Structuring of Computer-Generated Proofs by Cut Introduction

Uwe Egly[1] and Karin Genther[2]

[1] Abt. Wissensbasierte Systeme 184/3, TU Wien
Treitlstraße 3, A–1040 Wien
e-mail: uwe@kr.tuwien.ac.at
[2] Inst. für Theoretische Informatik, Med. Universität zu Lübeck
Wallstraße 40, D–23560 Lübeck
e-mail: genther@informatik.mu-luebeck.de

Abstract. As modern Automated Deduction systems rely heavily on the use of a machine-oriented representation of a given problem, together with sophisticated redundancy-avoiding techniques, a major task in convincing human users of the correctness of automatically generated proofs is the intelligible representation of these proofs. In this paper, we propose the use of the cut-rule in the human-oriented presentation of computer-generated proofs. The intelligent application of cuts enables the integration of essential lemmata and therefore shortens and structures proof presentation. We show that many translation techniques in Automated Deduction, such as antiprenexing and some forms of normal form translations, can be described as cuts and are indeed part of the deductive solution of a problem. Furthermore, we demonstrate the connection between symmetric simplification, quantorial extension principles and the application of the cut-rule.

1 Introduction

Most of today's Automated Deduction (AD) systems use low-level formulae and proof representations. Formulae are represented as (a set of) clauses while proofs are represented as trees or sequences of clauses resulting from applications of low-level inference rules like resolution. It has been observed for a long time that such machine-oriented representations of proofs are not well suited for humans. It is widely acknowledged that the performance of AD systems strongly depends on such machine-oriented representations together with powerful redundancy-avoiding techniques, but for the acceptance of AD systems, a user-friendly interface is indispensable. The situation is similar to the case of writing and debugging programs in higher programming languages. While a compressed machine-oriented representation of the program is the input of the central unit, additional information enables the debugger to provide source code information of the high-level program and allow stepping, tracing and manipulating data in terms of the high-level program. In contrast to the debugging scenario, the generation of human-oriented presentations of automatically obtained proofs is a difficult task, mainly because a good proof structure requires

ingenious lemmata and definitions. In today's AD systems, proofs are generated which are exclusively based on information provided in the given formula. Such proof systems are called *analytic* and *cut-free*. These properties correspond to the method of analyzing the structure of the given formula and to use no cuts (to be described later). If mathematicians prove theorems, they structure the theory and the proof by introducing names and new concepts by definitions and by using lemmata. The choice of appropriate lemmata is the key device to obtain readable, well-structured proofs.

A technical device how lemmata can be integrated into a calculus is the cut rule:

$$\frac{\Gamma_1 \vdash \Delta_1, F \quad F, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ cut$$

If we read this rule from bottom to top then this rule says that, instead of deriving $\Delta_1^\vee \vee \Delta_2^\vee$ from $\Gamma_1^\wedge \wedge \Gamma_2^\wedge$, we can derive $\Delta_1^\vee \vee F$ from $\Gamma_1^\wedge$ and $\Delta_2^\vee$ from $F \wedge \Gamma_2^\wedge$. Thereby, $\Delta^\vee$ ($\Gamma^\wedge$) denotes the disjunction (conjunction) of all elements in Δ (Γ). The formula F is the ingenious lemma which usually does not occur in the lower sequent. This is the main problem with this rule in automated proof search: how can we know what F is?

Things change if we already have a proof in some calculus (say resolution) and we want to translate it into another calculus (say NK or LK). Most of the techniques proposed so far for such translations (and many techniques in AD systems) are indeed applications of the cut rule. For systems where proofs have to be manipulated in order to fulfill other tasks, such as extraction of programs or explanation-based generalization, it is crucial not only to understand the use of literals in the proof, but also to make visible the fundamental structure of the proof.

In this paper, we examine different translation techniques and show how they are related to the cut rule. In Section 3, we show how translation schemata of computer-generated proofs into NK-proofs can be improved. Most of these translations suffer from a frequent use of proofs by contradiction which are mainly introduced if more than one substitution instance of a quantified subformula occurs in the proof. The ad-hoc solution, symmetric simplification [25], introduced to reduce the number of proofs by contradiction is identified as a specific form of the cut rule in Section 4. In Section 5, we show that antiprenexing and some translation schemata to normal forms are indeed restricted forms of the cut rule. Moreover, well-known preprocessing reductions can be described in a similar way.[3] In Section 6, we consider quantorial extension principles. It is well known [5, 11] that such extension principles are (restricted) variants of the cut rule. We show, if we already have a (cut-free) proof, how the structure (and length) of such a proof can be improved by introducing cuts representing applications of quantorial extension principles.

