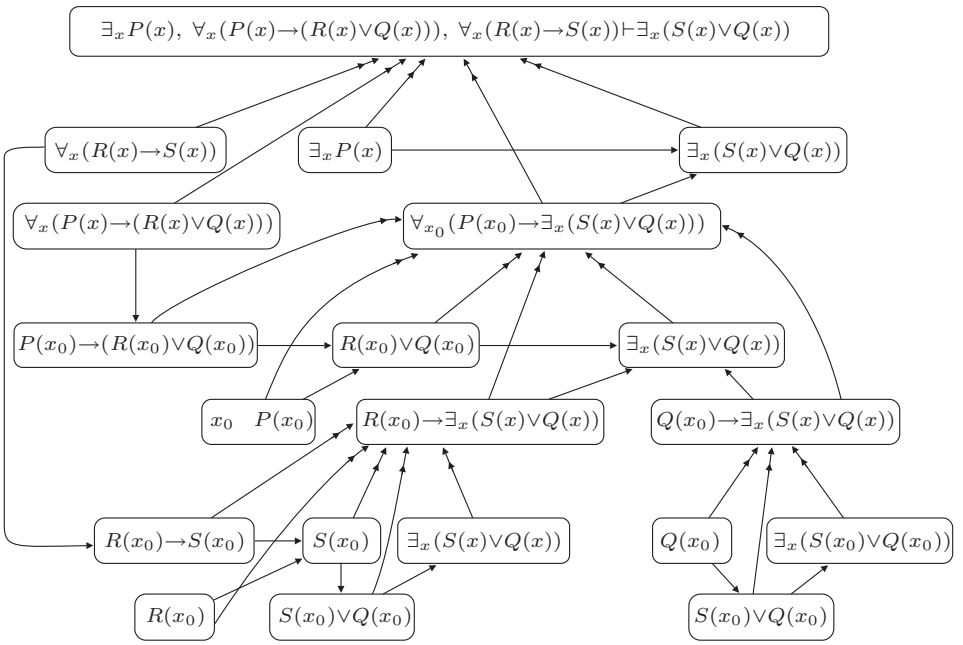
- $n \leq k$,
- there exists a path directed to meta-edges joining β and a chosen vertex in $\mathfrak{G}$.

This approach was created by Milewski [7] however, it does not describe the sufficient number of relations necessary to proof's reorganization.

As an illustration of the above-mentioned reasoning, let us consider the following example based on the Fitch notation of natural deduction. This exemple will demonstrate this lack of the relations.

1		$\exists_x P(x)$			premise
2		$\forall_x (P(x) \to (R(x) \vee Q(x)))$			premise
3		$\forall_x (R(x) \to S(x))$			premise
4	x_0	$P(x_0)$			assumption
5		$P(x_0) \to (R(x_0) \vee Q(x_0))$			$\forall_x$e 2
6		$R(x_0) \vee Q(x_0)$			$\to$e 5,4
7		$R(x_0)$	assumption	$Q(x_0)$	assumption
8		$R(x_0) \to S(x_0)$	$\forall_x$e 3	$S(x_0) \vee Q(x_0)$	$\vee$i$_2$ 7
9		$S(x_0)$	$\to$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 $\to$ arrows represent the references and $\twoheadrightarrow$ arrows illustrate the meta-edges.



The structure defined in this way enables the use of known facts and algorithms from the 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:

- 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 $P(x_0) \rightarrow (R(x_0) \vee Q(x_0))$) 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 mode, were the definition 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 $\mathfrak{P} := \langle V, M, E\rangle$ will be called the proof graph, if and only if,

1. $\mathfrak{M} := \langle V, M\rangle$ 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) ([4]).
2. An arbitrary arc (u, v) in the directed graph $\mathfrak{E} := \langle V, E\rangle$ fulfills the condition: every nearest successor of u is a predecessor of v in the forest $\mathfrak{M}$.
3. The directed graph $\mathfrak{G} := \langle V, M \cup E\rangle$ 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 $\mathfrak{l} : V \rightarrow \mathbb{N}$. The value of $\mathfrak{l}(v)$ is equal to the length of the directed path from v to the root plus one in the connected maximal tree, which contains v.

