J Autom Reasoning (2013) 50:217–228
DOI 10.1007/s10817-012-9267-0

Methods of Lemma Extraction in Natural Deduction Proofs

Karol Pąk

Received: 6 December 2011 / Accepted: 10 October 2012 / Published online: 4 November 2012

Abstract The existing examples of natural deduction proofs, either declarative or procedural, indicate that often the legibility of proof scripts is of secondary importance to the authors. As soon as the computer accepts the proof script, many authors do not work on improving the parts that could be shortened and do not avoid repetitions of technical sub-deductions, which often could be replaced by a single lemma. This article presents selected properties of reasoning passages that may be used to determine if a reasoning passage can be extracted from a proof script, transformed into a lemma and replaced by a reference to the newly created lemma. Additionally, we present methods for improving the legibility of the reasoning that remains after the extraction of the lemmas.

Keywords Lemma · Extraction · Natural · Deduction · Legibility

1 Introduction

The databases of formalized mathematics are constantly being enlarged and are growing into considerable sizes because of adding more and more developments. Unfortunately, this enlargement of the databases is not always accompanied by improvement in the quality of formalization (see [4, 8, 12]). The readability is a subjective notion and a matter of individual taste, but the general idea that formal reasoning should be as similar as possible to informal mathematical proofs is unquestionable. However, an analysis of existing formalized proofs, especially long and more complex ones, leads to the conclusion that the readability of proof scripts might be very far from the general goal. Often, deductions that are correct for the verification system, might be very chaotic from the point of view of a human reader, and require a lot of effort to understand. Many proof authors seem to ignore this

K. Pąk (✉)
Institute of Informatics, University of Białystok, Białystok, Poland
e-mail: pakkarol@uwb.edu.pl

problem and hope that maybe in the future there will be automatic tools available for detecting and shortening reasoning passages, and also for eliminating repetitions of sub-deductions that could be replaced by a single auxiliary lemma.

Methods of improving proof readability based on finding reasoning passages which can be shortened and on better linearisation of argument were described in [7, 8] However, new problems appear while trying to specify formal properties of deduction passages, which, deserve to be called a lemma. In practice, the choice of passages to be extracted as auxiliary lemmas depends on the knowledge and experience of a user rather than on a formal specification. The ability to automatically find such passages that we call *packets*, is crucial e.g. for identifying the main idea of a proof based on lemma extraction from existing deductions (see [9]).

Two independent methods of lemma extraction can be considered, either based on *capsulation* of local deduction, or on generating cuts in the reasoning. Preliminary analysis of the notion of packet shows that the extracted reasoning packets must be disjoint and have a limited number of assumptions and conclusions. The subject of this article is to analyse an additional property and the method of introducing cuts in existing reasonings, which guarantee that the modified reasoning and the generated auxiliary lemma are correct for the verification system. The experiments are based on the notion of natural deduction proofs used in the Mizar system (see [2, 5, 10]).

2 The Selected Methods of Extracting Packets from Deductions

A common expectation of many authors who formalize mathematics using Mizar is that the readability of proofs would be improved if one could automatically divide the whole reasoning into a minimal number of passages, each of which presents an important step of the reasoning. This expectation stems from popular methods of improving the legibility of informal mathematical proofs, where complicated theorems are proved

– by highlighting main reasoning steps within the proof,
– or by using a series of auxiliary lemmas.

Therefore, highlighting the main idea of a proof based on the methods used for informal deductions seems to be a desirable direction for improving the legibility of formal proofs.

We can obtain the described properties in existing deduction in two ways.

1st method: “Encapsulation” of less important (often technical) passages of sub-deduction (packets) on the deeper level of a proof in the form of a nested lemma whose proof is generated on the basis of deduction steps extracted from the packet. The extracted packet is then replaced in the reasoning by a new step that describes the information carried by the packet.

2nd method: “Removal” of packets outside of the proof. The packets are replaced by new steps with references to the external lemmas that were generated from the removed packets.

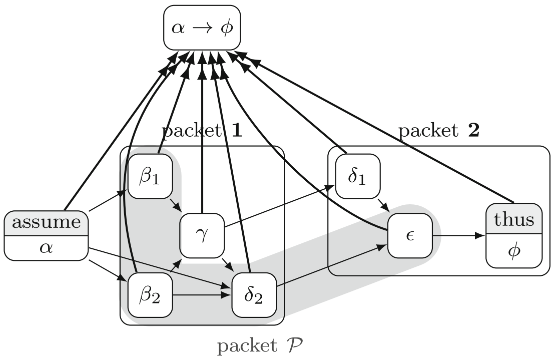
In both cases the statement associated with a new step of reasoning is a conjunction of statements that makes it possible to use the information included inside the packet outside its area. The necessity of introducing a new step results from the fact that together with the extracted packet all reference arcs leading from interior of the packet to outside its area must also be removed. The *reference arcs* represent the flow of information between a step—a statement (the head of the arc) and a previously justified step—a premise (the tail of the arc) used as the justification of that statement. To ensure the correctness of the modified reasoning, all premises of external steps originally contained in the packet are replaced by the reference to the new step. The steps within a packet that contain statements used outside the packet's area will be called *packet's conclusion*. Similarly, the steps outside the packet's area that contain statements used within the packet's steps will be called *packet's assumption*.