[3] It is common practice to consider such preprocessing activities not to be part of the deductive solution. Our results indicate that this viewpoint is wrong; all these techniques are indeed "inferential" and belong to the deductive solution.

2 Definitions and Notations

We consider a usual first-order language with function symbols. We use Gentzen's calculus of classical natural deduction (NK) as given in [19]. The calculus NK consists of the axiom of the excluded middle (*tertium non datur*), the rule *ex falso quodlibet* and logical and quantifier rules. Let A, B, C, and F denote formulae and F_x^t the formula which results from F by substituting all free occurrences of the variable x in F by the term t. The term t is any term not containing a bound variable.

tertium non datur: $A \vee \neg A$ *ex falso quodlibet*: $\dfrac{\bot}{A}\ \bot\mathrm{E}$

Logical Rules

Negation: $\dfrac{\begin{array}{c}[A]\\ \bot\end{array}}{\neg A}\ \neg\mathrm{I}$ $\qquad$ $\dfrac{A \quad \neg A}{\bot}\ \neg\mathrm{E}$

Conjunction: $\dfrac{A \quad B}{A \wedge B}\ \wedge\mathrm{I}$ $\qquad$ $\dfrac{A \wedge B}{A}\ \wedge_1\mathrm{E}$ $\qquad$ $\dfrac{A \wedge B}{B}\ \wedge_2\mathrm{E}$

Disjunction: $\dfrac{A}{A \vee B}\ \vee_1\mathrm{I}$ $\qquad$ $\dfrac{B}{A \vee B}\ \vee_2\mathrm{I}$ $\qquad$ $\dfrac{A \vee B \qquad \begin{array}{c}[A]\\ C\end{array} \qquad \begin{array}{c}[B]\\ C\end{array}}{C}\ \vee\mathrm{E}$

Implication: $\dfrac{\begin{array}{c}[A]\\ B\end{array}}{A \to B}\ \to\mathrm{I}$ $\qquad$ $\dfrac{A \quad A \to B}{B}\ \to\mathrm{E}$

Quantifier Rules

Quantifier $\forall$: $\dfrac{F(u)}{\forall x.F(x)}\ \forall\mathrm{I}$ $\qquad$ $\dfrac{\forall x.F(x)}{F(t)}\ \forall\mathrm{E}$

Quantifier $\exists$: $\dfrac{F(t)}{\exists x.F(x)}\ \exists\mathrm{I}$ $\qquad$ $\dfrac{\exists x.F(x) \qquad \begin{array}{c}[F(u)]\\ C\end{array}}{C}\ \exists\mathrm{E}$

$\forall$I and $\exists$E must fulfill the *eigenvariable condition*, i.e., the variable u does not occur in $\forall x.F(x)$, $\exists x.F(x)$ or C nor in any assumption on which $\forall x.F(x)$ or C (except for $F(u)$) depends.

The *major* premise is the formula containing the occurrence of the logical operator which is eliminated in the E-rule application; all other premises are called *minor*.

It is well known (see [19]) that *tertium non datur* can be replaced by the following inference.

$$\frac{\neg\neg A}{A}\ \neg\neg\mathrm{E}$$

For convenience, we assume to have both principles at hand, even in one proof.

The intuitionistic calculus of natural deduction NJ consists of all the rules of NK except for *tertium non datur* and $\neg\neg$E.

In a *normal* proof, each major premise of an E-rule is either an assumption or a conclusion of an E-rule different from $\vee$E, $\exists$E.

We assume familiarity with sequent calculi and use the following form of Gentzen's calculus LK. In contrast to [19], rule applications are allowed at arbitrary places in a sequent, thus the exchange rule can be omitted. Let Γ, Δ, Π and Λ denote sequences of formulae and let A and B denote arbitrary formulae. Initial sequents (axioms) are of the form $A \vdash A$ for a formula A.

