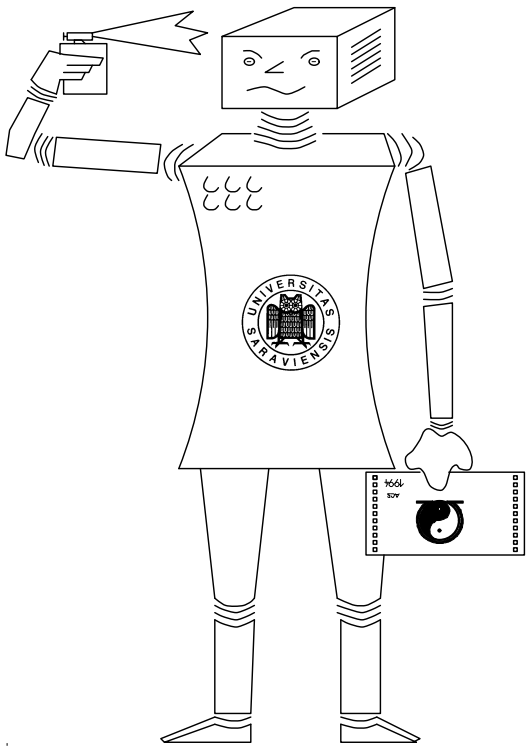
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



Reconstructing Proofs at the Assertion Level

Xiaorong Huang

Published as: In Alan Bundy (ed.), *Proceedings of CADE-94*, pp 738–752, LNAI-814, Springer, 1994

Reconstructing Proofs at the Assertion Level

Xiaorong Huang

Fachbereich Informatik, Universität des Saarlandes
Postfach 15 11 50, D-66041 Saarbrücken, Germany
huang@cs.uni-sb.de

Abstract. Most automated theorem provers suffer from the problem that they can produce proofs only in formalisms difficult to understand even for experienced mathematicians. Effort has been made to *reconstruct* natural deduction (ND) proofs from such machine generated proofs. Although the single steps in ND proofs are easy to understand, the entire proof is usually at a low level of abstraction, containing too many tedious steps. To obtain proofs similar to those found in mathematical textbooks, we propose a new formalism, called ND style proofs at the *assertion level*, where derivations are mostly justified by the application of a definition or a theorem. After characterizing the structure of compound ND proof segments allowing assertion level justification, we show that the same derivations can be achieved by domain-specific inference rules as well. Furthermore, these rules can be represented compactly in a tree structure. Finally, we describe a system called *PROVERB*, which substantially shortens ND proofs by *abstracting* them to the assertion level and then transforms them into natural language.

1 Introduction

This paper concerns the presentation of machine generated proofs. Viewing automated theorem provers as a special sort of expert systems, this problem is very similar to that of the explanation component of an expert system. In order to aid the understanding of an end-user, methods are devised to augment, to prune, or even to transform the trace of reasoning left behind by an expert system [Sho76, WS89]. Explanations produced in this way are in general *tightly bound* with the authentic movement of an expert system from the initial data to the conclusion. Although such explanations are apparently appropriate for system developers or knowledge engineers, they do not meet the requirement of a typical end-user. To solve this problem, a new, so called *reconstructive* paradigm for explanation has emerged in recent years [WT92]. The central idea of this approach is that a distinct knowledge base should be used to reconstruct a new solution based on the original one found by the expert system.

The *reconstructive* approach for explanation has been pursued in the field of automated reasoning as well, because not only the line of reasoning can be unnatural and obscure, the formalism in which the proofs are encoded is usually extremely machine oriented. Procedures have been developed to transform proofs from machine oriented formalisms into more natural formalisms [And80, Mil83, Pfe87, Lin90]. As the target formalism, usually a variation of the *natural deduction* (ND) proof first proposed by G. Gentzen [Gen35] is chosen. Heuristics of various kinds are developed

to improve the quality of the target ND proof. For instance, C. Lingenfelder utilizes the topological structures of the refutation graph both to produce more direct proofs as well as to avoid redundancy by inserting lemmas [Lin90]. Another technique for inserting lemmas is reported in [PN90].

Until now the reconstruction stops here and ND proofs are used as inputs by systems producing proofs in natural language. The first such attempt was made by D. Chester [Che76]. His system EXPOUND is usually characterized as an example of *direct translation*. Although a sophisticated linearization is applied on the input ND proofs, the steps are translated locally in a template driven way. Equipped with more advanced techniques developed in the field of natural language generation, a more coherent translation was obtained by the MUMBLE system of D. McDonald [McD83], where emphasis was laid on the generation of utterances highlighting important global structures of the proofs, as well as utterances mediating between subproofs. A more recent attempt can be found in THINKER [EP93], where different styles for explaining ND proofs are exploited. In short, it was believed that ND proofs can be adequately presented by resorting solely to *ordering*, *pruning*, and *augmentation*.

All these systems suffer from the same problem: The derivations they convey are exclusively at the level of the inference rules of the ND calculus. In contrast to informal proofs found in standard mathematical textbooks, such proofs are composed of derivations familiar from elementary logic, where the focus of attention is on syntactic manipulations rather than on the underlying semantic ideas. The main problem, we believe, lies on the lack of intermediate structures of ND proofs, which allow atomic justifications at a higher level of abstraction.

To gain more reliable experience with the *levels of justifications*, we have analyzed proofs in mathematical textbooks like [Deu71]. Based on our preliminary empirical study, justifications are provided at three levels.