To illustrate how to modify the reasoning, we use an abstract model of proof. The abstract proof graph was introduced in [8]. Let us recall that the graph is a triple $\langle V, M, A\rangle$ such that

1. $\langle V, M\rangle$ is a forest of arborescences that represent dependencies between a step justified by a nested lemma (the head of the arc) and a step contained in that lemma (the tail of the arc),
2. $\langle V, A\rangle$ represents the flow of information contained in reference arcs, and e.g., dependence between steps which introduce variables into the reasoning and steps which use these variables in the expressions,
3. $\langle V, M \cup A\rangle$ is an acyclic digraph with the following properties: for each $\langle u, v\rangle \in A$ every direct successor of u is a predecessor of v in the forest $\langle V, M\rangle$.

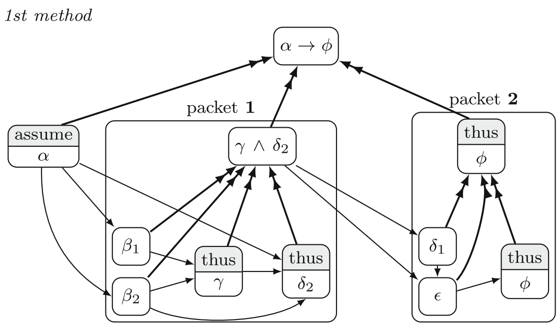
Let us consider the graph that represents a justification for an example statement of the form $\alpha \to \phi$ with two indicated packets (Fig. 1) and modifications of the proof graph that arise from the application of the two extraction methods described above (Fig. 2). The arrows $\twoheadrightarrow$ in the pictures represent elements of M and the solid arrows represent elements of A—reference arcs. The additional packet $\mathcal{P}$ is analysed in the further part of this article.

We apply the *1st method* here by capsulation of the packets 1 and 2 fragments of the reasoning. In this case, new intermediary formulas are added, $\gamma \wedge \delta_2$ and ϕ, that represent output of the capsules. When the *2nd method* is used, the arcs that lead to the packets are turned to explicit assumptions of the extracted lemmas, "*assume*

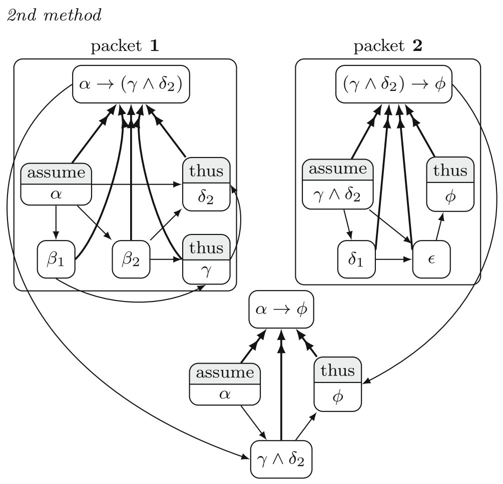


```
α → φ
proof
   assume A: α;
   B1: β₁ by A;
   B2: β₂ by A;
   C: γ by B1,B2;
   D1: δ₁ by C;
   D2: δ₂ by A,B2,C;
   E: ϵ by D1,D2;
   thus F: φ by E;
end;
```

Fig. 1 The abstract proof graph that represents a justification of an example statement of the form $\alpha \to \phi$ and a proof script written in the Mizar style



```
$\alpha \to \phi$
proof
  assume $A$: $\alpha$;
  $New1$: $\gamma \wedge \delta_2$
  proof
    $B1$: $\beta_1$ by $A$;
    $B2$: $\beta_2$ by $A$;
    thus $C$: $\gamma$ by $B1$,$B2$;
    thus $D2$: $\delta_2$ by $A$,$B2$,$C$;
  end;
  thus $F$: $\phi$
  proof
    $D1$: $\delta_1$ by $New1$;
    $E$: $\epsilon$ by $D1$,$New1$;
    thus $F$: $\phi$ by $E$;
  end;
end;
```



```
$Lemma1$: $\alpha \to (\gamma \wedge \delta_2)$
proof
  assume $A'$: $\alpha$;
  $B1$: $\beta_1$ by $A'$;
  $B2$: $\beta_2$ by $A'$;
  thus $C$: $\gamma$ by $B1$,$B2$;
  thus $D2$: $\delta_2$ by $A$,$B2$,$C$;
end;

$Lemma2$: $(\gamma \wedge \delta_2) \to \phi$
proof
  assume $C \wedge D2$: $\gamma \wedge \delta_2$;
  $D1$: $\delta_1$ by $C \wedge D2$;
  $E$: $\epsilon$ by $D1$,$C \wedge D2$;
  thus $F'$: $\phi$ by $E$;
end;

$\alpha \to \phi$
proof
  assume $A$: $\alpha$;
  $New1$: $\gamma \wedge \delta_2$ by $Lemma1$;
  thus $F$: $\phi$ by $New1$,$Lemma2$;
end;
```