Then the value of the function $\mathfrak{l}$ represents vertices which belong to the suitable family from the sequence of graphs (more exactly, the vertex v

belongs to $\mathfrak{l}(v)$ the family). 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 subfamilies

- $\mathcal{R}$ – the 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.

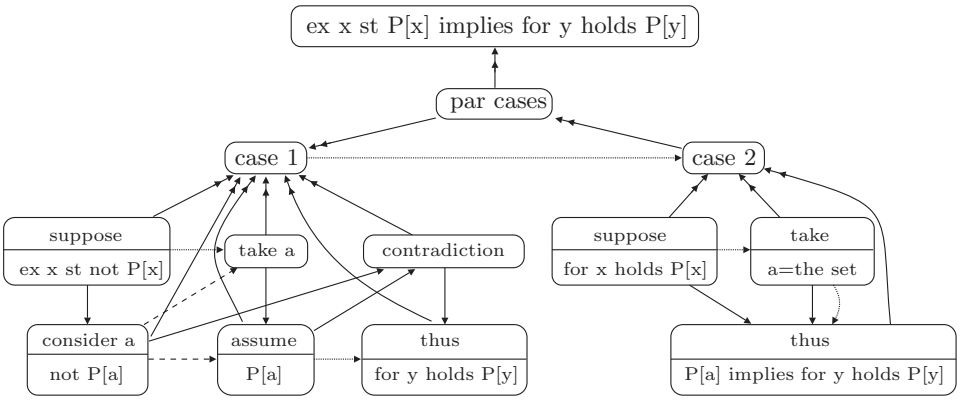
To illustrate the above-mentioned definition, let us consider the following example. We will represent the drinker's principle described in the article [14]. "This says that in every group of people one can point to one person in the group such that if that person drinks, then all the people in the group 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 $\twoheadrightarrow$ arrows are subordinated to meta-edges and the continuous, dashed and dotted arrows are subordinated to suitable families $\mathcal{R}$, $\mathcal{V}$, $\mathcal{S}$.

The large number of arcs reduces the legibility of the graph, but enables to reconstruct the dependencies in reasoning. Such a graph form generally does not enable an unambiguous reconstruction of the order of reasoning steps. In the above-mentioned example, when we close transitively the graphs of individual cases, we obtain complete graphs, what one can easily prove, impose an unambiguous order of steps, however the removal of the arc "case 1 $\rightarrow$ case 2" enables changing the order of two cases.

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

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

```
α1 implies αn          β1 implies βn          α1 & β1 implies αn & βn
proof                  proof                  proof
  assume α1;             assume β1;             assume α1 & β1;
  then α2;               then β2;               then α2 & β2;
  then α3;               then β3;               then α3 & β3;
  ⋮                      ⋮                      ⋮
  then αn;               then βn;               then αn & βn;
  hence thesis;          hence thesis;          hence thesis;
end                    end                    end
```

Such a merge does not cause mistakes in the reasoning, but can contain some repeated expressions (if the lengths of deduction are different). 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, consists of removing by the author some part of the thesis (e.g. β_n) 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 algoritm that could automatically find such cases is incomparably more difficult than the creation of the above-mentioned auxiliary application. Such algoritm should consider all deductions and not only one step as it is done by the auxiliary applications created until now.

Our auxiliary application enables resolving of 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;
      then α4 & β4;
      hence thesis;
    end;
  hence thesis;
end;
```

Our auxiliary application has found numerous cases of such problem not only in lemma but also in 541 theorems in MML (it means, on average, one in every 100 theorems).

The presented solution which consists of breaking chosen conjunctions and removing unnecessary steps that were created in this way, in the whole base MML, is the subject of Section 4 of this article. Is too 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_n, can look as follows:

```
α1 & β1 implies αn & βn
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;
```

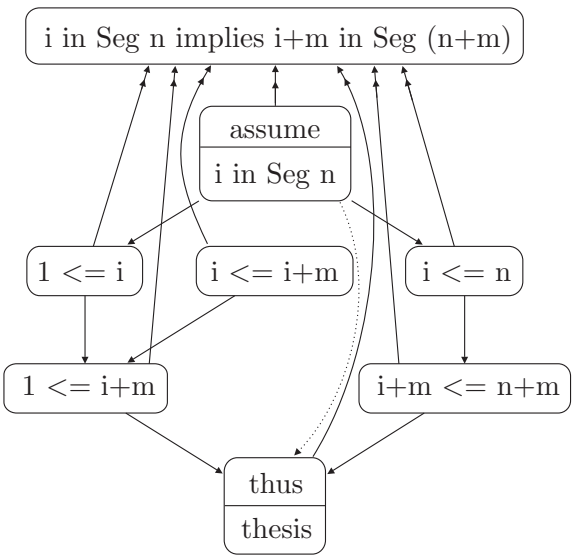
were `i, n, m` are natural numbers and `Seg n={1,...,n}`. We can also analyze the above-mentioned reasoning in a different system, e.g. in the Isabelle style [11]:

```
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:



We have left in the above-mentioned graph only the arcs resulting from references in order to enable simple understanding of the next steps of reasoning and finding references used in consecutive steps.

The presentation of the above-mentioned graph in equally legible way in the system of natural deduction is a crucial idea of the proof's reorganization. This essential point consists of determining the 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 in which every consecutive justification should refer to a previous line and, if it is possible, to a minimal number of different labels,
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 it has been 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 even by users with opposed criteria's hierarchy of legibility. The users chose the most often two first criteria or the third one as the predominant. If we compare the results obtained for theses two different situations, we get two various figures, which are presented below:

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

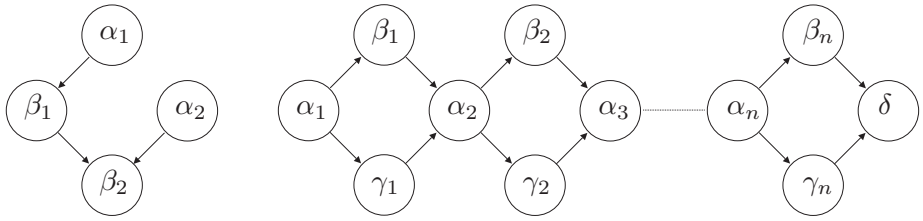
```
theorem
  i in Seg n implies
  i+m in Seg(n+m)
proof
 assume A1: i in Seg n;
 then A2: 1<=i by FINSEQ_1:3;
 i<=n by A1,FINSEQ_1:3;
 then A3: i+m<=n+m by XREAL_1:9;
 i<=i+m by NAT_1:11;
 then 1<=i+m by A2,XXREAL_0:2;
 hence thesis by A3,FINSEQ_1:3;
end;
```

In the first case, we find two chains of three elements, whereas in the second case there is only one chain of three elements and two chains containing two elements. The first criterion does not define whether it is more natural to create one maximal chain with many, often one-element chains,

or to formulate several chains of average length, without chains containing only one-element.

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 exactly by 2.

If we count the number of labels in the above-mentioned reasoning, we see that there are exactly three 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).



(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 outdegree is at least two; and the number of the arcs (u, v) for which the indegree 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 it.

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 signification, such as:

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 the 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 theses messages are removed on average after 1,532 uses 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 the deductions. It is interesting to notice that after removing unnecessary steps, the application could find other 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 comptibility 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 the deduction. The statistical results are presented in the following table:

Stages	Unnecessary references	Message "604"	Unnecessary inferences	s 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 the global solution, but in contrast to other algorithms, enables a oherent presentation of local sub-deduction.

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 functions:

$$deg_A^-(v) = |\{w : w \in V \ \wedge \ (w, v) \in A\}|,$$

$$deg_A^+(v) = |\{w : w \in V \ \wedge \ (v, w) \in A\}|,$$

$$I_A(F) = |\{(i, j) : 1 \leq i, j \leq n : \wedge(F(i), F(j)) \in A\},$$

where v is an arbitrary vertex, and F is an n-length 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 (were F_i is i-sequence of the n_i-length).