Logical Rules

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\ \neg\mathrm{l} \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ \neg\mathrm{r}$$

$$\frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge_1\mathrm{l} \qquad \frac{B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge_2\mathrm{l} \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\ \wedge\mathrm{r}$$

$$\frac{A, \Gamma \vdash \Delta \qquad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}\ \vee\mathrm{l} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B}\ \vee_1\mathrm{r} \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}\ \vee_2\mathrm{r}$$

$$\frac{\Gamma \vdash \Delta, A \qquad B, \Gamma \vdash \Delta}{A \to B, \Gamma \vdash \Delta}\ \to\mathrm{l} \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B}\ \to\mathrm{r}$$

Quantifier Rules

$$\frac{A(t), \Gamma \vdash \Delta}{\forall x.A(x), \Gamma \vdash \Delta}\ \forall\mathrm{l} \qquad \frac{\Gamma \vdash \Delta, A(u)}{\Gamma \vdash \Delta, \forall x.A(x)}\ \forall\mathrm{r}$$

$$\frac{A(u), \Gamma \vdash \Delta}{\exists x.A(x), \Gamma \vdash \Delta}\ \exists\mathrm{l} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, \exists x.A(x)}\ \exists\mathrm{r}$$

$\forall$r and $\exists$l must fulfill the eigenvariable condition, i.e., the variable u does not occur in the lower sequent. Again, the term t is any term not containing a bound variable.

Structural Rules

Cut: $$\frac{\Gamma \vdash \Delta, A \qquad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}\ cut$$

Weakening: $$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \mathrm{wl} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ \mathrm{wr}$$

Contraction: $$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \mathrm{cl} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ \mathrm{cr}$$

3 Transformation of Computer-Generated Proofs

A major problem in Automated Deduction is the presentation of proofs found by sophisticated and efficient deduction systems in a form more suitable for humans to read. Calculi which are often considered (more) adequate for representing proofs are Gentzen's NK and LK (see [19]). A number of publications develop

procedures for transforming automatically generated proofs stated in various deduction formalisms (such as expansion-tree proofs or refutation graphs) into variants of NK and LK (e.g. see [2], [7], [17], [21], [22], [23], and [24]).

Many of the transformations to NK suffer from a frequent use of proofs by contradiction, which seem to be not "intuitive" for the reader. For instance, such proofs are introduced if more than one substitution instance of a subformula is required in the automatically generated proof. Consider the following NK-proof "by contradiction" of the formula $p(a) \vee p(b) \rightarrow \exists x.p(x)$ as an example.

$$\dfrac{\dfrac{\dfrac{\dfrac{[\neg\exists x.p(x)]_2}{\neg p(a)}\ \neg\exists\mathrm{E} \quad \dfrac{[\neg\exists x.p(x)]_2}{\neg p(b)}\ \neg\exists\mathrm{E}}{\neg p(a) \wedge \neg p(b)}\ \wedge\mathrm{I}}{\neg(p(a) \vee p(b))}\ \neg\wedge\text{-taut} \qquad [p(a) \vee p(b)]_1}{\dfrac{\dfrac{\dfrac{\bot}{\neg\neg\exists x.p(x)}\ \neg\mathrm{I}_2}{\exists x.p(x)}\ \boxed{\neg\neg\mathrm{E}}}{(p(a) \vee p(b)) \rightarrow \exists x.p(x)}\ \rightarrow\mathrm{I}_1}\ \neg\mathrm{E}$$

The rules $\neg\exists$E, $\neg\wedge$-taut and $\neg\neg$E are derived rules in NK and are used here to keep the size of the example manageable. The framed application of $\neg\neg$E is not necessary for the proof of the formula because it can be proven intuitionistically, i.e., without the use of *tertium non datur*. The proof structure is very obscure because of the use of this classical inference and does not reflect the simple structure of e.g. a connection proof.

In [18], a transformation procedure from an extension procedure to NK is suggested that avoids the use of proofs by contradiction in case that more than one substitution instance of a quantified subformula occurs in the proof. The underlying principle of the transformation rule used in such a case is first to generalize each of the substitution instances and then to use a "contracting" rule in order to derive the quantified subformula. An example for the case of two substitution instances of the quantified subformula is the following short proof of the formula $p(a) \vee p(b) \rightarrow \exists x.p(x)$.

$$\dfrac{\dfrac{[p(a) \vee p(b)]_1 \qquad \dfrac{[p(a)]_2}{\exists x.p(x)}\ \exists\mathrm{I} \qquad \dfrac{[p(b)]_2}{\exists x.p(x)}\ \exists\mathrm{I}}{\exists x.p(x)}\ \vee\mathrm{E}_2}{(p(a) \vee p(b)) \rightarrow \exists x.p(x)}\ \rightarrow\mathrm{I}_1$$

Obviously, this direct proof is shorter, more readable, and no derived rules are necessary. In the next section, we consider a technique, symmetric simplification, which was introduced in order to remove "obscure" parts of translated proofs. We show that this technique is a special form of cut introduction.