- *Logic level* justifications are simply verbalizations of the ND inference rules, such as the rule of Modus Ponens.
- *Assertion level* justifications account for a derivations in terms of the application of an axiom, a definition or a theorem (collectively called an assertion). The following is an example:

  "since a is an element of the set S_1, and S_1 is a subset of S_2, *according to the definition of subset*, a is an element of S_2".
- *Proof level* justifications are at a still higher level and are comparatively rare. One example is justifying a proof segment as a whole by resorting to its similarity to a previous proof segment.

Among the three levels mentioned above, the assertion level plays a dual role in presentation. On the one hand assertion level justifications are *logically compound*, that is, mathematicians can explain such steps by providing a logic level proof segment. On the other hand, assertion level justifications are primitive with respect to presentation, since proof segments justifiable atomically at the assertion level is practically never expanded to a logic level proof segment. On account of this, while proof level structures are also very useful, the reconstruction of assertion level units in ND proofs is of paramount importance and is indispensable for the purpose of presenting proofs in a natural way.

Section 2 first defines the structure of the logic level proof segments which can be justified atomically at the assertion level. Section 3 accounts for the acquisition of domain-specific assertion level inference rules and shows how they can be organized in a tree structure. Then in section 4, we illustrate how this tree structure can be used to abstract ND proofs to the assertion level and report our experience with them in the subsequent translation into natural language. Finally, a look into the future work concludes this paper.

2 Compound Proof Segment at the Assertion Level

The existence of a hierarchy of proof units in proofs constructed by mathematicians can be accounted for by a computational model of human deductive reasoning [Hua93]. Following A. Bundy [Bun88], this theory cast theorem proving as a planing process, where a planner constructs a proof by applying *methods* (called tactics in some earlier systems [GMW79, CAB$^+$86]) on open goals. The proof under construction is represented as a hierarchical and partially elaborated plan called a *proof tree*. The execution of each method results in the integration of a subtree constituting a proof unit with internal structure. In the light of this, the intuitive notion of the application of an assertion is technically realized either by a compound proof unit composed of applications of ND rules, or by a atomic proof unit justified by a *domain-specific* inference rule.

Fig. 1 is an example of a compound proof unit inferring $a_1 \in F_1$ from $U_1 \subset F_1$ and $a_1 \in U_1$ by applying the definition of subset encoded as

$$\forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow \forall_x x \in S_1 \Rightarrow x \in S_2 \tag{1}$$

The leaf with the label $\mathcal{A}$ contains the *assertion* being applied.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}:\ \ \forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow (\forall_x x \in S_1 \Rightarrow x \in S_2)}{U_1 \subset F_1 \Leftrightarrow (\forall_x x \in U_1 \Rightarrow x \in F_1)}\forall D}{U_1 \subset F_1 \Rightarrow (\forall_x x \in U_1 \Rightarrow x \in F_1)}\Leftrightarrow D,\ \ U_1 \subset F_1}{\forall_x x \in U_1 \Rightarrow x \in F_1}\Rightarrow D}{a_1 \in U_1 \Rightarrow a_1 \in F_1}\forall D,\ \ a_1 \in U_1}{a_1 \in F_1}\Rightarrow D$$

Fig. 1. Natural Expansion 1 for Subset Definition

Actually, the procedure applying assertions by constructing a compound proof segment is specified in terms of a so called *decomposition-composition* constraint imposed on such proof segments identified in our preliminary empirical study [Hua92]. The following two definitions are necessary for the discussion of this constraint.

Definition: An inference rule of the form $\frac{\triangle\vdash F, \triangle\vdash P_1, \ldots, \triangle\vdash P_n}{\triangle\vdash Q}$ is a *decomposition* rule with respect to the formula schema F, if all applications of it, written as $\frac{\triangle\vdash F', \triangle\vdash P'_1, \ldots, \triangle\vdash P'_n}{\triangle\vdash Q'}$ satisfy the following condition: each $P'_1, \ldots, P'_n$ and Q' is

– a proper subformula of F', or
– a specialization of F' or of one of its proper subformula, or
– a negation of one of the first two cases.

Under this definition, $\wedge D, \Rightarrow D, \forall D$ are the only elementary decomposition rules in the natural deduction calculus $\mathcal{NK}$. Compare Fig. 1 for the meaning of the rules.

Definition: An inference rule of the form $\frac{\triangle \vdash P_1, \ldots, \triangle \vdash P_n}{\triangle \vdash Q}$ is called a *composition* rule if all applications of it, written as $\frac{\triangle \vdash P'_1, \ldots, \triangle \vdash P'_n}{\triangle \vdash Q'}$, satisfy the following condition: each $P'_1, \ldots P'_n$ is

– a proper subformula of Q', or
– a specialization of Q' or of one of its proper subformula, or
– a negation of one of the first two cases.

As illustrated in Fig. 1, the decomposition-composition constraint requires that a logic level proof segment applying an assertion $\mathcal{A}$ consists of a linear decomposition of $\mathcal{A}$ along the branch from $\mathcal{A}$ to the root. Other premises needed in the series of decompositions (the leaves $U_1 \subset F_1$ and $a_1 \in U_1$ in Fig. 1) can be obtained by compositions. For an example of such composition, see Fig. 2. For a precise definition of this constraint, the readers are referred to [Hua92]. In the sequel, proof segments satisfying this constraint will be referred to as the *natural expansion* of corresponding assertion level justification. This constraint is closely related to one of Johnson-Laird's *effective procedures* [JL83], aimed at accounting for spontaneous daily reasoning. Unfortunately, the psychological explanations provided by him can not be extended to predicate logic straightforwardly.

