Simultaneous Quantifier Elimination

Serge Autexier[1], Heiko Mantel[2], and Werner Stephan[2]

[1] Universität des Saarlandes, FB Informatik
Postfach 15 11 50, 66041 Saarbrücken, Germany
autexier@ags.uni-sb.de
[2] German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{mantel,stephan}@dfki.de

Abstract. We present a sequent calculus which allows the simultaneous elimination of multiple quantifiers. The approach is an improvement over the well-known skolemization in sequent calculus. It allows a lazy handling of instantiations and of the order of certain reductions. Simultaneous quantifier elimination is justified from a semantical as well as from a proof theoretical point of view.

1 Introduction

Sequent calculi are a very common search space representation. Originally developed by Gentzen [6] they have been applied in automated deduction, in logic programming, in formal program development, and other areas. During analytic proof search formulas in a sequent are decomposed into sub-formulas in a stepwise manner. The structure of sub-formulas and of formulas which are not decomposed is preserved. The preservation of structure is especially beneficial when user interaction is required. A user can recognize structures which e.g. in the context of formal methods [7] originate from a specification.

The relation between standard presentations of Hilbert type, natural deduction, and sequent calculi has been investigated by Avron [2] for the propositional case. The additional structure in sequent calculi usually provides advantages in proof search. In the presence of quantifiers additional differences between these type of calculi arise. Gentzen's rules for the elimination of quantifiers employ an eager handling of instantiations. This causes a high-degree of non-determinism in proof search which can be avoided by a lazy handling of instantiations with meta-variables together with a computation of instantiations by unification. *Skolemization* [13] is a well-known technique which guarantees that proofs constructed with a lazy handling of instantiations can be validated in general. In the context of sequent calculi, skolemization has been investigated for classical [4] as well as for non-classical logics [12, 9].

The technique for simultaneous quantifier elimination presented in this article is specific to sequent calculi. It provides an optimization over the usual approach for lazy handling of instantiations. The two justifications of its soundness yield different insights in the dependencies between formulas of a sequent in the presence of quantifiers.

After some fundamentals we present in section 3 a sequent calculus $\mathcal{K}$ with a rule for simultaneous quantifier elimination. We point out its advantages in comparison to usual handling of quantifiers in sequent calculus proof search. The soundness of $\mathcal{K}$ is demonstrated in section 4 using semantical and in section 5 using syntactical arguments. We conclude with some remarks on related work.

2 Fundamentals

Basing on [10], we define syntax and semantics of first-order logic. A *signature* Σ is a pair $(\mathcal{F}, \mathcal{P})$ consisting of a set $\mathcal{F}$ of operation symbols and a set $\mathcal{P}$ of predicate symbols. Each $f \in \mathcal{F}$ has an arity $n_f \in \mathbb{N}$ and each $p \in \mathcal{P}$ has an arity $n_p \in \mathbb{N}$. A *Σ-algebra A* has a *carrier set S_A* and assigns to each n_f-ary operation $f \in \mathcal{F}$ a total function $A(f) : (S_A)^{n_f} \to S_A$ and to each predicate $p \in \mathcal{P}$ a n_p-ary relation $A(p) \subseteq (S_A)^{n_p}$. *Constants* are 0-ary operations.

Syntax of First-Order Logic. The set $T_\Sigma(\mathcal{V})$ of *first-order terms* for a signature Σ and a set $\mathcal{V}$ of variables is defined recursively. For each $x \in \mathcal{V}$ holds $x \in T_\Sigma(\mathcal{V})$. If $t_1, \ldots, t_{n_f} \in T_\Sigma(\mathcal{V})$ then for any n_f-ary operation $f \in \mathcal{F}$ holds $f(t_1, \ldots, t_{n_f}) \in T_\Sigma(\mathcal{V})$. The set $\mathit{wff}(\Sigma, \mathcal{V})$ of *first-order formulas* for $\Sigma = (\mathcal{F}, \mathcal{P})$ and $\mathcal{V}$ is defined recursively. For $t_1, \ldots, t_{n_p} \in T_\Sigma(\mathcal{V})$ and $p \in \mathcal{P}$ with arity n_p the expression $p(t_1, \ldots, t_{n_p})$ is an *atomic formula* in $\mathit{wff}(\Sigma, \mathcal{V})$. If $\varphi, \psi \in \mathit{wff}(\Sigma, \mathcal{V})$ and $x \in \mathcal{V}$ then $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \forall x.\varphi, \exists x.\varphi \in \mathit{wff}(\Sigma, \mathcal{V})$ are formulas.