Fig. 2 The modification of the abstract proof graphs (presented in the Fig. 1) and proof scripts written in the Mizar style that represent two extraction methods

α" and "*assume* $\gamma \wedge \delta_2$". As in the *1st method*, new intermediary formulas $\gamma \wedge \delta_2$, ϕ are also added in the second one, but in that case they represent conclusions of the extracted lemmas in the main reasoning.

The analysis of the modifications to the proof graph leads to the conclusion that the *1st method* of extraction provides a reader only with the information about the theses obtained within a packet. Finding the assumptions which are necessary to prove these theses requires that the reader examine all external references existing in the packet. If the number of assumptions is big—which may become apparent if the *2nd method* is attempted putting them in the statement does not seem to be a good solution. The application of the *2nd method* creates the statements associated with a lemma generated from the packet; the statement can generally be represented by the implication "assumptions" $\to$ "theses". However, introducing the explicit assumptions enlarges the proof by adding one vertex that introduces assumptions in the proof of the lemma ("*assume* α", "*assume* $\gamma \wedge \delta_2$").

The application of the *2nd method* can also be used to find and remove repeated passages of a sub-deduction by replacing several packets with references to a generated lemma. This method makes it possible to compare packets independently from the context of reasoning and chosen proof methods, e.g., by testing the equivalence of generated lemma statements, which allows further simplification of the proof structure.

In our analysis, the extracted packets coincided with coherent passages of reasoning. The chosen packets were disjoint and had the property of being closed with respect to directed paths, i.e., directed paths connecting two vertices belonging to the packet must be included in the packet. To show the correspondence between the property of closedness and the composition of the packet's statement, let us consider the packet $\mathcal{P}$ contained in the reasoning presented in Fig. 1 that contains vertices: β_1, β_2, δ_2, and ϵ. The packet bridges the end of the directed path $\beta_1 \rightarrow \gamma \rightarrow \delta_2$, which goes out of the area of $\mathcal{P}$.

Regardless of the chosen extraction method of the packet $\mathcal{P}$, the path generates a directed cycle that disallows a correct modification of the abstract proof graph. It yields from the fact that new vertices replacing the extracted packet $\mathcal{P}$ uses as its assumption the statement γ that is the conclusion of statement β_1, which is simultaneously the conclusion of the packet $\mathcal{P}$. Note that the application of the *2nd method* generates a lemma with the statement $(\alpha \wedge \gamma \wedge \delta_1) \rightarrow (\beta_1 \wedge \beta_2 \wedge \epsilon)$, which we can justify on the basis of the reasoning included in the packet $\mathcal{P}$. However, it is not possible to use this lemma correctly in the modified reasoning, because justification of one of the implication's assumptions (γ) requires using two conclusions of the implication (β_1, β_2).

Nevertheless, we can correctly extract the packet from the reasoning, because, in fact, the reasoning included in the packet is represented by two implications $(\alpha \rightarrow (\beta_1 \wedge \beta_2)) \wedge ((\alpha \wedge \gamma \wedge \delta_1) \rightarrow \epsilon)$. Generally, the reasoning included in a packet can be represented by a conjunction of implications, where the consequent of a given implication is one of the packet's conclusions $\mathfrak{t}$, and the antecedent is a conjunction of the packet's assumptions that are predecessors of $\mathfrak{t}$ in the abstract proof graph. In the sequel we will call this conjunction of implications *the basic packet's statement.* Naturally, any formula that is equivalent to the basic packet's statement (e.g. $\alpha \rightarrow (\beta_1 \wedge \beta_2 \wedge ((\gamma \wedge \delta_1) \rightarrow \epsilon)))$ can be the statement that represents reasoning included in the packet.

The statement of the modified reasoning that replaces the extracted packet has a *modified basic packet's statement*. In this statement these packet's assumptions that are predecessors of all the packet's conclusions (e.g. α in the packet $\mathcal{P}$, Fig. 3) are omitted.