3 Assertion Level Inference Rules

In this section, we show that deductions justifiable by the application of a particular assertion $\mathcal{A}$ can be covered by a finite set of *domain-specific* inference rules at the assertion level. In the sequel, we denote this set of rules applying an assertion $\mathcal{A}$ by $Rules(\mathcal{A})$. It is this finiteness that makes this concept useful both for proof presentation, as well as for interactive proof development environments [HKK+93].

3.1 Acquisition of Assertion Level Inference Rules

There are two ways for acquiring new assertion level rules:

– learning by *chunking-and-variablization*,
– learning by *contraposition*.

Chunking-and-Variablization First, since there is evidence that input-output patterns of repeated actions will be remembered as new operators, we believe that patterns of repeated applications of an assertion may be remembered as new rules. Similar phenomena is called in other systems the learning of *macro-operators* [FHN72],

or *chunking* [New90]. On account of this, domain-specific rules are also referred to as compound rules or macro-rules. We continue with our subset example to illustrate this.

Example 1 (Continued): Suppose that a reasoner has just derived $a_1 \in F_1$ from the premises $a_1 \in U_1$ and $U_1 \subset F_1$ by applying the definition of subset (1). Our assumption is that apart from merely drawing a concrete conclusion from the premises, possibly he learns the following macro-rule as well:

$$\frac{\triangle \vdash a \in U, \triangle \vdash U \subset F}{\triangle \vdash a \in F} \tag{2}$$

where a, U and F are metavariables standing for object variables. More generally, hand in hand with deductive steps corresponding to the natural expansions with $P'_1, ..., P'_m$ as the leaves and P' as the root, the inference rule below may be acquired:

$$\frac{\triangle \vdash P_1, \ldots, \triangle \vdash P_m}{\triangle \vdash P} \tag{3}$$

where $P_1, \ldots, P_n$ are formula schemata generalized from $P'_1, ..., P'_m$ and P is the formula schema generalized from P'. This generalization replaces constant symbols not originally occurring in $\mathcal{A}$, the assertion being applied, by new metavariables. A similar *variablization* is a standard technique employed in the context of explanation based generalization [Moo90]. Obviously, the replaced constant symbols must occur in formulas serving as premises, such as a_1, U_1 and F_1 in $a_1 \in U_1$ and $U_1 \subset F_1$ in our example.

Contraposition The second way of acquiring assertion level rules can be viewed as a generalized contraposition, described by the following schema: if r is an existing rule of the form:

$$r = \frac{\triangle \vdash p_1, \ldots, \triangle \vdash p_n}{\triangle \vdash q}$$

then r' below can be acquired by contraposition:

$$r' = \frac{\triangle \vdash p_1, \ldots, \triangle \vdash p_{i-1}, \triangle \vdash p_{i+1}, \ldots, \triangle \vdash p_n, \triangle \vdash \neg q}{\triangle \vdash \neg p_i}$$

For instance, after the acquisition of

$$\frac{\triangle \vdash a \in U, U \subset F}{\triangle \vdash a \in F}$$

two other rules

$$\frac{\triangle \vdash a \in U, a \notin F}{\triangle \vdash U \not\subset F} \qquad \text{and} \qquad \frac{\triangle \vdash a \notin F, U \subset F}{\triangle \vdash a \notin U}$$

can be derived as contrapositions (see [Hua92] for more details).

3.2 The Complete Set of Assertion Level Rules

Now let us turn to our main concern, namely the set of inference rules $Rule(\mathcal{A})$, associated with a particular assertion $\mathcal{A}$. As we have argued, rules in $Rule(\mathcal{A})$ are either generated in a chunking-and-variablization manner, or by contraposition. Therefore:

$$Rules(\mathcal{A}) = R(\mathcal{A}, \mathcal{NK} \cup Contra(\mathcal{NK})) \cup Contra(R(\mathcal{A}, \mathcal{NK} \cup Contra(\mathcal{NK}))) \quad (4)$$

where $R(\mathcal{A}, \mathcal{B})$ denotes the set of rules applying $\mathcal{A}$, which can be acquired in a chunking-and-variablization manner with respect to $\mathcal{B}$, denoting the set of logic level rules at the disposal of the reasoner for constructing logic level proof segment. In our theory, we assume the ND calculus $\mathcal{NK}$ [Gen35], together with their contrapositions, as the available rules at the logic level. $Contra(S)$ denotes the set of rules which are contrapositions of rules in the set of rules S. There are redundancies in $R(\mathcal{A}, \mathcal{NK} \cup Contra(\mathcal{NK}))$ and $Contra(R(\mathcal{A}, \mathcal{NK} \cup Contra(\mathcal{NK})))$, because many rules in the latter may have a direct derivation as well.

