The Presentation of Proofs at the Assertion Level

Xiaorong Huang*
Fachbereich Informatik, Universität des Saarlandes
Postfach 1150, 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. Efforts have been made to transform such machine generated proofs into natural deduction (ND) proofs. Although the single steps are now easy to understand, the entire proof is usually at a low level of abstraction, containing too many tedious steps. Therefore, it is not adequate as input to natural language generation systems.

To overcome these problems, we propose a new intermediate representation, called ND style proofs at the *assertion level*. After illustrating the notion intuitively, we show that the assertion level steps can be justified by domain-specific inference rules, and that these rules can be represented compactly in a tree structure. Finally, we describe a procedure which substantially shortens ND proofs by abstracting them to the assertion level, and report our experience with further transformation into natural language.

*This work was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2, D3)

1 Introduction

This paper concerns the presentation of machine generated proofs. In particular, the natural deduction proofs containing steps at the *assertion level* are introduced, serving as an intermediate representation before proofs are translated into natural language (NL). Most of the existing automated reasoning systems suffer from the problem of only delivering protocols difficult for human users to understand. Therefore, attempts have been made to transform proofs from machine oriented formalisms to natural deduction (ND) proofs [And80, Mil83, Pfe87, PN90, Lin90]. Also some effort has been spent on constructing NL generators, which take as input a ND proof [Che76, McD83].

While proofs in various machine oriented formalisms have rather smoothly been transformed into natural deduction proofs, the latter, however, turn out to be an inappropriate basis for formulating strategies concerning the presentation of argumentative text. The first attempt of transforming natural deduction proofs into natural language was made by D. Chester [Che76]. His program is usually characterized as an example of *direct translation*: After a linerization of an input ND proof, 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], in particular, emphasis was laid on the generation of utterances highlighting important logical structures in the proofs, as well as utterances mediating between subproofs. However, both systems operate on a very low level: the proof steps they translate are exclusively at the level of inference rules of the ND calculus. As a consequence, natural language proofs thus produced contain too many minute details, and are therefore very tedious. If it was still tolerable for the toy examples containing less than a dozen of proof lines, the results would be intolerable, if they were assigned the task of presenting realistic ND proofs containing hundreds of lines, as our system must cope with.

The main problem, lies on the totally flat structure of ND proofs, which can be viewed as a tree with no further structures imposed upon. Since human theorem proving can be viewed as a planning process [Hua93b], the hierarchical structure of proof units plays a crucial role in human proof presentation. First, proof time hierarchical structures lend themselves very naturally to the specification of presentation strategies. For instance, a proof to be presented can often be split and ordered in terms of such structures [Hua93a]. This structure is also important while discussing the depth of proofs to which presentation should be carried out [EP91]. Second, even more important, some compound proof units allow atomic justifications at a higher level of abstraction. A proof is usually substantially shortened by presenting complex units as atomic steps.

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. First, *logic level* justifications are simply verbalizations of the ND inference rules. One example is if we know $P \Rightarrow Q$ and P, we may claim that Q is derived by applying the rule of Modus Ponens. Second, there are justifications at the *assertion level*, evidently at a higher level of abstraction. The following is an example: "since a is an element of 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". Axioms, theorems, and even previously proved intermediate results are *applied* in the same way. We call these justifications collectively the *applications of assertions*. Technically, by an assertions we mean any fact encoded as a logic formula, which is either assumed as true, such as axioms and definitions, or proved to be true previously in the context, such as theorems or lemmas. Third, *proof level* justifications are at a still higher level, accounting for segments containing proof steps at the two lower levels. Proof level justifications are comparatively rare. One occasion is that when a subproof of the current problem is similar to a proof known to the reader, we may resort to this similarity as a justification.

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, human beings can explain such steps by providing a logic level proof segment. On the other hand, assertion level justifications are primitive with respect to pragmatics, since during the presentation process, an assertion level step 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.