Extraction of Non-Closed Packets The extraction of the packets that are not closed with respect to directed paths in an abstract proof graph requires a more advanced modification of the graph. The reasoning justifying the packet's statement must be constructed from a series of independent and separate proofs that justify every single basic implication. Such a solution is generally accompanied by duplicating the vertices from the packet, because the proof of each implication contains all vertices of the packet that in the proof graph are simultaneously the successor of that implication's antecedent and the predecessor of that implication's consequent. Naturally, a proper selection of the statement that will be equivalent to the

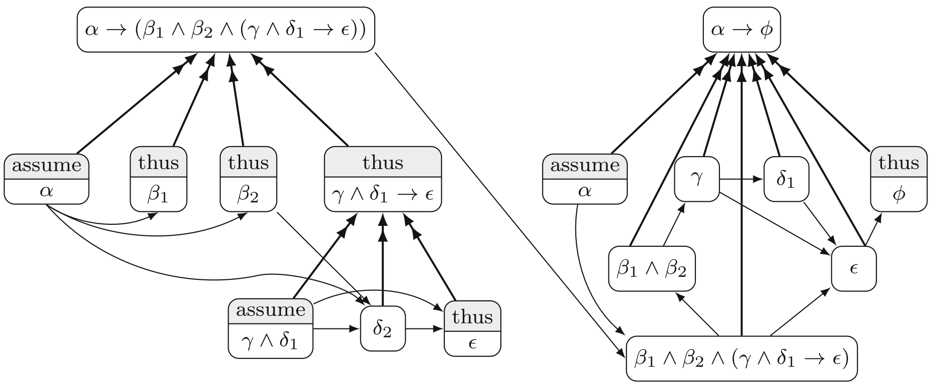

$\alpha \to (\beta_1 \wedge \beta_2 \wedge (\gamma \wedge \delta_1 \to \epsilon))$
assume α
thus β_1
thus β_2
thus $\gamma \wedge \delta_1 \to \epsilon$
assume $\gamma \wedge \delta_1$
δ_2
thus ϵ
$\alpha \to \phi$
assume α
γ
δ_1
thus ϕ
$\beta_1 \wedge \beta_2$
ϵ
$\beta_1 \wedge \beta_2 \wedge (\gamma \wedge \delta_1 \to \epsilon)$

Fig. 3 The modification of the abstract proof graphs (presented in Fig. 1) which represent the extraction of the packet $\mathcal{P}$

conjunction of implications, where the implications that have identical assumptions are conjoined (using the law of conjoining antecedent in conjunction), and where common antecedents of the implications were extracted (using the law of exportation and the law of conjoining consequent in conjunction) enables minimization of the number of duplicated vertices.

The modification of the reasoning part that is left after the extraction of such a packet is more complicated. To each implication included in the basic packet's statement

$$\left(assumption_1 \wedge assumption_2 \wedge \ldots \wedge assumption_n\right) \to statement, \tag{1}$$

a new reasoning step is assigned:

$$\begin{aligned} &statement \ \texttt{by} \ reference\ to\ assumption_1\texttt{,}\\ &\qquad reference\ to\ assumption_2\texttt{,}\ \ldots\texttt{,}\ reference\ to\ assumption_n\texttt{,}\\ &\qquad reference\ to\ the\ step\ which\ replaces\ the\ packet\texttt{;} \end{aligned} \tag{2}$$

The references that before the modification indicated the packet's conclusion, in the modified reasoning indicate the respective reasoning steps obtained using the basic implications. The modification of the abstract proof graph (Fig. 1) containing the packet $\mathcal{P}$ is presented in Fig. 3.

Naturally, steps generated of the form 2 are sufficient to preserve reasoning correctness, but some of them could be removed by an appropriate modification of reasoning. In Fig. 3, the arc from $\beta_1 \wedge \beta_2$ to γ could be replaced by an arc from $\beta_1 \wedge \beta_2 \wedge (\gamma \wedge \delta_1 \to \epsilon)$ to γ, without affecting the reasoning correctness in the Mizar system, and then the step $\beta_1 \wedge \beta_2$ could be removed, as a step that is not used in the reasoning. We can solve this problem with auxiliary tools like RELINFER, INACC (see [6, 7, 11]), RENINFER (see [8]) available in the Mizar system. These tools one can first detect the references (which indicate steps of the form 2) that can be replaced by all references used to justify these steps. Then one can remove all deduction steps which in consequence are not used in any justification.

The presented group of modifications that consist of partitioning of reasoning into passages included in the non-closed packet (where every single passage justifies the implications from basic packet's statement), does not raise doubts about properties of an abstract proof graph in the modified proof graph (see [8]). In particular, the property of an acyclic character of this graph is presented (see [3]). However, modifications of the part that remains after the packet extraction from the reasoning are not so intuitive.