Example 1 (continued):
With a rule $\frac{a_1 \in U_1, U_1 \subset F_1}{a_1 \in F_1}$ already acquired from the subset definition, supported by the ND proof segment illustrated in Fig. 1, it is only natural for a human to be able to apply the following contraposition: $\frac{a_1 \in U_1, a_1 \notin F_1}{U_1 \not\subset F_1}$. This, however, has a corresponding compound proof segment of its own, given in Fig. 2.

$$\dfrac{\dfrac{\forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow (\forall_x x \in S_1 \Rightarrow x \in S_2)}{U_1 \subset F_1 \Leftrightarrow \forall_x x \in U_1 \Rightarrow x \in F_1}, \dfrac{a_1 \in U_1, a_1 \notin F_1}{\neg(a_1 \in U_1 \Rightarrow a_1 \in F_1)}}{\neg(\forall_x x \in U_1 \Rightarrow x \in F_1)}}{U_1 \not\subset F_1}$$

Fig. 2. Natural Expansion 2 for Subset Definition

In general, if Fig. 3(a) is the corresponding tree schema for a rule $\frac{c1, c2, b1}{b2}$, acquired, the corresponding tree schema for its contraposition $\frac{b1, \neg b2, c1}{\neg c2}$ can be constructed, using the corresponding contrapositions of the logic level rules, as depicted in Fig. 3(b).

The following property makes a more succinct representation of equation (4) possible [Hua91]:

$$R(\mathcal{A}, Contra(\mathcal{B}) \cup \mathcal{B}) = R(\mathcal{A}, \mathcal{B}) \cup Contra(R(\mathcal{A}, \mathcal{B}))$$

where $\mathcal{B}$ is an arbitrary set of logic level inference rules. A natural corollary is:

$$Contra(R(\mathcal{A}, Contra(\mathcal{B}) \cup \mathcal{B})) \subset R(\mathcal{A}, Contra(\mathcal{B}) \cup \mathcal{B})$$

Intuitively, this means contraposition of compound rules will bring forth no more new rules if contraposition is already applied to elementary rules. Thus

$$Rules(\mathcal{A}) = R(\mathcal{A}, \mathcal{NK}) \cup Contra(R(\mathcal{A}, \mathcal{NK})) \quad (5)$$

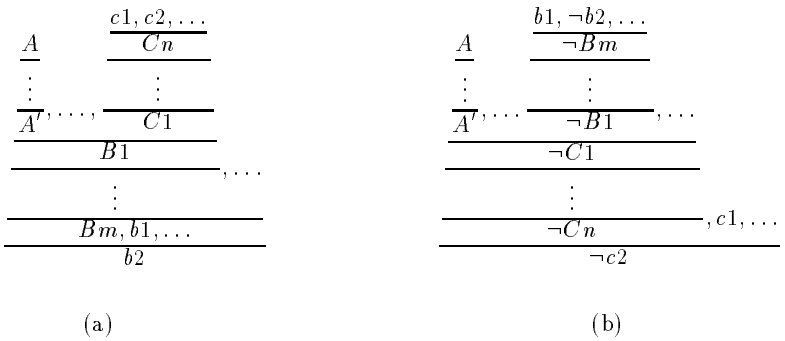


Fig. 3. Expansion for Contrapositions

3.3 Tree Schemata for Assertion Level Inference Rules

As stated in (5), the application of an assertion $\mathcal{A}$ can be covered by the union of $R(\mathcal{A},\mathcal{NK})$ and its contraposition. According to the definition of $R(\mathcal{A},\mathcal{NK})$, each of its rules corresponds to a tree schema generalized from a natural expansion. Therefore $R(\mathcal{A},\mathcal{NK})$ can be represented by a set of tree schemata covering all natural expansions, denoted by $\mathit{Tree}(\mathcal{A},\mathcal{NK})$. Since some members in $\mathit{Tree}(\mathcal{A},\mathcal{NK})$ are subtrees of others and can therefore be omitted, we show in this section that this set can be represented in a very compact way. For almost all examples, $\mathit{Tree}(\mathcal{A},\mathcal{NK})$ consists usually of only one or two trees.

Example 1. (continued)

If we apply the variablization described in section 3 on the proof segment in Fig. 1 by replacing a_1, U_1 and F_1 by metavariables a, U and F, respectively, the tree schema in Fig. 4 can be obtained.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}:\forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow \forall_x x \in S_1 \Rightarrow x \in S_2}{U \subset F \Leftrightarrow \forall_x x \in U \Rightarrow x \in F}}{U \subset F \Rightarrow \forall_x x \in U \Rightarrow x \in F}, \quad U \subset F}{\forall_x x \in U \Rightarrow x \in F}}{a \in U \Rightarrow a \in F}, \quad a \in U}{a \in F}$$

Fig. 4. Tree Schema for Subset Definition

Because every subtree (with the subset definition as one of its leaves) of the tree schema in Fig. 4 is a schema of natural expansion, this tree contains a whole set of assertion level inference rules. Apart from the one listed in (2), $\frac{U \subset F}{\forall_x x \in U \Rightarrow x \in F}$ is another rule contained in this tree, for instance.

Now we are ready to examine the set of proof tree schemata designated by $\mathit{Tree}(\mathcal{A},\mathcal{NK})$. We do this by defining $\mathit{Tree}(\mathcal{A},\mathcal{B})$ as a restricted deductive closure of the composition and decomposition rules in $\mathcal{B}$, an arbitrary set of logic level inference rules. Technically, for all $r \in R(\mathcal{A},\mathcal{B})$, there is a tree schema $t \in \mathit{Tree}(\mathcal{A},\mathcal{B})$, such that r can be accounted for by a subtree of t.