In section 2, we first introduce formally the notion of assertion level proof units, and show that they can be achieved by assertion level inference rules. Section 3 defines a tree structure which very compactly represents the set of assertion level inference rules originating from one assertion. Then in section 4, we illustrate how this tree structure can be used to abstract ND proofs to assertion level proofs, and report our experiences with them in the subsequent translation into natural language. Finally, a look into future work concludes this paper.

2 Assertion Level Proof Unit

The existence of a hierarchy of proof units in proofs constructed by human beings and the dual role of the assertion level can be accounted for by a computational model of human deductive reasoning [Hua93b, HKK+92b]. In that theory, a planner constructs a proof by applying proof methods (called tactics in some other systems [GMW79, Con86]) on still pending open goals. The proof under construction is represented as a hierarchical and partially elaborated plan called a *proof tree*. The execution of each proof method results in the integration of a subtree constituting a proof unit with internal structure. According to our theory, 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.

Figure 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.

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

Figure 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]. Roughly speaking, a natural expansion consists of primarily a linear decomposition of the assertion being applied. In Figure 1, this decomposition is carried out along the branch from $\mathcal{A}$ to the root. By decomposition we mean the derivation of an instance or a subformula (compare section 3.1). Other premises involved in the series of decompositions (the leaves $U_1 \subset F_1$ and $a_1 \in U_1$ in Figure 1) can be constructed by compositions. For an example of such composition, see Figure 2. For more details, the readers are referred to [Hua92].

We also proved 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 the 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 [WMF93, Pas93, HKK92a, HKK+92b]. Our computational model also postulates two ways for acquiring new assertion level rules. 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. 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 meta-variables standing for object variables. More generally, hand in hand with deductive steps corresponding to natural expansions with $P'_1, ..., P'_m$ as formulas attached to the leaves and P' as the formula attached to 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 abstracted from $P'_1, ..., P'_m$ and P is the formula schema obtained from P'. This abstraction process replaces constant symbols not originally occurring in $\mathcal{A}$, the assertion being applied, by new meta-variables. A similar *variablization* is a standard technique employed in the context of explanation based learning [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.

The second way of acquiring assertion level rules is 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 as a rule associated with r:

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

For instance, after the acquisition of

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

two other rules

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

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

To summarize, an assertional proof unit is either a compound unit composed of applications of natural deduction inference rules and satisfying the composition-decomposition constraint, or an atomic unit justified by an assertion level inference rule. Assertion level inference rules can either be acquired as byproducts of deduction by *variablization* and *chunking*, or they can be derived as an associated rule of another existing rule. Since compound assertion level units will be presented in the same way as atomic ones, namely as atomic steps justified by a single assertion level rule of inference, only the atomic units are later used to abstract natural deduction proofs.

3 A Structured Representation of Assertion Level Inference Rules

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

$$Rules(\mathcal{A}) = R(\mathcal{A}, \mathcal{NK} \cup Assoc(\mathcal{NK})) \cup Assoc(R(\mathcal{A}, \mathcal{NK} \cup Assoc(\mathcal{NK})))$$

where $R(\mathcal{A}, \mathcal{B})$ denotes the set of rules applying $\mathcal{A}$, which can be acquired in a variablization-and-chunking 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 rules associated with them, as the available rules at the logic level. $Assoc(S)$ denotes the set of rules associated with rules in the set of rules S. However, there are redundancies in $R(\mathcal{A}, \mathcal{NK} \cup Assoc(\mathcal{NK}))$ and $Assoc(R(\mathcal{A}, \mathcal{NK} \cup Assoc(\mathcal{NK})))$, because many rules in the latter may have a direct derivation as well.