Theorem 1 *The modification of an abstract proof graph resulting from the extraction of a packet that is not closed with respect to directed paths, does not generate a directed cycle in this proof graph.*

Proof Let us consider an abstract proof graph $\mathfrak{P}$ and its modification $\widetilde{\mathfrak{P}}$. Suppose, contrary to our claim, that there exists a directed cycle $\mathfrak{a}$ of $\widetilde{\mathfrak{P}}$. By definition, every abstract proof graphs is acyclic, hence $\mathfrak{a}$ has to cross new vertices introduced during modifications. To obtain a contradiction, we show that it is possible to replace every maximal subpath of $\mathfrak{a}$, which is given by new vertices, by a path of $\mathfrak{P}$ in such a way that the modified cycle $\mathfrak{a}$ will be a cycle of $\mathfrak{P}$.

Let us take a subpath $\mathfrak{a}' = a_0 \to a_1 \to a_2 \to \ldots \to a_k \to a_{k+1}$ of $\mathfrak{a}$ such that a_i is a new vertex of $\widetilde{\mathfrak{P}}$ for $i = 1, 2, \ldots, k$ and a_0, a_{k+1} are vertices of $\mathfrak{P}$. It is easy to check that only two cases are possible:

- $k = 1$, a_0 is an assumption $\mathfrak{a}$ of the packet; a_1 is the statement $\mathfrak{s}'$ which is a consequent of the selected basic implication which uses $\mathfrak{a}$ in justification; a_2 is a statement which uses $\mathfrak{s}'$ in justification,
- $k = 2$, a_0 is an assumption of the packet, which is a predecessor of all packet's conclusion; a_1 is the new step which replaces packet in reasoning; a_2 is the statement $\mathfrak{s}$ which is a consequent of the selected basis implication; a_3 is a statement which uses $\mathfrak{s}$ in justification.

In both cases, $\mathfrak{a}'$ describes a relation between the packet's assumption and a conclusion of the packet's conclusion proved, among others, on the basis of this packet's assumption. However, from the construction of the basic packet's statement it results that in $\mathfrak{P}$ the packet's assumption must have been the predecessor of the packet's conclusion, so simultaneously it must have been the predecessor of this conclusion. Consequently, each directed sequence of the new vertices in cycle $\mathfrak{c}$ of $\widetilde{\mathfrak{P}}$ can be replaced by a corresponding directed sequence of vertices of $\mathfrak{P}$. The sequence formed in that way contradicts the assumption of the acyclic character of $\mathfrak{P}$. □

The process of extracting packets that are not closed with respect to directed paths does not lead to errors in the modified reasoning. Extraction of packets whose basic packet's statements consists of too many implications does not improve the legibility, and even makes the understanding of the proof's concept more difficult.

3 The Methods of Packet's Extraction which Consider the Ordering Arc

So far in this article we have consciously ignored the information included in the abstract proof graph resulting from using variables in the reasoning. Intuitively, the use of variables in the packet should cause only appearance of the universal quantifiers in the beginning of the packet's statement, binding these variables.

Now, let us focus on dependencies induced by the variables. Note that these dependencies can influence the closedness with respect to directed paths used in the process of packet extraction. Therefore, it is necessary to examine the existence of a directed path in the abstract proof graph that can be constructed from the reference arcs and the ordering arcs. Let us remind that the dependencies between the introduction of the variable and its use in an abstract proof graph are called the *ordering arcs* (see [8]).

The process of packet extraction from the reasoning is not modified significantly if the packets fulfil the following condition: if new variables are introduced to the packet's reasoning, then these variables are used only within this packet.

The application of the *2nd method* presented in the previous section now generates analogous packet's statement preceded by an extra universal quantifier that binds variables used in the packet's statement. Additionally, this universal quantifier creates the need to extend the proof from the packet by adding a new step that introduces these bound variables to the reasoning.

The *1st method* of hiding packets on the deeper nesting level is very similar to the method which does not consider the variables in the reasoning. The differences appear if the extracted packet $\mathcal{P}$ has an outgoing directed path that contains the vertices introducing to the reasoning the variable used in the packet's statement. More precisely, there is a directed path from $x \in \mathcal{P}$, to another $y \in \mathcal{P}$ that goes through a vertex $z \notin \mathcal{P}$ and z is the tail of some ordering arc, whose head is the assumption or the conclusion of the packet $\mathcal{P}$.

In that case, the statement used in the new step replacing the extracted packet, has to be equivalent to the modified basic packet's statement and additionally has to contain all universal quantifiers that bind the variables, introduced in the vertices (as z) and used in the statement of the packet $\mathcal{P}$.