Below is a constructive definition:

i Start with the tree in Fig. 5(a), which corresponds to the rule $\frac{}{\triangle A \vdash A}$,

ii If there is a tree t in the form of Fig. 5(b), $r = \frac{\triangle A \vdash a, \triangle A \vdash p1, \ldots, \triangle A \vdash pn}{\triangle A \vdash Q} \in \mathcal{B}$ is a decomposition rule with respect to a, and if there exists a substitution σ, such that $A' = a\sigma$, then extend t to a tree t' in form of Fig. 5(c).

iii If there is a tree t in the form of Fig. 5(b), and $r = \frac{\triangle A \vdash p'1, \ldots, \triangle A \vdash p'n}{\triangle A \vdash Q} \in \mathcal{B}$ is a composition rule with respect to Q, now if there exists a substitution σ, such that $p = Q\sigma$, then extend t to a tree t' in form of Fig. 5(d).

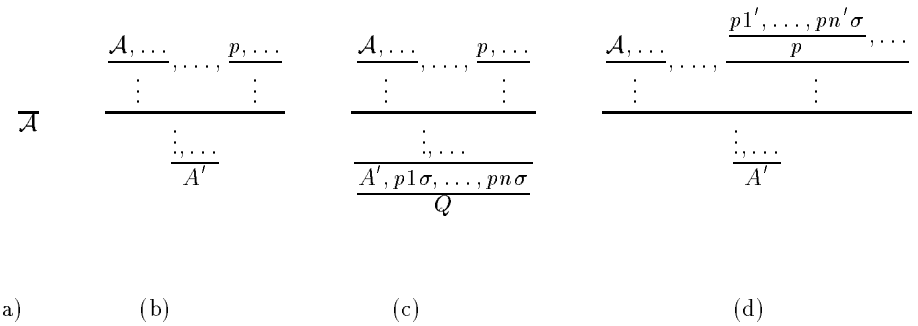


Fig. 5. Construction of Tree Schemata

Some explanations: i) initializes a tree with only one node, corresponding to the initial inference rule $\frac{}{\triangle \vdash A}$, ii) and iii) extend existing trees by decomposing the root or the leaves. The informations contained in this set can be redundant, since some rules accounted for by one tree schema are contrapositions of rules accounted for by another tree schema.

Example 1. (Continued): We illustrate the structure of the tree schemata introduced above by continuing with the example used throughout this paper. Two trees are needed as shown in Fig. 6 and Fig. 7, since the equivalence "$\Leftrightarrow$" is understood as the shorthand of the conjunction of two implications and therefore can be decomposed in two different ways. Table 1 is a list of some of the rules in $Rule(\mathcal{A})$, with their corresponding tree schemata indicated.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A} : \forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow \forall_x x \in S_1 \Rightarrow x \in S_2}{[a] : U \subset F \Leftrightarrow \forall_x x \in U \Rightarrow x \in F}}{[b] : U \subset F \Rightarrow \forall_x x \in U \Rightarrow x \in F}, \quad U \subset F}{[c] : \forall_x x \in U \Rightarrow x \in F}}{[d] : a \in U \Rightarrow a \in F}, \quad a \in U}{[e] : a \in F}$$

Fig. 6. Tree Schema 1 for Subset Definition

Notice, as we argued above, every subtree containing the subset definition as a leaf corresponds to a rule of inference, if we take the other leaves as preconditions

No.	Inference Rule	Derivation(Tree or Contraposition)
(1)	$\frac{\triangle \vdash a \in U, \triangle \vdash U \subset F}{\triangle \vdash a \in F}$	Tree in Fig. 6
(2)	$\frac{\triangle \vdash a \notin F, \triangle \vdash U \subset F}{\triangle \vdash a \notin U}$	Contraposition of (1)
(3)	$\frac{\triangle \vdash a \in U, \triangle \vdash a \notin F}{\triangle \vdash U \not\subset F}$	Contraposition of (1)
(4)	$\frac{\triangle \vdash U \subset F}{\triangle \vdash \forall_x x \in U \Rightarrow x \in F}$	Subtree of Fig. 6, rooted at node [c]
(5)	$\frac{\triangle \vdash a \in U \Rightarrow a \in F}{\triangle \vdash U \subset F}$ where a does not occur in $\mathcal{A}$	Tree in Fig. 7.
(6)	$\frac{\triangle \vdash \forall_x x \in U \Rightarrow x \in F}{\triangle \vdash U \subset F}$	Subtree of Fig. 7 rooted at node [c]
(7)	$\frac{\triangle \vdash U \not\subset F}{\triangle \vdash \neg\forall_x x \in U \Rightarrow x \in F}$	Contraposition of (5)

Table 1. Some Inference Rules for Subset Definition

$$\dfrac{\dfrac{\dfrac{\mathcal{A}:\forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow \forall_x x \in S_1 \Rightarrow x \in S_2}{[a]: U \subset F \Leftrightarrow \forall_x x \in U \Rightarrow x \in F}}{[b]: (\forall_x x \in U \Rightarrow x \in F) \Rightarrow U \subset F}, \dfrac{a \in U \Rightarrow a \in F}{\forall_x x \in U \Rightarrow x \in F}}{[c]: U \subset F}$$

Fig. 7. Tree Schema 2 for Subset Definition

and the root as the conclusion. In other words, only subtrees rooted along the path from the leave which is the assertion being applied to the root, called the *main branch*, are of interest. Fig. 6 has five such subtrees and Fig. 7 has three, namely, the length of the main branch. Nodes along the main branch are numbered in Fig. 6 and Fig. 7 for convenience. Each such subtree represents a rule of inference, directly, and its contraposition, indirectly.

For instance, rule (1) is directly represented by Fig. 6 itself and (2), (3) are contrapositions of (1). Rule (4) is represented by a subtree rooted at node [c] in Fig. 6. Rule (5) is represented by one of the subtrees rooted at [c] in Fig. 7, which has no associated rules because of its variable condition.