4 Symmetric Simplification

In [25], a simplification and restructuring method for NK-proofs is proposed. The proofs which are restructured have been obtained by a translation of expansion-tree proofs into NK-proofs. In almost all translations of expansion-tree proofs or resolution proofs into NK-proofs [21, 23, 24], an overwhelming number of indirect proofs are present. The main reasons for this are the generation of $\bot$ and the use of *ex falso quodlibet* if more than one substitution instance of a quantified (sub)formula occurs in the proof. We first recapitulate symmetric simplification and then we show the connection to the cut rule.

The algorithm introduced in [25] uses the fact that, instead of the rule of indirect proof, the law of excluded middle ($A \vee \neg A$) may be included in the logical calculus (i.e., NK). Given a set Γ of assumptions and a conclusion C to be deduced from Γ, there are three typical situations where a classical rule of inference has to be applied:

1. $C = F' \vee F''$ and neither F' nor F'' alone follow from Γ,
2. $C = \exists x.F'(x)$ and there exists no single instance t such that $F'(t)$ follows from Γ,
3. C is atomic[4]

and no progress can be made by applying an intuitionistic rule to formulae of Γ. These situations are referred to as *translation impasses*: a standard translation procedure can no longer make progress by applying intuitionistic rules of inference and hence uses the rule of indirect proof (e.g., the proof of $(p(a) \vee p(b)) \rightarrow \exists x.p(x)$ has to cope with an impasse of the second type). The following illustrates the general pattern for an NK-proof using the law of excluded middle:

$$\dfrac{\begin{array}{ccc} & [A]_1 & [\neg A]_1 \\ & \vdots\ \mathcal{D}_1 & \vdots\ \mathcal{D}_2 \\ A \vee \neg A & C & C \end{array}}{C}\ \vee\mathrm{E}_1$$

In case one of the impasses occurs in the translation process, the symmetric simplification procedure uses heuristics for choosing a formula A (see [25] for detailed examples). The algorithm then tries to simplify A and $\neg A$ simultaneously in such a way that both subproof obligations ($\mathcal{D}_1$ and $\mathcal{D}_2$) can still be fulfilled. Since A and $\neg A$ remain identical apart from the negation sign the method is called "symmetric". The simplification of A consists of three steps:

1. Single Instantiation: Subformulae of A of the form $\forall x.A'(x)$ are replaced by $A'(t)$, if the term t instantiates x somewhere in the deduction $\mathcal{D}_1$ (dual for $\neg A$ and $\mathcal{D}_2$).

[4] In this case, there is at least one assumption of the form $F' \rightarrow F''$ or $\neg F'$ and for any such an assumption, C does not follow from the remaining assumptions.

2. Single Deletion: If there exists a positive (negative) subformula occurrence $A' \wedge A''$ ($A' \vee A''$) in A and none of the instances of A' is used in $\mathcal{D}_1$, erase A' from A and $\neg A$ (dual for positive subformulae $\neg A' \vee \neg A''$ in $\neg A$ and $\mathcal{D}_2$).
3. Propositional Restructure (Single Mating Change): Roughly speaking, this step tries to change $\mathcal{D}_1$ in such a way as to avoid the use of an assumption A' (as in the scenario for *Single Deletion*), i.e., redundancies introduced when creating $A \vee \neg A$ can be deleted. This step may enable further *Single Instantiations* and *Single Deletions*.

The following illustrates that symmetric simplification is indeed cut introduction.[5]

$$\dfrac{A \vee \neg A \qquad \begin{matrix}[A]_1 \\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[\neg A]_1 \\ \vdots \\ C\end{matrix}}{C}\ \vee\mathrm{E}_1 \qquad \dfrac{\vdash A \vee \neg A \qquad \dfrac{\dfrac{A, \Gamma \vdash C}{A, \Gamma, \Delta \vdash C}\ wl \qquad \dfrac{\neg A, \Delta \vdash C}{\neg A, \Gamma, \Delta \vdash C}\ wl}{A \vee \neg A, \Gamma, \Delta \vdash C}\ \vee l}{\Gamma, \Delta \vdash C}\ cut$$

The formula $A \vee \neg A$ and the corresponding $\vee$E-inference are only needed in the NK-proof to connect the two subderivations of C and to close the indicated occurrences of A and $\neg A$. If the formula A is a quantified formula, for instance, of the form $\forall x.B(x)$ then symmetric simplification essentially coincides with an introduction of a classical tautology of the form $\forall x.(B(x) \vee \neg B(x))$ and a *functional extension* (F-extension)[6] step resulting in the formula $\forall x.B(x) \vee \exists x.\neg B(x)$ (see [13, 14] and [5] for a detailed discussion). We will come back to this issue in Section 6.