The ordering of all introduced universal quantifiers in the packet's statement, regardless of the chosen extraction method has to coincide with the topological sorting of verities of the proof graph's sub-graph (see [3]) that includes all vertices introducing variables that are bound by the universal quantifiers which are in the packet's statement. The arc $\langle x, y\rangle$ is an arc of this sub-graph if and only if $x \neq y$ and there is a directed path from x to y in the proof graph, where the directed path contains only the ordering arcs. Moreover, the introduced universal quantifiers must be distributed in a way that they precede only these conjuncts of the packet's statement which use the variables bound by these quantifiers. This is possible since conjunction is distributive over universal quantifiers and then the property that enables the elimination of quantifiers that involve a variable not occurring within the scope of a quantification for that variable.

The case when a variable created outside of the area of the packet is used in the packet's reasoning but does not occur in any assumption or any conclusion is noteworthy. In that case only the properties of this variable's type are used and its every occurrence can be replaced e.g. by "the global choice" of variable of this type (see [2]).

The extraction of packets containing the reasoning that has the introduced variables used outside of the packet, causes much more problems connected with the reasoning modifications that result from the packet's extraction.

To simplify further analysis we assume that the extracted packets do not contain the skeleton steps (see [2]) that introduce the universal and existential quantifiers and the implications. To this aim we also assume that the packet does not contain the steps that introduce local functors and local predicates. It also seems reasonable to expand in the reasoning all abbreviations of terms and variables, defined in the skeleton steps which introduce the existential quantifiers to the reasoning.

To describe how the packet's statement is created, let us set the family of the packet's steps introducing to the reasoning the variables used from the packet as $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$; and the family of steps that do not belong to the packet but introduce to the reasoning the variables used within this packet as $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$. Let us also set the sub-graf $\mathfrak{G}$ of the proof graph that contains the steps of families $\mathcal{S}$, $\mathcal{R}$, and fulfils the following condition: the arc $\langle x, y\rangle$ is the arc of this sub-graph if and only if $x \neq y$ and there is the directed path from x to y in proof graph that contains only the ordering arcs.

Let us define two recursive families of vertices $\{\mathcal{E}_i\}_{i=1}^{\infty}$, $\{\mathcal{A}_i\}_{i=1}^{\infty}$, where $\mathcal{E}_i \subseteq \mathcal{S}$, $\mathcal{A}_i \subseteq \mathcal{R}$, for each $i = 1, 2, \ldots$. Mark $\mathcal{A}_1$ as the family of these vertices with $\mathcal{R}$, such that no vertex in $\mathcal{A}_1$ is a successor of a vertex in $\mathcal{S}$.

The vertex v is an element of family $\mathcal{E}_i$ ($\mathcal{A}_{i+1}$) if and only if

- $v \in \mathcal{S}$ ($v \in \mathcal{R}$),
- there is the directed path that leads from some vertex that belongs to $\mathcal{A}_i$ ($\mathcal{E}_i$) to v, and this path is given by concatenating two sequences: the sequence of the vertices that belong to $\mathcal{A}_i$ ($\mathcal{E}_i$) and the sequence of the vertices that belong to $\mathcal{S}$ ($\mathcal{R}$),
- and every directed path that leads from some vertices that belongs to $\mathcal{A}_i$ ($\mathcal{E}_i$) to v, is given by concatenating two sequences: the sequence of the vertices that belong to $\mathcal{A}_i$ ($\mathcal{E}_i$) and the sequence of the vertices that belong to $\mathcal{S}$ ($\mathcal{R}$),

where $i = 1, 2, \ldots$.

Let us fix the ordering of the variables that are introduced by steps from the families $\mathcal{A}_i$, $\mathcal{E}_i$, similarly as we ordered all variables bound by the universal quantifiers using the topological sorting for the packet's statement. Let us set respectively $\mathfrak{a}_i$, $\mathfrak{e}_i$ for these ordering, where $i = 1, 2, \ldots$. Then we get the statement's formulation from the passages of formulas obtained by the successive steps (written in Mizar style):

1° `for` *the ordered variables* $\mathfrak{a}_1$ `holds` *the conjunction of the basic packet's statement that use only variables belong to* $\mathfrak{a}_1$

2° `&` *the packet's assumptions necessary to justify the existence of the variables that belong to* $\mathfrak{e}_1$ *(called* $A2°$*)* `implies ex` *the ordered variables* e_1 `st` *conjunction of the basic packet's statement that use only variables which belong to* a_1 *and* e_1 *but were not indicated in* 1°. *Moreover, in the antecedent of each implication of the basic packet's statement, the packet's assumptions* $A2°$ *were omitted*

3° `& for` *the ordered variables* $\mathfrak{a}_2$ `holds` *conjunction of the basic packet's statement that use only variables which belong to* $\mathfrak{a}_1$, $\mathfrak{e}_1$, *and* $\mathfrak{a}_2$ *that were not indicated in* 1° *and* 2°. *Moreover, similarly as in* $A2°$, *in the antecedent of each implication of the basic packet's statement, the packet's assumptions* $A2°$ *were omitted*