For a term t the function *Var* returns the variables and *Op* the operations which occur in t. The function *free* which returns the free variables of a formula is defined recursively over the structure of formulas, i.e. $\mathit{free}(p(t_1, \ldots, t_{n_p})) = \bigcup_{i=1}^{n_p} \mathit{Var}(t_i)$, $\mathit{free}(\neg\varphi) = \mathit{free}(\varphi)$, $\mathit{free}(\varphi \overset{\wedge}{\vee} \psi) = \mathit{free}(\varphi) \cup \mathit{free}(\psi)$, and $\mathit{free}(\overset{\forall}{\exists} x.\varphi) = \mathit{free}(\varphi) \setminus \{x\}$. *Op* returns for φ the operations which occur in φ.

Semantics of First-Order Logic. The *value* $A(\alpha)(t)$ of a term $t \in T_\Sigma(\mathcal{V})$ and the *value* $A(\alpha)(\varphi)$ of a formula $\varphi \in \mathit{wff}(\Sigma, \mathcal{V})$ for a Σ-algebra A and an assignment $\alpha : \mathcal{V} \to S_A$ where $\mathit{free}(\varphi) \subseteq \mathcal{V}$ is respectively an element of the carrier set S_A or a truth value (*true* or *false*).

- $A(\alpha)(x) = \alpha(x)$ for $x \in \mathcal{V}$,
- $A(\alpha)(f(t_1, \ldots, t_{n_f})) = A(f)(A(\alpha)(t_1), \ldots, A(\alpha)(t_{n_f}))$,
- $A(\alpha)(p(t_1, \ldots, t_{n_p})) = \mathit{true}$ iff $(A(\alpha)(t_1), \ldots, A(\alpha)(t_{n_p})) \in A(p)$,
- $A(\alpha)(\neg\varphi) = \mathit{true}$ iff $A(\alpha)(\varphi) = \mathit{false}$,
- $A(\alpha)(\varphi_1 \overset{\wedge}{\vee} \varphi_2) = \mathit{true}$ iff $A(\alpha)(\varphi_1) = \mathit{true} \overset{\text{and}}{\text{or}} A(\alpha)(\varphi_2) = \mathit{true}$,
- $A(\alpha)(\overset{\forall}{\exists} x.\varphi) = \mathit{true}$ iff $(A(\alpha[a/x])(\varphi) = \mathit{true} \overset{\text{for all}}{\text{for some}} a \in A(s))$.

where $\alpha[a/x]$ is the assignment: $\alpha[a/x](x) = a$ and $\alpha[a/x](y) = \alpha(y)$, if $y \neq x$. A formula φ is *valid in a Σ-algebra A* ($A \models_\Sigma \varphi$) iff for any assignment α holds $A(\alpha)(\varphi) = \mathit{true}$. A formula φ is *valid* ($\models_\Sigma \varphi$) iff it is valid in every Σ-algebra.

Substitutions. Let Σ be a signature and $\mathcal{V}$ be a set of variables for Σ. A function $\sigma : \mathcal{V} \to T_{\Sigma(\mathcal{V})}$ is called a *substitution*. The application of a substitution to a formula $\varphi \in \mathit{wff}(\Sigma, \mathcal{V})$ yields a formula $\sigma(\varphi)$, where all free occurrences of variables $x \in \mathcal{V}$ are replaced by $\sigma(x)$. If σ is the identity except for a finite number of variables $x_1, \ldots, x_n$, we denote σ by $[\sigma(x_1)/x_1, \ldots, \sigma(x_n)/x_n]$. $\mathit{Dom}(\sigma) = \{x_1, \ldots, x_n\}$ is called the *domain* of σ. A substitution σ is *admissible* for φ if for every sub-formula $Qx.\varphi'$ of φ holds $x \notin \sigma(y)$ for all $y \in \mathit{free}(Qx.\varphi')$. We require substitutions to be idempotent and admissible.

The following theorem states a fundamental relationship between substitutions and assignments. For a proof we refer the interested reader to [10].

Theorem 1 (Substitution Theorem). *Let $\mathcal{V}$ be a set of variables for a signature Σ, $\sigma : \mathcal{V} \to T_{\Sigma(\mathcal{V})}$ a substitution, A a Σ-algebra, and $\beta : \mathcal{V} \to S_A$ an assignment. Then for every $t \in T_{\Sigma(\mathcal{V})}$ holds $A(\beta)(\sigma(t)) = A(\alpha)(t)$, where $\alpha : \mathcal{V} \to S_A$ is an assignment defined by $\alpha(x) := A(\beta)(\sigma(x))$ for every $x \in \mathcal{V}$.*

We restrict ourselves throughout this article to formulas in *negation-normal form*, i.e. formulas where negation $\neg$ occurs only directly in front of atomic formulas. Using the de-Morgan laws any first-order formula can be transformed into an equivalent formula which is in negation normal form.