5 Preprocessing and Cuts

Early books on first-order automated theorem proving favored prenex forms because of the simplicity of describing skolemization. From a logical point of view, this favor does not matter because a first-order formula is logically equivalent to any of its prenex forms. From a computational (proof search) point of view, there is a significant distinction between proof search for prenex forms and proof search for non-prenex forms. Baaz and Leitsch [6] showed that there is a class of formulae $(F_i)_{i \in \mathbb{N}}$ [26] for which the following holds. There exist triple-exponential analytic proofs of F_i, but there exists a prenex form of F_i such that any analytic proof of this prenex form is not elementary. Since the search space is elementarily related to the shortest analytic cut-free proof, a similar relation holds for the different search spaces.

[5] The relation between symmetric simplification and cut is even more obvious in the context of tableaux with cut, because there, cut *directly* encodes the principle of bivalence.

[6] F-extension is introduced in Section 6.

The important observation is that, given a prenex form, a non-prenex form can be obtained by an application of cut. Hence, antiprenexing [7, 15] is a form of cut introduction.

$$\dfrac{\overset{\alpha}{\vdash F'} \qquad \overset{\beta}{F' \vdash F}}{\vdash F}\ cut \tag{1}$$

The proof of the antiprenexed formula F' is denoted by α. The proof β of the sequent $F' \vdash F$ is an "instance" of the correctness proof of antiprenexing. In a presentation of the proof, this proof is not visible by default, but can be generated easily in a systematic way without search.

In many cases, antiprenexing does not have such an extreme (non-elementary) impact on proof length and search space. From a practical point of view, however, antiprenexing can simplify proof search even if there is not a considerable length reduction of the proof. For illustration, consider a normal NK-proof of

$$G = \exists x \forall y.(p(x) \rightarrow p(y)). \tag{2}$$

This formula has a simple proof in LK (even without cut), but any normal NK-proof is rather complex.[7]

If antiprenexing is applied, we get $H = (\forall x.p(x)) \rightarrow \forall y.p(y)$. The cut introduced by antiprenexing is encoded as an application of $\vee$E in the NK-proof. Again, β is an instance of the correctness proof of antiprenexing.

$$\dfrac{\begin{array}{c}\beta \\ \vdots \\ G \vee \neg H\end{array} \qquad \dfrac{\dfrac{\dfrac{\dfrac{[\forall x.p(x)]_2}{p(u)}\,\forall\mathrm{E}}{\forall y.p(y)}\,\forall\mathrm{I}}{H}\,\rightarrow\mathrm{I}_2 \qquad [\neg H]_1}{\dfrac{\bot}{G}\,\bot\mathrm{E}}\,\neg\mathrm{E}}{G}\,\vee\mathrm{E}_1$$

Skolemization *cannot* be described as a cut in this simple manner. The problem is that a sequent of the form

$$F' \vdash F$$

is required where F' is a skolemized form of F. If we can derive a sequent of the form $\vdash F'$ then an application of cut would suffice to derive $\vdash F$. However, the right upper sequent in (1) is not derivable. Let $F := \forall x.(p(x) \vee \neg p(x))$. Then F' is of the form $p(c) \vee \neg p(c)$ for a Skolem constant c. But

$$p(c) \vee \neg p(c) \vdash \forall x.(p(x) \vee \neg p(x))$$

[7] The interested reader is invited to search for a normal proof of G. We use G because it is identified as a "hard" formula for symmetric simplification (see [25]).

is not derivable because of the eigenvariable restriction of $\forall r$. But since skolemization is a validity-preserving operation, we can allow nonlogical axioms of the form $F' \vdash F$. Then skolemization can be described as an application of cut as in (1).

Various skolemization techniques are available [10, 1, 3, 20] which fit into the above scheme (by modifying F'). Usually, skolemization is a computationally inexpensive operation but some extended and more expensive forms are available. In general, these extended skolemization techniques produce much "better" skolemized forms [20].