4 Abstracting ND-Proofs to Assertion Level Proofs

As describe above, assertion level justifications are used both for compound logic level proof segments satisfying the composition-decomposition constraint and atomic derivations justified by an assertion level inference rule. Logically, the two kinds of derivations are equivalent. In this section an algorithm is devised which replaces as many compound proof segments in machine found ND proofs as possible, by atomic derivations justified by assertion level rules. Most importantly, the replacement is not restricted to natural expansions, but includes other logically equivalent compound segments. This procedure is the preprocessor of *PROVERB*, a system transforming natural deduction proofs into natural language [Hua94]. *PROVERB* is

the explanation component for Ω–MKRP, an interactive proof development environment [HKK+93].

As argued above, in order to produce natural language proofs comparable with proofs found in typical mathematical textbooks, we should first try to replace as many complex proof units as possible by atomic assertion level steps. One straightforward solution is a strict abstraction of the input ND proof by replacing all subproofs satisfying the decomposition-composition constraint by an atomic step justified by the corresponding assertion level inference rule. This approach, however, has a severe drawback: Since automated theorem provers usually work in a manner fundamentally different to that of human beings, the input ND proofs are often quite twisted so that not many units satisfying this constraint can be found. Another approach based on the assertion level rules rather than on the constraint avoids this problem. One way to do so is to go through the entire input proof, and test for every proof node N, if N can also be justified by the application of an assertion. As candidates for such assertions all proof nodes that depends on less assumptions than N could be considered. Apparently, such a procedure nearly reproves the problem based on the input proof. Although this exhaustive procedure may find optimal proofs and its complexity is theoretically still polynomial, it is quite search intensive in the practice. A more restrictive variation is employed in our system that mainly abstracts an existing proof as it is proved, but utilizes the assertion level inference rules instead of the decomposition-composition constraint.

Algorithm: Go through the entire proof tree starting from the root, for each proof node N,

1. Choose as the set of assertions AS the definitions and theorems contributed to the proof of N, namely the leaves of the subtree rooted by N, which are definitions or theorems used.
2. Among the nodes in the subtree rooted by N, test if there exist nodes $p_1, \ldots, p_n$, from that N can be derived by applying an assertion $\mathcal{A}$ in AS. In this case, reduce the proof so that N has $p_1, \ldots, p_n$ as its only direct children and the assertion $\mathcal{A}$ as its new justification.

The applicability of a particular assertion $\mathcal{A}$ can be tested by finding a subtree in $\mathit{Tree}(\mathcal{A}, \mathcal{NK})$ (or one of its contrapositions), and a subtree in the input proof tree rooted by the conclusion N to be justified, so that the leaves match. To maximize the factor of abstraction, we proceed in a top-down manner and gradually search for maximal subtrees satisfying the condition above. For example, suppose we are at the proof node $[e']$ in a segment of an input proof as shown in Fig. 8 (the ND rules used as justifications are omitted). The label *Subset* indicate this hypothesis is the definition of subset.

Since it is recorded that the definition of subset is used as one of its hypothesis, it is tested if any assertion level rule associated with this definition can be applied. For this purpose, we search for a node in the tree schemata in Fig. 6 and Fig. 7 for a node which matches the node $[e']$ in Fig. 8. $[e]$ in Fig. 6 is found. Now we try to find two maximal subtrees rooted by $[e]$ and $[e']$ respectively which match. In this case, they are the two trees themselves. $[p_1]$ and $[p_2]$ are used as new premises of $[e']$ in

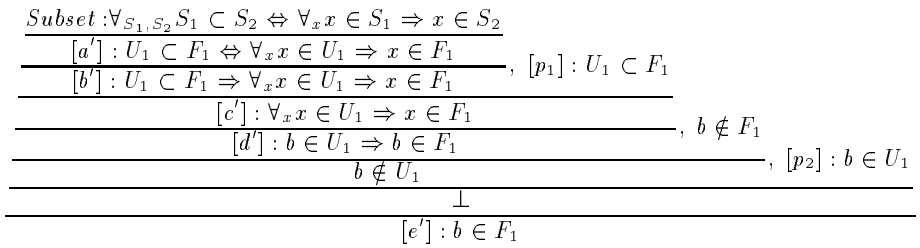


Fig. 8. A Segment of the Input Proof

the abstracted proof, and the definition of subset as the new justification. Thereby the proof segment in Fig. 8 is abstracted to the proof segment in Fig. 9 below:

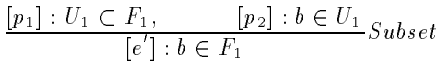


Fig. 9. The abstracted proof segment

Note that the leaf $b \notin F_1$ in Fig. 8 need not be matched, since it is a temporary assumption for the indirect proof step. The search for maximal matching subtrees is carried out in a breath-first manner, upwards from both of the roots. Note also, that not every intermediate node in the input proof segment needs to be matched and only the leaves count. The indirect proof step in the input proof segment in Fig. 8, which is apparently a detour made by the machine, is absorbed.

Searching for maximal subtrees in a breath-first manner may lose the optimal abstraction, since all intermediate nodes of the maximally matched subtree in $Tree(\mathcal{A}, \mathcal{NK})$ must be matched by a node in the input proof. However, this restriction significantly accelerates the process. The worst case of our abstraction algorithm is now only of the order $O(n^2)$, including the cost of generating tree schemata. This happens when no abstraction can be performed. For neatly written input proofs containing segments which structurally resemble tree schemata representing assertion level rules, it can even be nearly linear.