4° `&` *the packet's statement necessary to justify the existence of the variables that belong to* $\mathfrak{e}_2$, *where the assumption of packet* $A2°$ *were omitted (called* $A4°$*)* `implies ex` *the ordered variables* $\mathfrak{e}_2$ `st` *conjunction of the basic packet's statement that use only variables which belong to* $\mathfrak{a}_1$, $\mathfrak{e}_1$, $\mathfrak{a}_2$, *and* $\mathfrak{e}_2$ *but were not* *indicated in* 1°, 2° *and* 3°. *Moreover, in the antecedent of each implication of the basic packet's statement, the packet's assumptions* $A2°$, $A4°$ *were omitted*

⋮

The presented packet's statement has the necessary information about the reasoning contained in the packet that suffices to provide the correctness of the reasoning part remaining after the packet's extraction. Let us note that the packet's statement can be generalized, e.g., if in the packet the variable x of type Θ was constructed, such that $P(x)$ holds, and there is the justified packet's conclusion $Q(x)$, then the information about this reasoning is formulated as: `ex` x `be` Θ `st` Q`(`x`)`, despite the possibility of justification of a more general fact: `( ex` x `be` Θ `st` Q`(`x`)) & (for` x `be` Θ `st` P`(`x`)` `holds` Q`(`x`))`. Such generalization is not necessary to provide the correctness of the modified reasoning.

The extraction of such packets causes many modifications of the reasoning. In this article, we will only say that using this packet's statement requires adding to the reasoning new steps which introduce the ordering variables $\mathfrak{e}_1, \mathfrak{e}_2, \ldots$. The steps have the following form:

`consider`*ordered variables* $\mathfrak{e}_i$ `such that`
the further part of the packet's statement that occurs after "st" in the step $2i°$
`by` *the references to the packet's assumptions* $A2i°$`,` *the reference to the step*
introducing the variables $\mathfrak{e}_{i-1}$ *(*$i > 1$*) or the extracted packet (*$i = 1$*)*`;` (3)

where $i = 1, 2, \ldots$. For a fixed i, this step in the proof graph must be the successor of the packet's assumptions $A2i°$ and of the steps that belong to $\mathcal{A}_i$.

4 The Methods of Eliminating Hidden Information

It should be noted that the ordering arcs which contain information about the dependencies between the variable's introduction and its use, can also describe the reasoning's dependence, which intuitively corresponds to the reference arcs. Let us analyze the reasoning passage written in the Mizar language (Fig. 4) to present the sort of information that can be transmitted by the ordering arcs in an abstract proof graph. The statement f `is` *one-to-one* `&` f `is` *onto* contained in this reasoning does not require using label $A1$ for justification. The information about the function f's properties in the Mizar system results from the fact that label $A2$ is used in the justification and points to the statement that uses the identifier P (the statement P `is` *Function-like* can be replaced by the trivial $P = P$).

Finding the ordering arcs in an abstract proof graph that contains information about these "hidden reference arcs" only on the basis of the proof graph is not possible. Thus, it is also not possible to determine the set of all of the packet's assumptions on the basis of the reference arcs. Naturally, the probability of encountering "hidden

Fig. 4 The reasoning passage presenting the hidden reference arcs

```
let X be non empty set ;
let f be Function of X,X ;
assume A1: f is one-to-one onto ;
reconsider P=f as Permutation of X by A1, FUNCT_2: def 4 ;
A2: P is Function-like ;
A3: f is one-to-one & f is onto by A2 ;
```

reference arcs" during the extraction of the packets that are not closed with respect to directed paths is incomparably greater than with the packets that possess this property.

To avoid problems with "hidden reference arcs", the modification which removes the Mizar construction `reconsider` from the proof is necessary. Generally, this modification requires two steps. The first step should guarantee that the construction `reconsider` does not overwrite variable identifiers. Thus, in this step we replace:

$$\begin{aligned}&\texttt{reconsider}\ \{\mathit{Equating\text{-}List}\texttt{,}\}\ \mathit{Variable\text{-}Identifier}\ \{\texttt{,}\ \mathit{Equating\text{-}List}\}\\&\quad\texttt{as}\ \mathit{Type\text{-}Expression}\ \mathit{Simple\text{-}Justification}\ \texttt{;}\end{aligned}\tag{4}$$

by

$$\begin{aligned}&\texttt{reconsider}\ \{\mathit{Equating\text{-}List}\texttt{,}\}\ \mathit{New\text{-}Variable\text{-}Identifier} = \mathit{Variable\text{-}Identifier}\\&\quad\{\texttt{,}\ \mathit{Equating\text{-}List}\}\ \texttt{as}\ \mathit{Type\text{-}Expression}\ \mathit{Simple\text{-}Justification}\texttt{;}\end{aligned}\tag{5}$$