Translations to normal forms (e.g., disjunctive normal form (DNF) or negation normal form (NNF)) can be described in an analogous way. For some normal forms, translations can be very harmful with respect to the minimal proof length of the resulting normal form (e.g., for DNF or for some kinds of structure-preserving translations) [4, 16]. Assume we want to translate a given formula F into DNF F''. As we already know, skolemization can be encoded as an application of the cut rule. Consider the following derivation.

$$\dfrac{\dfrac{\vdash F'' \quad F'' \vdash F'}{\vdash F'}\ \mathit{cut}\ (\mathrm{DNF}) \qquad F' \vdash F}{\vdash F}\ \mathit{cut}\ (\mathrm{Sk})$$

The lower cut encodes skolemization, the upper cut the translation of a skolemized form of F into DNF. Other forms of translations to a normal form can be described similarly. In the same manner, preprocessing reduction steps [8] (like, e.g., purity reduction) can be encoded as cuts. As the examples suggest, any translation of a formula into a normal form as well as many simplifications of the resulting normal form can be described as cuts. Hence, these operations always belong to the deductive solution of a problem.

6 Quantorial Extension Principles

In this section, we explain the basic ideas about quantorial extension principles. We first start with an example and introduce the definition afterwards.

Let us consider a formula of the form

$$\forall x \forall y.(p(x,y) \vee q(x,y)). \tag{3}$$

The variables x and y occur in both literals; for this reason, no antiprenexing is possible. By the valid formula

$$(\forall x \forall y.(p(x,y) \vee q(x,y))) \rightarrow \forall x.(\forall y.p(x,y) \vee \exists y.q(x,y)) \tag{4}$$

together with (3), the consequent of (4) can be derived by modus ponens. Obviously, a formula with a newly introduced $\exists$-quantifier is derived. If this formula is skolemized then $p(x,y) \vee q(x,g(x))$ is obtained where g is a new Skolem function symbol. Now, the name *quantorial extension* (and *functional extension* in case of skolemization) becomes clear; the $\exists$-quantifier (or the Skolem function symbol) extends (the language of) the formula.

Definition 1. Let $A^- = \forall x_1 \ldots \forall x_m\, [\forall \mathbf{u}\, B_1 \vee \forall \mathbf{v}\, B_2]$ be a negative occurrence of a formula and let $A^+ = \exists x_1 \ldots \exists x_m\, [\exists \mathbf{u}\, B_1 \wedge \exists \mathbf{v}\, B_2]$ be a positive occurrence of a formula. Let $\mathbf{u}$ denote the free variables occurring only in B_1 and let $\mathbf{v}$ denote the free variables occurring only in B_2. Let $\{y_1, \ldots, y_k\} \subseteq \{x_1, \ldots, x_m\}$, $\{z_1, \ldots, z_l\} = \{x_1, \ldots, x_m\} \setminus \{y_1, \ldots, y_k\}$, $\mathsf{Q}_i \in \{\forall, \exists\}$, and let Q_i^d be the quantifier dual to Q_i $(1 \leq k \leq m, 1 \leq i \leq k)$. Then

$$\forall z_1 \ldots \forall z_l\, [\mathsf{Q}_1 y_1 \ldots \mathsf{Q}_k y_k \forall \mathbf{u}\, B_1 \vee \mathsf{Q}_1^d y_1 \ldots \mathsf{Q}_k^d y_k \forall \mathbf{v}\, B_2] \tag{5}$$

$$\exists z_1 \ldots \exists z_l\, [\mathsf{Q}_1 y_1 \ldots \mathsf{Q}_k y_k \exists \mathbf{u}\, B_1 \wedge \mathsf{Q}_1^d y_1 \ldots \mathsf{Q}_k^d y_k \exists \mathbf{v}\, B_2] \tag{6}$$

are called *quantorial extensions* of A^- and A^+, respectively. If the strong quantifiers (i.e., negative occurrences of $\exists$ and positive occurrences of $\forall$) are removed by skolemization, then we call the result an *F-extension* of A^- and A^+, respectively.[8]

Quantifier extension is considered as an extension of calculi used for automated proof search. Since such calculi usually rely on a reduced syntax type (e.g., conjunctive normal form), F-extension is usually defined for clauses only, and skolemization is used in order to remove the newly introduced $\exists$-quantifiers.

Since $(\forall\, (B_1 \vee B_2)) \rightarrow (5)$ and $(6) \rightarrow \exists\, (B_1 \wedge B_2)$, quantorial extension (and F-extension) can be considered as restricted variants of the cut rule. Results how F-extension can speed-up proofs can be found in [5, 12, 6, 14]. We now come back to symmetric simplification and show how some kinds of quantifier extension can be "simulated" by symmetric simplification.