1. The aim of of the first criterion is finding the pairs of sequences for which the following condition is fulfilled: $(F_1(n_1), F_2(1)) \in R$, and than it chooses the pairs for which the values of function

$$deg_R^+(F_1(n_1)), \left(\sum_{i=1}^{n_1} deg_R^+(F_1(i))\right) - I_R(F_1)$$

   are 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:

$$n_1 \cdot \left(\sum_{i=1}^{n_2} deg_R^-(F_2(i))\right) + n_2 \cdot \left(\sum_{i=1}^{n_1} deg_R^+(F_1(i))\right) - (n_1 + n_2) \cdot I_R(F_1^\frown F_2).$$

Apart from the main criteria there are several auxiliary criteria, 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 a significant 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 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%.

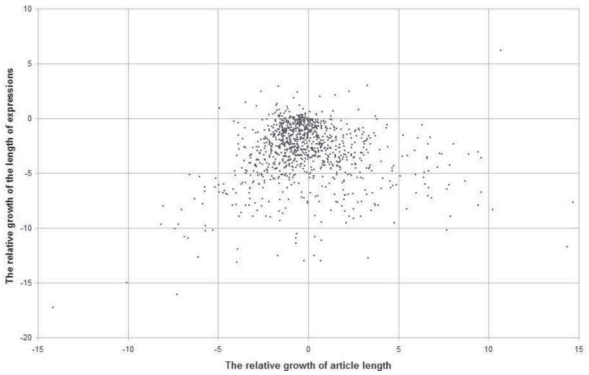


Fig. 1. The change of article length

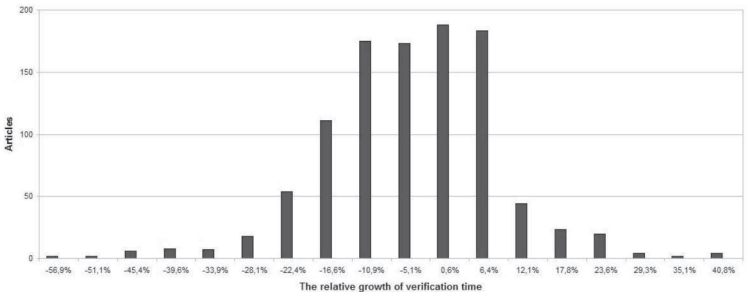


Fig. 2. The change of verification time

Final Remarks

Thanks to algorithms presented in our paper many advantages have been obtained. Firstly, our algorithms constitute one of the first (and the first in the Mizar system) fully automatic ways of standardizing the reasoning's structure that is 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 an important way and the particular steps of the deduction have became more readable for users of the database. In consequent, using the database has become easier. Thirdly, our algorithm of the proof's reorganization can be used independently to introducte of the results of different experiments which have not been able 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 than perseverance and even improvement of its legibility. Fourthly, an average time of the verification of articles of the database has been shortened. And last but not least, shortening of the number of steps of the deduction (about 3%) and the 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] F. B. Fitch. *Symbolic Logic: an Introduction.* The Ronald Press Co., New York, 1952.

[3] 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.

[4] B. Jorgen, G. Gregory, *Digraphs: Theory, Algorithms and Applications*, Springer, ISBN 1-85233-268-9, (2000).

[5] W. Marciszewski, *A Jaśkowski-Style System of Computer-Assisted Reasoning*, Philosophical Logic in Poland, Kluwer, 1993.

[6] R. Matuszewski, P. Rudnicki, *MIZAR: the first 30 years*, Mechanized Mathematics and Its Applications, 4 (**1**), pp. 3–24, 2005.

[7] R. Milewski, *Algorithms analyzing formal deduction support systems* – PhD thesis, The Computer Science Faculty of Białystok Technical University, Białystok 2008.

[8] R. Milewski, *New Auxiliary Software for MML Database Management Mechanized Mathematics and Its Applications*, ISSN 1345-8272 1345-8272, Vol. 5, No. 2: 1–10, 2006.

[9] R. Milewski, *Transformations of MML Database's Elements Lecture Notes in Computer Science*, Springer-Verlag, ISSN 0302-9743, Vol. 3863/2006: 376–388, 2006.

[10] *Mizar Home Page*, http://mizar.uwb.edu.pl/.

[11] L. C. Paulson, *The Isabelle Reference Manual*, 2000.

[12] 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.

[13] A. Trybulec, *Some features of the Mizar language*, Presented at a workshop in Turin, Italy, 1993.

[14] 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.

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