Sequents. A *(one-sided) sequent* s is a set Γ of formulas in negation-normal form denoted by $\longrightarrow \Gamma$. We define $\mathit{free}(\longrightarrow \Gamma) = \bigcup_{\varphi \in \Gamma} \mathit{free}(\varphi)$. Given an algebra A and an assignment $\alpha : \mathcal{V} \to S_A$ with $\mathit{free}(s) \subseteq \mathcal{V}$. The *value* $A(\alpha)(s)$ is *true* iff $A(\alpha)(\varphi) = \mathit{true}$ for some $\varphi \in \Gamma$. s is *valid in* an algebra A ($A \models_\Sigma s$) if for all assignments α $A(\alpha)(s) = \mathit{true}$. s is *valid* ($\models_\Sigma s$) if it is valid in all algebras.

A *sequent calculus* is a pair $\langle Ax, \mathit{Inf} \rangle$. Ax is a finite set of axiom schemes each of which is a decidable set of sequents. *Inf* is a finite set of inference rules. Each inference rule consists of a decidable set of pairs $(s_1, \ldots, s_n), s$ where $s_1, \ldots, s_n$ and s are sequents. s is called the *conclusion* and $s_1, \ldots, s_n$ the *premises* of the inference rule. A *principal formula* is a formula that occurs in the conclusion but not in any premise. Formulas which occur in a premise but not in the conclusion are called *side formulas*. All other formulas compose the *context*. Sequent rules can be represented graphically where the conclusion is written underneath the premises and separated from them by a horizontal line. A *derivation* of a sequent s from a set of sequents S is a finite sequence of sequents $s_1, \ldots, s_k$ with $k \geq 1$ and $s_k = s$ such that for each $i \leq k$ holds $s_i \in S$, s_i is an axiom in Ax, or there exist indices $i_1, \ldots, i_n$ such that there is an inference rule in *Inf* with conclusion s_i and premises $s_{i_1}, \ldots, s_{i_n}$. A sequent s is said to be *derivable* from a set of sequents S ($S \vdash s$) if there exists a derivation from S for it.

The one-sided sequent calculus $\mathcal{K}_c$ for formulas in negation normal form[1] is:

$$\frac{}{\longrightarrow \Gamma, \varphi, \neg\varphi}\,ax \qquad \frac{\longrightarrow \Gamma, \varphi_1 \quad \longrightarrow \Gamma, \varphi_2}{\longrightarrow \Gamma, \varphi_1 \wedge \varphi_2}\,\wedge \qquad \frac{\longrightarrow \Gamma, \varphi_1, \varphi_2}{\longrightarrow \Gamma, \varphi_1 \vee \varphi_2}\,\vee \qquad \frac{\longrightarrow \Gamma, \varphi[c/x]}{\longrightarrow \Gamma, \forall x.\varphi}\,\forall^{*} \qquad \frac{\longrightarrow \Gamma, \varphi[t/x]}{\longrightarrow \Gamma, \exists x.\varphi}\,\exists^{**}$$

$*$ c must not occur in $\longrightarrow \Gamma, \forall x.\varphi$ (*Eigenvariable condition*). $**$ t may be any term.

In *analytic proof search* with $\mathcal{K}_c$ one starts with the sequent to be proven and reduces it by application of rules until the *ax*-rule is applicable.

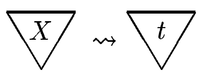
3 Simultaneous Quantifier Elimination

The quantifier rules of $\mathcal{K}_c$ cause problems in analytic proof search. Whenever the $\exists$-rule is applied a term t must be guessed immediately. To postpone the choice of t until more information about good choices of t are at hand is a superior approach. In order to do so the rule $\exists'$ depicted below inserts a free variable

[1] The restriction to formulas in negation-normal form and to one-sided sequents has only presentational purposes. The theory presented in this article could also be developed for arbitrary formulas and two-sided sequents.