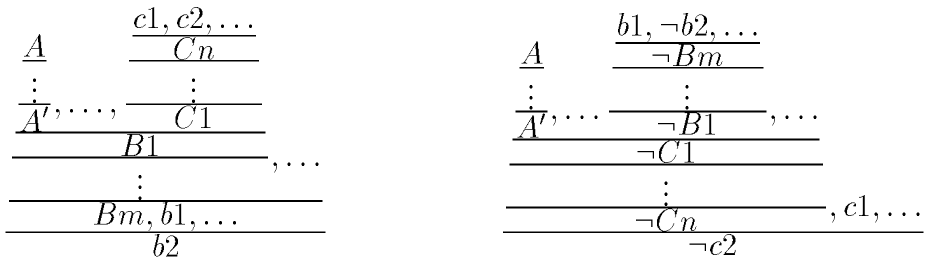
Example 1 (continued):

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 Figure 1, it is only natural for a human to be able to apply the following associated rule: $\frac{a_1 \in U_1, a_1 \notin F_1}{U_1 \not\subset F_1}$. This, however, has as a matter of fact a corresponding compound proof segment of its own, given in Figure 2.

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

Figure 2: Natural Expansion 2 for Subset Definition

In general, if Figure 3(a) is the corresponding tree schema for a rule $\frac{c1,c2,b1}{b2}$, acquired from assertion $\mathcal{A}$, the corresponding tree schema for the associated rule $\frac{b1,\neg b2,c1}{\neg c2}$ can certainly be constructed, using corresponding associated logic level rules, as shown in Figure 3(b)



(a) (b)

Figure 3: Expansion for Associated Rules

The following property makes a more succinct representation possible [Hua91]:

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

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

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

Intuitively, this means if all rules associated to elementary rules are already at the disposal of the variablization-and-chunking process, the derivation of associated rules will bring forth no more new rules. Thus

$$Rules(\mathcal{A}) = R(\mathcal{A}, \mathcal{NK}) \cup Assoc(R(\mathcal{A}, \mathcal{NK}))$$

3.1 Tree Schemata for Assertion Level Inference Rules

As illustrated above, every assertion level proof segment can be abstracted into a tree schema, and every tree schema corresponds to a rule in $R(\mathcal{A}, \mathcal{NK})$. Therefore $R(\mathcal{A}, \mathcal{NK})$ can be represented by a set of tree schemata covering all assertion level compound segment, denoted by $Tree(\mathcal{A}, \mathcal{NK})$. Since some members in $Tree(\mathcal{A}, \mathcal{NK})$ 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, $Tree(\mathcal{A}, \mathcal{NK})$ consists usually of only one or two trees.

Example 1. (continued)

If we apply the variablization process described in the last section on the proof segment in Figure 1 by replacing a_1, U_1 and F_1 by meta-variables a, U and F, respectively, the tree schema in Figure 4 can be derived.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{A}: \forall_{S_1,S_2} S_1 \subset S_2 \Leftrightarrow \forall_x x \in S_1 \Rightarrow x \in S_2}{U_1 \subset F_1 \Leftrightarrow \forall_x x \in U_1 \Rightarrow x \in F_1}}{U_1 \subset F_1 \Rightarrow \forall_x x \in U_1 \Rightarrow x \in F_1}, \quad U_1 \subset F_1}{\forall_x x \in U_1 \Rightarrow x \in F_1}}{a \in U_1 \Rightarrow a \in F_1}, \quad a \in U_1}{a \in F_1}$$

Figure 4: Tree Schema for Subset Definition

Because every subtree (with the subset definition as one of its leaves) of the tree schema in Figure 4 is a schema of assertion level proof segment, 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.

Before providing a constructive definition of $Tree(\mathcal{A}, \mathcal{NK})$, let us first introduce some notions categorizing rules in $\mathcal{NK}$.

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

- a proper subformula of F', or
- a specialization of F' or 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 $\mathcal{NK}$.