In what follows, we assume that we have an initial proof whose structure should be improved by applying quantorial extension. Let us reconsider (2) from Section 5. There, we used antiprenexing to derive $(\forall x.p(x)) \rightarrow \forall y.p(y)$ in order to simplify the proof. As an alternate, the lemma $\forall x.p(x) \vee \neg\forall x.p(x)$ can be chosen and (2) can be proven easily as follows.

$$\dfrac{\forall x.p(x) \vee \neg\forall x.p(x) \quad \dfrac{\dfrac{\dfrac{\dfrac{[\forall x.p(x)]_1}{p(u)}\,\forall\mathrm{E} \quad [p(t)]_3}{p(t) \rightarrow p(u)}\,{\rightarrow}\mathrm{I}_3}{\forall y.(p(t) \rightarrow p(y))}\,\forall\mathrm{I}}{\exists x\, \forall y.(p(x) \rightarrow p(y))}\,\exists\mathrm{I} \quad \alpha}{\exists x\, \forall y.(p(x) \rightarrow p(y))}\,\vee\mathrm{E}_{1,2}$$

α is as follows.[9]

[8] Strictly speaking, the whole formula has to be considered as the input for skolemization, because one has to assure that *globally* new Skolem function symbols are introduced.

[9] We use derived rules $\neg\exists$E, $\neg\forall$E, and $\neg{\rightarrow}$E in order to keep the example manageable.

$$\dfrac{\dfrac{\dfrac{[\neg(\exists x\ \forall y.(p(x)\ \rightarrow\ p(y)))]_4}{\neg(\forall y.(p(u)\ \rightarrow\ p(y)))}\ \neg\exists\mathrm{E} \qquad \dfrac{[\neg(p(u)\ \rightarrow\ p(v))]_5}{p(u)}\ \neg\rightarrow\mathrm{E}}{\dfrac{p(u)}{\forall x.p(x)}\ \forall\mathrm{I}}\ \neg\forall\mathrm{E}_5}{\dfrac{\dfrac{[\neg\forall x.p(x)]_2 \qquad \dfrac{p(u)}{\forall x.p(x)}\ \forall\mathrm{I}}{\bot}\ \neg\mathrm{E}}{\neg\neg\exists x\ \forall y.(p(x)\ \rightarrow\ p(y))}\ \neg\mathrm{I}_4}{\exists x\ \forall y.(p(x)\ \rightarrow\ p(y))}\ \neg\neg\mathrm{E}$$

If we apply quantifier extension to $\forall x.(p(x) \vee \neg p(x))$, we derive $\forall x.p(x) \vee \exists x.\neg p(x)$. Using this formula in the $\vee$E-inference, α becomes even simpler and no derived rule has to be used.

$$\dfrac{\dfrac{\dfrac{\dfrac{[\neg p(u)]_4 \qquad [p(u)]_5}{\bot}\ \neg\mathrm{E}}{p(v)}\ \bot\mathrm{E}}{p(u)\ \rightarrow\ p(v)}\ \rightarrow\mathrm{I}_5}{\forall y.(p(u)\ \rightarrow\ p(y))}\ \forall\mathrm{I}}{\dfrac{[\exists x.\neg p(x)]_2 \qquad \exists x\ \forall y.(p(x)\ \rightarrow\ p(y))}{\exists x\ \forall y.(p(x)\ \rightarrow\ p(y))}\ \exists\mathrm{E}_4}\ \exists\mathrm{I}$$

In [18], *logical flow graphs* [9] are used to decide whether an application of quantorial extension (in an LK-derivation) yields a correct derivation of the same end sequent. Usually, proofs with quantorial extensions tend to be shorter and have more structure, because a cut is introduced.

7 Conclusion

We have shown that many AD techniques can be considered as applications of the cut rule. Usually, these cuts are not part of an automatically generated proof because computationally advantageous calculi like resolution or connection calculi rely on reduced syntax classes like conjunctive normal form and simple inference rules. All these calculi are basically cut-free and, therefore, the use of essential lemmata is restricted. In contrast, for a presentation of an automatically generated proof, cuts are highly desirable, because their intelligent application shortens and structures proofs.

We identified one source of "unintuitive" proofs in current translations, namely if different instances of a subformula are required in a proof, and showed how to improve the translation in order to avoid these parts in the resulting proof. Another solution, symmetric simplification, uses the unintuitive proofs and tries to replace a proof by contradiction by a proof with *tertium non datur*. The relation of the latter technique to the cut rule and to quantorial extension is explained in detail. It turns out that symmetric simplification is a weak form of cut introduction. Moreover, some forms of quantorial extension are closely related to symmetric simplification.