and we replace all occurrences of overwritten variables respectively. In the second step we replace:

$$\begin{aligned}&\texttt{reconsider}\ \mathit{Variable\text{-}Identifier} = \mathit{Term\text{-}Expression}\\&\quad\{\texttt{,}\ \mathit{Variable\text{-}Identifier} = \mathit{Term\text{-}Expression}\}\\&\quad\texttt{as}\ \mathit{Type\text{-}Expression}\ \mathit{Simple\text{-}Justification}\ \texttt{;}\end{aligned}\tag{6}$$

by

$$\begin{aligned}&\texttt{consider}\ \mathit{Variable\text{-}Identifier}\ \{\texttt{,}\ \mathit{Variable\text{-}Identifier}\}\ \texttt{be}\ \mathit{Type\text{-}Expression}\\&\quad\texttt{such that}\ \mathit{New\text{-}Label\text{-}Identifier} : \mathit{Variable\text{-}Identifier} = \mathit{Term\text{-}Expression}\\&\qquad\qquad\{\ \texttt{\&}\ \mathit{Variable\text{-}Identifier} = \mathit{Term\text{-}Expression}\}\\&\quad\mathit{Simple\text{-}Justification}\texttt{;}\end{aligned}\tag{7}$$

and write respectively the label (which points to *New-Label-Identifier*) in the justification of every step that uses the variables introduced with the construction `consider` (see [1, 2]).

To check which of the additional labels have to be added, we can use the auxiliary application RELPREM (see [6, 11]).

The rest of the constructions contained in the Mizar language, e.g. the construction `set` that introduces the abbreviations of more complicated terms or the construction `deffunc` that introduces to the reasoning local functors, does not conceal reference arcs as the ordering arcs.

5 Conclusions

The methods of extracting reasoning passages presented in this article which preserve proof correctness showed that the property of packets closure with respect to directed paths is an important packet feature. We have also shown that this property, although intuitively it does not seem to be associated with the statement's structure describing the packet's reasoning, can cause a significant enlargement of this statement and the modified proof. Moreover, we have indicated that the obvious criteria based on restricting the number of the packet's assumptions and conclusions do not reflect the complexity of detecting such packet's as it happens in the case of the closure property.

There is no doubt that the presented methods of extraction can be used to solve the main problems arising during the extraction process of any packets, independently from the criteria imposed on the chosen packets. Further work should concern the search of methods to enable automatic selection of packets whose extraction ensures the improvement of legibility of the modified proof. However, due to the complexity of this problem, it constitutes a very ambitious challenge.

References

1. Bonarska, E.: An Introduction to PC Mizar. Mizar Users Group. Fondation Philippe le Hodey, Brussels (1990)
2. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a nutshell. J. Formal. Reason. **3**(2), 153–245 (2010)
3. Jorgen, B., Gregory, G.: Digraphs: Theory, Algorithms and Applications. Springer. ISBN 1-85233-268-9 (2000)
4. Kornilowicz, A.: Tentative experiments with ellipsis in Mizar. In: AISC/MKM/Calculemus, pp. 453–457 (2012)
5. Matuszewski, R., Rudnicki, P.: Mizar: the first 30 years. Mech. Math. Its Appl. **4**(1), 3–24 (2005)
6. Milewski, R.: New auxiliary software for MML database management. Mech. Math. Its Appl. **5**(2), 1–10 (2006)
7. Milewski, R.: Algorithms analyzing formal deduction support systems/Algorytmy analizy systemu wspomagania edukcji formalnej (in Polish). PhD thesis, Faculty of Computer Science, Białystok University of Technology (2008)
8. Pąk. K.: The algorithms for improving and reorganizing natural deduction proofs. Stud. Log. Gramm. Rheto. **22**(35), 95–112 (2010). ISBN 978-83-7431-273-8, ISSN 0860-150X
9. Rahul, S.P., Necula, G.C.: Proof optimization using lemma extraction. UCB/CSD-01-1143, Computer Science Division (EECS), University of California (2001)
10. Rudnicki, P.: An overview of the Mizar project. In: Proceedings of the 1992 Workshop on Types for Proofs and Programs, Chalmers University of Technology. Bastad, pp. 311–332 (1992)
11. Rudnicki, P., Trybulec, A.: On the integrity of a repository of formalized mathematics. In: Proceedings of MKM 2003, Lecture Notes in Computer Science, vol. 2594 (2003)
12. Urban, J.: Xml-izing Mizar: Making semantic processing and presentation of MML easy. In: Proceedings of MKM 2005, pp. 346–360 (2005)