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

- a proper subformula of Q', or
- a specialization of Q' or one of its proper subformula, or

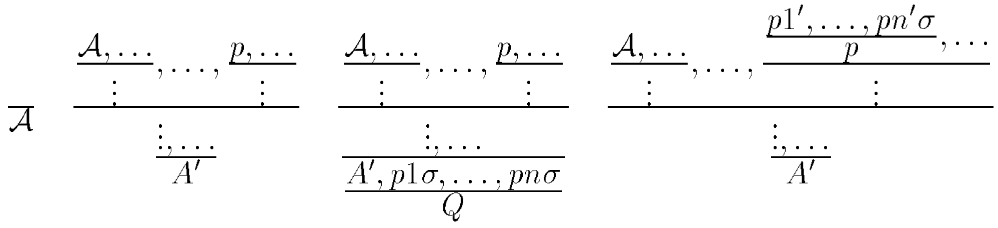
- a negation of one of the first two cases.

Now we are ready to examine the set of proof tree schemata designated by $Tree(\mathcal{A}, \mathcal{NK})$. We do this by defining $Tree(\mathcal{A}, \mathcal{B})$ 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 $t \in Tree(\mathcal{A}, \mathcal{B})$, such that r can be accounted for by a subtree of t. Below is a constructive definition:

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

ii If there is a tree t in the form of Figure 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 Figure 5(c).

iii If there is a tree t in the form of Figure 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 Figure 5(d).



(a) (b) (c) (d)

Figure 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 many rules accounted for by one tree schema are often associated to rules accounted for by another tree schema, described by the schema in section 2.

3.2 Examples

In this section, we illustrate the structure of tree schemata introduced above with the help of two examples. We will show that one or two trees are sufficient to represent an entire class of rules. The second example should also clarify that the concept of the application of assertions is not restricted to mathematics, but also useful for common sense reasoning, as far as logic is used as the representation language. For each example, we list some of the rules contained in $Rules(\mathcal{A})$ (assertion level rule applying the assertion $\mathcal{A}$, and illustrate how they are related to the subtrees in $Tree(\mathcal{A}, \mathcal{NK})$. We first finish the discussion of the subset example used throughout this paper.

Example 1. (Continued): Two trees are needed as shown in Figure 6 and Figure 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. The subset definition is repeated below:

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

Table 1 is a list of some of the rules in $Rule(\mathcal{A})$, and their corresponding tree schemata.

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

Table 1: Some Inference Rules for Subset Definition

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

Figure 6: Tree Schema 1 for Subset Definition

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

Figure 7: Tree Schema 2 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 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. Figure 6 has five such subtrees and Figure 7 has three, namely, the length of the main path. Nodes along the main branch are numbered in Figure 6 and Figure 7 for convenience. Each such subtree represents a rule of inference, directly, and rules associate with it indirectly.

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

The Schubert's steam-roller problem below should illustrate that similar observations can be made for non-mathematical reasoning.

Example 2. The encoding of an axiom is given below (E stands for Eats).

$$\forall_{a:animal}(\forall_{p:plant} E(a,p) \vee (\forall_{a':animal} a' < a \wedge \exists_{p':plant} E(a',p') \Rightarrow E(a,a')))$$

Although $\mathit{Tree}(\mathcal{A}, \mathcal{NK})$ in this case consists of two trees, due to the redundancy of the two symmetric $\vee$D rules, only one is shown in Figure 8. Since the two $\vee$D rules are associated with each other, this tree is complete by itself, see [Hua91]. Some inference rules are listed in Table 2.

No.	Inference Rules	Derivation(Tree or Association)
(1)	$\dfrac{\triangle \vdash \neg E(a,p), a' < a, E(a',p')}{\triangle \vdash E(a,a')}$	Tree in Figure 8.
(2)	$\dfrac{\triangle \vdash a' < a, E(a',p'), \neg E(a,a')}{\triangle \vdash E(a,p)}$	Associated to (1)
(3)	$\dfrac{\triangle \vdash \neg \forall_p E(a,p), a' < a, E(a',p')}{\triangle \vdash E(a,a')}$	Subtree of Figure 8.
(4)	$\dfrac{\triangle \vdash a' < a, E(a',p'), \neg E(a,a')}{\triangle \vdash \forall_p E(a,p)}$	Associated to (3)