The quality of the resulting proofs depends on the input proofs in the following way:

- The algorithm works well on neatly structured ND proofs. In these cases, the reduction factor depends on the average depth of the terms in the definitions and theorems. Since mathematicians usually avoid using both too trivial and too complicated definitions and theorems, a quite stable reduction factor (about two thirds in terms of the number of the proof lines) is normally achieved.
- Most significant reduction is observed with input proofs which are essentially direct proofs, but containing machine generated detours and redundancies. At the end of this section, we show an example where a machine generated ND proof of 134 lines is shortened to a proof of 15 lines.
- The complete proof transformation procedures described in [And80, Mil83, Pfe87, Lin90] work fairly similar to a tableau prover. They tend to produce proofs which are mainly indirect, if not properly guided by heuristics. Our algorithm performs

poorly on such indirect proofs, where in most of the node only $\perp$ is derived. Although such proofs are also often shortened to the half in length, the resulting proofs are still largely at the level of calculus rules and therefore still too tedious. This problem can be overcome by incorporating techniques that help to avoid indirect proofs (see [Lin90, PN90]) into the process transforming proofs in machine oriented formalisms to ND proofs. Techniques described in [Lin90] can also be adapted to be applied on ND proofs after the transformation.

Let us look at the example below, abstracted from an input proof of 134 lines, generated in the proof development environment Ω–MKRP. It is given in a linearized format, where the last column contains the justification as well as the premises. Eleven of the remaining fifteen steps are at the assertion level. The rest are justified by ND rules of more structural import: They introduce new temporary hypothesis and then discharge them (the *Hyp* and the *Choice rule* in this example). These steps are usually presented explicitly. Groups of trivial steps instantiating quantifiers or manipulating logical connectives are largely abstracted to assertion level steps. Line 7 corresponds to the proof step in Fig. 9, abstracted from the proof segment in Fig. 8. The definitions of *semigroup*, *group*, and *unit* are obvious and therefore omitted in the proof below. "$solution(a, b, c, F, *)$" should be read as "c is a solution of the equation $a * x = b$ in F." Notice, the proof segments replaced by assertion level steps are not necessarily a natural expansion of the latter. In contrast, they are usually proof segments produced by a automated theorem prover, which are logically equivalent to a natural expansion, but contain unnecessary detours. If we replace the assertion level steps in the proof below by their natural expansions, the result is a logic level proof of 43 lines, in contrast to the input proof of 134 lines.

Theorem: Let F be a group and U a subgroup of F, if 1_U is a unit element of U, then $1 = 1_U$.

Abstracted Proof about Unit Element of Subgroups

NN	S;D	Formula	Reason
1.	1; ⊢	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*)$	(Hyp)
2.	1; ⊢	$U \subset F$	(Def-subgroup 1)
3.	1; ⊢	$1_U \in U$	(Def-unit 1)
4.	1; ⊢	$\exists_x x \in U$	(∃ 3)
5.	;5 ⊢	$u \in U$	(Hyp)
6.	1;5 ⊢	$u * 1_U = u$	(Def-unit 1 5)
7.	1;5 ⊢	$u \in F$	**(Def-subset 2 5)**
8.	1;5 ⊢	$1_U \in F$	(Def-subset 2 3)
9.	1;5 ⊢	$semigroup(F,*)$	(Def-group 1)
10.	1;5 ⊢	$solution(u,u,1_U,F,*)$	(Def-solution 6 7 8 9)
11.	1;5 ⊢	$u * 1 = u$	(Def-unit 1 7)
12.	1;5 ⊢	$1 \in F$	(Def-unit 1)
13.	1;5 ⊢	$solution(u,u,1,F,*)$	(Def-solution 7 11 12 9)
14.	1;5 ⊢	$1 = 1_U$	(Th-solution 11 10 13)
15.	1; ⊢	$1 = 1_U$	(Choice 4 14)

The appropriateness of the assertion level is supported by our experience in the verbalization of abstracted proofs using the system *PROVERB* [Hua94]. Taking as input ND style proofs at assertion level, the resulting texts are at an acceptable level of abstraction. Below is the natural language proof generated by *PROVERB*:

The Natural Language Proof
(1)Let F be a group, U be a subgroup of F, 1 be a unit element of F and 1_U be a unit element of U. (2)According to the definition of unit element, $1_U \in U$. (3)Therefore there is an X, $X \in U$. (4)Now suppose that u is such an X. (5)According to the definition of unit element, $u * 1_U = u$. (6)Since U is a subgroup of F, $U \subset F$. (7)Therefore $1_U \in F$. (8)Similarly $u \in F$, since $u \in U$. (9)Since F is a group, F is a semigroup. (10)Since $u * 1_U = u$, 1_U is a solution of the equation $u * X = u$. (11)Since 1 is a unit element of F, $u * 1 = u$. (12)Since 1 is a unit element of F, $1 \in F$. (13)Since $u \in F$, 1 is a solution of the equation $u * X = u$. (14)Since F is a group, $1_U = 1$ by the uniqueness of solution. (15)This conclusion is independent of the choice of the element u.