References

1. P. B. Andrews. Resolution in Type Theory. *J. Symbolic Logic*, 36:414–432, 1971.
2. P. B. Andrews. Transforming Matings into Natural Deduction Proofs. In W.Bibel and R. Kowalski, editors, *Proceedings of the 5^{th} Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 281–292. Springer Verlag, 1980.
3. P. B. Andrews. Theorem Proving via General Matings. *Journal of the ACM*, 28(2):193–214, 1981.
4. M. Baaz, C. Fermüller, and A. Leitsch. A Non-Elementary Speed Up in Proof Length by Structural Clause Form Transformation. In *Proceedings of the Logic in Computer Science Conference*, pages 213–219, 1994.
5. M. Baaz and A. Leitsch. Complexity of Resolution Proofs and Function Introduction. *Annals of Pure and Applied Logic*, 57:181–215, 1992.
6. M. Baaz and A. Leitsch. On Skolemization and Proof Complexity. *Fundamenta Informaticae*, 20:353–379, 1994.
7. W. Bibel. *Automated Theorem Proving.* Vieweg, Braunschweig, second edition, 1987.
8. W. Bibel. *Deduction: Automated Logic.* Academic Press, London, 1993.
9. S. R. Buss. The Undecidability of k-Provability. *Annals of Pure and Applied Logic*, 53:75–102, 1991.
10. C. L. Chang and R. C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.
11. E. Eder. *Relative Complexities of First Order Calculi.* Vieweg, Braunschweig, 1992.
12. U. Egly. Shortening Proofs by Quantifier Introduction. In A. Voronkov, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, pages 148–159. Springer Verlag, 1992.
13. U. Egly. On Different Concepts of Function Introduction. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Proceedings of the Kurt Gödel Colloquium*, pages 172–183. Springer Verlag, 1993.
14. U. Egly. *On Methods of Function Introduction and Related Concepts.* PhD thesis, TH Darmstadt, Alexanderstr. 10, D–64283 Darmstadt, 1994.
15. U. Egly. On the Value of Antiprenexing. In F. Pfenning, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, pages 69–83. Springer Verlag, 1994.
16. U. Egly. On Different Structure-preserving Translations to Normal Form. *J. Symbolic Computation*, 22:121–142, 1996.
17. A. P. Felty. Using Extended Tactics to do Proof Transformations. Technical Report MS-CIS-86-89 LINC LAB 48, Department of Computer and Information Science, Moore School, University of Pennsylvania, Philadelphia, PA 19104, 1986.
18. K. Genther. Repräsentation von Konnektionsbeweisen in Gentzen-Kalkülen durch Transformation und Strukturierung. Master's thesis, TH Darmstadt, 1995.
19. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation: "Investigations into Logical Deduction" in [27], pp. 68-131.
20. J. Goubault. A BDD-Based Simplification and Skolemization Procedure. *J. of the IGPL*, 3(6):827–855, 1995.
21. C. Lingenfelder. Transformation and Structuring of Computer Generated Proofs. Technical Report SR-90-26, Universität Kaiserslautern, 1990.

22. D. Miller and A. Felty. An Integration of Resolution and Natural Deduction Theorem Proving. In T. Kehler, S. Rosenschein, R. Folman, and P. F. Patel-Schneider, editors, *Proceedings of the 5^{th} AAAI National Conference on Artificial Intelligence*, pages 198–202. Morgan Kaufmann Publishers, 1986.
23. D. A. Miller. Proofs in Higher-Order Logic. Technical Report MS-CIS-83-37, Department of Computer and Information Science, Moore School, University of Pennsylvania, Philadelphia, PA 19104, 1983.
24. D. A. Miller. Expansion Tree Proofs and their Conversion to Natural Deduction Proofs. In R. E. Shostak, editor, *Proceedings of the 7^{th} Conference on Automated Deduction*, volume 170 of *Lecture Notes in Computer Science*, pages 375–393. Springer Verlag, 1984.
25. F. Pfenning and D. Nesmith. Presenting Intuitive Deductions via Symmetric Simplification. In M. E. Stickel, editor, *Proceedings of the 10^{th} Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 226–350. Springer Verlag, 1990.
26. R. Statman. Lower Bounds on Herbrand's Theorem. In *Proc. AMS 75*, pages 104–107, 1979.
27. M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. North–Holland, Amsterdam, 1969.

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