Table 2: Some Inference Rules for Steam-Roller Axiom

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall_a(\forall_p E(a,p) \vee (\forall_{a'} a' < a \wedge \exists_{p'} E(a',p') \Rightarrow E(a,a')))}{\forall_p E(a_1,p) \vee (\forall_{a'} a' < a_1 \wedge \exists_{p'} E(a',p') \Rightarrow E(a_1,a'))} \quad \dfrac{\neg E(a_1,p2)}{\neg\forall_p E(a_1,p)}}{\forall_{a'} a' < a_1 \wedge \exists_{p'} E(a',p') \Rightarrow E(a_1,a')}}{a'_1 < a_1 \wedge \exists_{p'} E(a'_1,p') \Rightarrow E(a_1,a'_1)} , \quad \dfrac{a'_1 < a_1, \dfrac{E(a'_1,p_1)}{\exists_{p'} E(a'_1,p')}}{a'_1 < a_1 \wedge \exists_{p'} E(a'_1,p')}}{E(a_1,a'_1)}$$

Figure 8: Tree Schema for Steam-Roller Example

4 Abstracting ND-Proofs to Assertion Level Proofs

This section is devoted to a procedure abstracting input ND proofs to the assertion level, with the help of the tree structure introduced above. This procedure is the preprocessor of PROVERB, a system transforming natural deduction proofs into natural language. Embedded in Ω–MKRP, an interactive proof development enviroment [HKK+92b], the input ND proofs are represented in a linearized version, introduced in [And80]. In this formalism, every proof is a sequence of proof lines, each of the form:

$$\textit{Label} \qquad \triangle \quad \vdash \qquad \textit{Derived-Formula} \qquad\qquad \mathcal{R}(\textit{reason-pointers})$$

where $\mathcal{R}$ is a rule of inference in $\mathcal{NK}$, which justifies the derivation of the derived formula using formulas in lines pointed to by reason-pointers as the preconditions. $\triangle$ is a finite set of formulas, being hypothesis on which the derived formula depends. We want to point out that although the linear format may be not suitable for some purpose, a linearization is required by the translation into natural language.

As argumented above, in order to produce natural language proofs comparable with proofs found in typical mathematical textbooks, we should first try to replace as much complex proof units as possible by atomic assertion level steps. One straightforward procedure would be going through the entire input proof, and test for every proof line, if it can also be justified by the application of an assertion. As candidate for such assertions all formula valid at this point of proof must be considered. To test the applicability of a particular assertion, we have to search the entire previous proof context for potential premises. Apparently, the above sketched procedure effectively reproves the problem based on the input proof. Although this procedure may find more better proofs, it is very search intensive and can easily get lost in a combinatorially explosive search. Another extreme is a strict abstraction of the input ND proof by simply replacing all subproofs satisfying the decomposition and composition constraint by a atomic assertion level step. This approach, however, has a severe drawback as well. Since automated theorem provers usually work in a manner fundamentally different to that of human being, the input ND proofs are often quite twisted so that not many units satisfying this constraint can be found. To reconcile the efficiency and the quality requirement, we employ an algorithm that mainly abstracts an existing proof as it is proved, but utilizes the assertion level inference rules instead of the decomposition-and-composition constraint. In fact, the global structure is taken over from the first approach: we go through the entire input proof, and test for every proof line, if it can also be justified by the application of an assertion. However, only definitions and theorems contributing to its proof, namely those in $\triangle$, are taken as candidates of applicable assertions. And only proof lines transitively reachable via the reason-pointers are considered as potential premises.

Algorithm:

1. go through the entire proof starting from the beginning, for each line,
   (a) choose as the set of assertions AS the set of hypothesis $\triangle$ of the current line
   (b) among the lines transitively reachable via the reason-pointers starting from the current line, test if there exist lines $p_1, \ldots, p_n$, from which the derived formula F of the current line can be derived by applying an assertion $\mathcal{A}$ in AS. This is done by finding a subtree in $Tree(\mathcal{A}, \mathcal{NK})$, so that the derivation is justified by a rule represented by this subtree. In this case, replace the rule of inference $\mathcal{R}$ in this line by a label standing for $\mathcal{A}$ as the new justification, and update reason-pointers so that they point to $p_1, \ldots, p_n$.
2. delete all lines, which are no more involved in the reason hierarchy starting from the conclusion of the entire proof, along the reason-pointers.