5 Conclusion and Future Work

This paper proposes a reconstructive approach toward the presentation of machine found proofs. It is argued that after machine found proofs are transformed into ND proofs, a reconstruction should be started anew, to obtain proofs containing justifications at a higher level of abstraction, which are intuitively understood as the application of a definition or of a theorem, collectively called an assertion. We have illustrated that compound proof segments which can be justified as the application of a certain assertion fulfill the so called decomposition-composition constraint. Furthermore, they are logically equivalent to atomic derivations justified by rules of inference at the assertion level. The complete set of such assertion level rules associated to a particular assertion can be represented in a very compact way in form of tree schemata. With the help of these tree schemata, we devised an efficient algorithm abstracting machine generated ND proofs to the assertion level. This algorithm works even better, if adequate heuristics are employed to generate well structured ND proofs.

The significance becomes more evident when it is viewed within the entire spectrum of transforming machine generated proofs into natural language. With natural deduction style proofs composed of mostly assertion level steps as an additional intermediate representation, the proofs passed to the text planner already resemble proofs produced by human mathematician, and therefore lend themselves to a natural specification of presentation strategies. Using the abstraction as a preprocessor which substantially shortens input proofs, we are able to tackle a broad class of proofs containing more than one hundred lines, and the final proofs generated are at a level of abstraction comparable with proofs found in typical mathematical text books, where authors choose a detailed style.

There is no doubt that proofs are often presented by mathematician at a even higher level of abstraction, since a loss factor of 10 to 20 is reported when using systems like AUTOMATH [dB80]. Even more radical expansion factors (about 5,000

to 10,000) are conjectured by experts for harder mathematical problems. To achieve a similar factor of reduction in the proof presentation, a much deeper understanding of the cognitive process of theorem proving is necessary. This work is only a first step toward this direction.

Acknowledgment

Thanks are due to Manfred Kerber and Daniel Nesmith, who read several drafts of this paper carefully, and to Armin Fiedler, who implemented the abstraction algorithm. I am also indebted to two CADE referees for their constructive suggestions.

References

[And80] Peter B. Andrews. Transforming matings into natural deduction proofs. In *Proc. of the 5th CADE*, pages 281–292. Springer, 1980.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In *Proc. of 9th CADE*, pages 111–120. Springer, 1988.

[CAB+86] R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, J.T. Sasaki, and S.F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice Hall, New Jersey, 1986.

[Che76] Daniel Chester. The translation of formal proofs into English. *AI*, 7:178–216, 1976.

[dB80] Nicolaas Govert de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry - Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, 1980.

[Deu71] Peter Deussen. *Halbgruppen und Automaten*. Springer, 1971.

[EP93] Andrew Edgar and Francis Jeffry Pelletier. Natural language explanation of natural deduction proofs. In *Proc. of the first Conference of the Pacific Association for Computational Linguistics*. Centre for Systems Science, Simon Fraser University, 1993.

[FHN72] R. R. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Math. Zeitschrift*, 39:176–210, 1935.

[GMW79] Michael Gordon, Robin Milner, and Christopher Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation*. LNCS 78. Springer, 1979.

[HKK+93] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω–MKRP – a proof development environment. In Hélène Kirschner, editor, *Proc. of the Workshop on Automated Theorem Proving at IJCAI-93*, pages 35–36, Chambéry, France, 1993.

[Hua91] Xiaorong Huang. An extensible natural calculus for argument presentation. SEKI-Report SR-91-03, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1991.

[Hua92] Xiaorong Huang. Applications of assertions as elementary tactics in proof planning. In V. Sgurev and B. du Boulay, editors, *Artificial Intelligence V - Methodology, Systems, Applications*, pages 25–34. Elsevier Science, the Netherlands, 1992.

[Hua93] Xiaorong Huang. An explanatory framework for human theorem proving. In Hans Jürgen Ohlbach, editor, *GWAI-92: Advances in Artificial Intelligence*, LNAI 671, pages 55–66. Springer, 1993.

[Hua94] Xiaorong Huang. *Human Oriented Proof Presentation: A Reconstructive Approach*. PhD thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[JL83] Philip N. Johnson-Laird. *Mental Models*. Harvard Univ. Press, Cambridge, Massachusetts, 1983.

[Lin90] Christoph Lingenfelder. *Transformation and Structuring of Computer Generated Proofs*. PhD thesis, Universität Kaiserslautern, Kaiserslautern, Germany, 1990.

[McD83] David D. McDonald. Natural language generation as a computational problem. In *Brady/Berwick: Computational Models of Discourse*. MIT Press, 1983.

[Mil83] Dale A. Miller. *Proofs in Higher-Order Logic*. PhD thesis, CMU, Pittsburgh, Pennsylvania, USA, 1983.

[Moo90] Raymond J. Mooney. Learning plan schemata from observation: Explanantion-based learning for plan recognition. *Cognitive Science*, 14:483–509, 1990.

[New90] Allen Newell. *Unified Theories in Cognition*. Harvard University Press, Cambridge, MA, 1990.

[Pfe87] Frank Pfenning. *Proof Transformation in Higher-Order Logic*. PhD thesis, CMU, Pittsburgh, Pennsylvania, USA, 1987.

[PN90] Frank Pfenning and Daniel Nesmith. Presenting intuitive deductions via symmetric simplification. In Mark E. Stickel, editor, *Proc. of 10th CADE*, LNAI 449, pages 336–350. Springer, 1990.

[Sho76] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York, 1976.

[WS89] M. R. Wick and J. R. Slagle. An explanation facility for today's expert systems. *IEEE Expert*, 4(1):26–36, 1989.

[WT92] Michael R. Wick and William B. Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54:33–70, 1992.

This article was processed using the LaTeX macro package with LLNCS style