A refinement is also made to tackle the situation where more than one applicable assertion level rule is found. In the current implementation, the one that deletes most lines is chosen. Although locally optimal choices do not always lead to a globally optimal choice, it turns out to be tolerable since such cases rarely occur. The search of a subtree in $Tree(\mathcal{A}, \mathcal{NK})$ is significantly accelerated by the subformula relation among nodes in the tree schemata.

Notice the order of searching for abstractions is deliberately chosen for efficiency reasons. By proceeding from the more *shallow* lines gradually towards the final conclusion, the more later abstractions may profit from the earlier ones, since the reason pointers of the current proof line are updated with every abstraction.

Despite its simpleness, the current algorithm substantially shortens input ND proofs of a broad class. 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 with 134 lines is abstracted to a proof of 15 lines. The algorithm also 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 involved in the proof. Since mathematicians usually avoid using both too trivial and too complicated definitions and theorems, a quite stable reduction factor (about two thirds in length) is normally achieved. Finally, the algorithm performs very poorly on machine generated proofs which are mainly indirect, i.e., in most of the lines only bottom is derived. Despite of a reduction factor of about one third in length, the remaining proof lines are still largely at the level of calculus rules and the proof is therefore still too tedious.

Let us look at the example below, abstracted from an input proof of 134 lines, generated in the proof development environment Ω–MKRP. 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, or split a problem into cases. These steps will usually be presented explicitly later. Groups of trivial steps instantiating quantifiers or manipulating logical connectives are largely abstracted to assertion level steps. A proof segment with four extra lines, being the linearized version of the proof tree in Figure 1 as the matter of fact, is needed even in a neatly written input ND proof to achieve step 7 from step 2 and 5. 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, it is usually a logically equivalent but structurally more complex proof segments produced by automated theorem provers. If we replace the assertion level steps in the proof below by their natural expansions correspondingly, 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

NNo	S;D		Formula	Reason
1.	;1	$\vdash$	$G : group(F, *) \wedge subgroup(U, F, *) \wedge$ $unit(F, 1, *) \wedge unit(U, 1_U, *)$	(hyp)
2.	1;	$\vdash$	$U \subset F$	(Def-subgroup 1)
3.	1;	$\vdash$	$1_U \in U$	(Def-unit 1)
4.	1;	$\vdash$	$\exists_x x \in U$	($\exists$ 3)
5.	;5	$\vdash$	$u \in U$	(hyp)

⋮

6.	1;5	$\vdash$	$u * 1_U = u$	(Def-unit 1 5)
7.	1;5	$\vdash$	$u \in F$	**(Def-subset 2 5)**
8.	1;5	$\vdash$	$1_U \in F$	(Def-subset 2 3)
9.	1;5	$\vdash$	$semigroup(F, *)$	(Def-group 1)
10.	1;5	$\vdash$	$solution(u, u, 1_U, F, *)$	(Def-solution 6 7 8 9)
11.	1;5	$\vdash$	$u * 1 = u$	(Def-unit 1 7)
12.	1;5	$\vdash$	$1 \in F$	(Def-unit 1)
13.	1;5	$\vdash$	$solution(u, u, 1, F, *)$	(Def-solution 7 11 12 9)
14.	1;5	$\vdash$	$1 = 1_U$	(Th-solution 11 10 13)
15.	1;	$\vdash$	$1 = 1_U$	(Choice 4 14)
Thm.	1;	$\vdash$	$1 = 1_U$	()

The appropriateness of the assertion level is supported by a computational model for human proof presentation, realized in the system PROVERB [Hua93a]. We have already tested it with assertion level proofs as input and the experience is quite positive. Even without sophisticated language generation mechanism, resulting texts are at an acceptable level of abstraction [Hua90]. Below is the natural language proof generated by PROVERB, using the generator TAG-GEN-7 [Kil93] as its linguistic component.

The Natural Language Proof
(1)Let F be a group and let U be a subgroup of F and let E be an unit element of F and let E_U be an unit element of U. (2)According to the definition of unit element $E_U \in U$. (3)Therefore there is a X, $X \in U$. (4)Now suppose that U is such an X. (5)According to the definition of unit element $U1 * EU = U1$. (6)Since E_U is an unit element of U $E_U \in U$. (7)Since U is a subgroup of F $U \subset F$. (8)Therefore $E_U \in F$. (9)Similarly $U_1 \in F$ since $U_1 \in U$. (10)Since F is a group F is a semigroup. (11)Since $U_1 * E_U = U_1$ EU is a solution of the equation $U * X = U$. (12)Since E is an unit element of F $U_1 * E = U_1$. (13)Since E is an unit element of F $E \in F$. (14)Since $U_1 \in F$ E is a solution of the equation $U * X = U$. (15)Since F is a group $E_U = E$ by the uniqueness of solution. (16)This conclusion is independent of the choice of the element U.

5 Conclusion and Future Work

This paper discussed with the presentation of natural deduction proofs, translated from proofs in more machine oriented formalisms. An algorithm is proposed to abstract such ND proofs to the assertion level, to allow for justifications at a higher level of abstraction. We have illustrated that proof units which can be justified as the application of a certain assertion can also be justified by an assertion level rule of inference. The complete set of such assertion level rules related 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.

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.

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.

Acknowledgement

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.

References

[And80] Peter B. Andrews. Transforming Matings into Natural Deduction Proofs. In *Proc. of the CADE-80*, pages 281–292, 1980.

[Che76] Daniel Chester. The Translation of Formal Proofs into English. *AI*, 7:178–216, 1976.

[Con86] Robert L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, Inc., 1986.

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

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

[EP91] Andrew Edgar and Francis Jeffry Pelletier. Natural language explanation of natural deduction proofs. Personal Communication, 1991.

[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 Verlag, 1979.

[HKK92a] Xiaorong Huang, Manfred Kerber, and Michael Kohlhase. Methods - the basic units for planning and verifying proofs. SEKI Report SR-92-20, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[HKK+92b] Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Daniel Nesmith, Jörn Richts, and Jörg Siekmann. Ω–MKRPa proof development environment. SEKI Report SR-92-22, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[Hua90] Xiaorong Huang. Reference Choices in Mathematical Proofs. In *Proc. of ECAI-90, L. C. Aiello (Ed)*, pages 720–725. Pitman Publishing, 1990.

[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, *Proc. of the 5th International Conference on Artificial Intelligence - Methodology, Systems, Applications*, pages 25–34. Elsevier Science Publishers B.V., the Netherlands, 1992.

[Hua93a] Xiaorong Huang. A Computational Model for Proof Presentation. SEKI Report to appear, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1993.

[Hua93b] Xiaorong Huang. An explanatory framework for human theorem proving. In Hans Jürgen Ohlbach, editor, *Proc. of GWAI-92*, pages 55–66. Springer Verlag, 1993. LNAI 671.

[Kil93] Anne Kilger. Using utags for incremental and parallel generation. *Computational Intelligence*, to appear, 1993.

[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, Cambridge, MA, 1983.

[Mil83] Dale A. Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, 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.

[Pas93] Dominique Pastre. Automated theorem proving in mathematics. *Annals of Artificial Intelligence and Mathematics*, 1993.

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

[PN90] Frank Pfenning and Daniel Nesmith. Presenting Intuitive Deductions via Symmetric Simplification. In Mark E. Stickel, editor, *Proc. of the CADE-90*, pages 336–350. Springer Verlag, 1990. LNAI 449.

[WMF93] F. Javier Thayer William M. Farmer, Joshua D. Guttman. Imps: An interactive mathematical proof system. *Jaornal of Autmated Reasoning*, to appear, 1993.