X (sometimes also called *meta-variable*) which is implicitly existentially quantified. Thus, it may be instantiated later during proof search. However, precautions must be taken to guarantee the correctness of the resulting proofs because not all possible instantiations are admissible. *Skolemization* is used for this purpose. The rule *Skolem* inserts a *skolem-term* consisting of a new function symbol with all free variables of the sequent as arguments. Free variables may be instantiated during proof search. The instantiation of a variable affects all parts of a derivation where the variable occurs, i.e. *Inst* is a rewrite rule on derivations rather than an ordinary sequent rule. The *occur-check* ensures that a variable X can only be substituted by terms t which do not contain X.

$$\frac{\longrightarrow \Gamma, \varphi[X/x]}{\longrightarrow \Gamma, \exists x.\varphi}\ \exists' \qquad \frac{\longrightarrow \Gamma, \varphi[f(Z)/x]}{\longrightarrow \Gamma, \forall x.\varphi}\ \mathit{Skolem}^{*}$$



$\mathit{Inst}(X,t)^{**}$

* f must not occur in $s = \longrightarrow \Gamma, \forall x.\varphi$ and Z must contain all free variables of s.
** X must not occur in t and all variables and operations in t must also occur in the left-hand side proof-tree.

The calculus $\mathcal{K}_{sk}$ results from $\mathcal{K}_c$ by adding the rules $\exists'$, *Skolem*, and *Inst* while the rules $\exists$ and $\forall$ are removed.

The use of free variables and skolemization allows to postpone the instantiation until it can be computed, e.g. by unification. Nevertheless, if multiple quantified formulas occur in a sequent a principal formula must be determined. Although in some cases a principal formula can be chosen in a safe way, in general, the right order of reductions cannot be calculated from a sequent. This is demonstrated by the following example.

Example 1. Below a $\mathcal{K}_{sk}$-derivation with six rule applications is depicted.

$$\begin{array}{cl}
\longrightarrow \varphi(X_1,f_1(X_1),Z_1),\neg\varphi(Z_2,X_2,f_2(X_1,X_2)) & \\ \hline
\longrightarrow \varphi(X_1,f_1(X_1),Z_1),\exists z_2.\neg\varphi(z_2,X_2,f_2(X_1,X_2)) & \exists' \\ \hline
\longrightarrow \exists z_1.\varphi(X_1,f_1(X_1),z_1),\exists z_2.\neg\varphi(z_2,X_2,f_2(X_1,X_2)) & \exists' \\ \hline
\longrightarrow \exists z_1.\varphi(X_1,f_1(X_1),z_1),\forall y_2.\exists z_2.\neg\varphi(z_2,X_2,y_2) & \mathit{Skolem} \\ \hline
\longrightarrow \exists z_1.\varphi(X_1,f_1(X_1),z_1),\exists x_2.\forall y_2.\exists z_2.\neg\varphi(z_2,x_2,y_2) & \exists' \\ \hline
\longrightarrow \forall y_1.\exists z_1.\varphi(X_1,y_1,z_1),\exists x_2.\forall y_2.\exists z_2.\neg\varphi(z_2,x_2,y_2) & \mathit{Skolem} \\ \hline
\longrightarrow \exists x_1.\forall y_1.\exists z_1.\varphi(x_1,y_1,z_1),\exists x_2.\forall y_2.\exists z_2.\neg\varphi(z_2,x_2,y_2) & \exists'
\end{array}$$

The proof attempt would have failed if we first had reduced the second formula.

A Rule for Simultaneous Quantifier Elimination. Simultaneous quantifier elimination is based on skolemization. However, it is superior since it allows a lazy handling of both instantiations and reduction orderings on quantified formulas.

We define *quantifier lists ql* recursively starting from the empty list ϵ and for a variable x by $\forall x$.ql' and $\exists x$.ql'. In order to simplify the following definition we assume generators *vgen* and *fgen* which respectively generate new symbols for variables and operations on every call.

We define the *quantifier elimination function* QE which takes a quantifier list ql, a formula φ, and a set Z of variables as arguments and returns a formula. ql determines which quantifiers shall be eliminated from φ. Z is used in order to determine the arguments of skolem functions.

- QE$(\epsilon, \varphi, Z) := \varphi$,
- QE$(\forall x.\mathrm{ql}, \varphi, Z) :=$ QE$(\mathrm{ql}, \varphi[f(Z)/x], Z)$, where $f :=$ *fgen* is new.
- QE$(\exists x.\mathrm{ql}, \varphi, Z) :=$ QE$(\mathrm{ql}, \varphi[X/x], Z \cup \{X\}$ where $X :=$ *vgen* is new.

The rule *SQEl* for *simultaneous quantifier elimination* is depicted below.

$$\frac{\longrightarrow \Gamma, \psi_1, \ldots, \psi_n}{\longrightarrow \Gamma, \mathrm{ql}_1 . \varphi_1, \ldots, \mathrm{ql}_n . \varphi_n} \; \mathit{SQEl}^*$$

$*$ For each i $(1 \leq i \leq n)$ must hold $\psi_i = \mathrm{QE}(\mathrm{ql}_i, \varphi_i, \mathit{free}(\longrightarrow \Gamma, \mathrm{ql}_1 . \varphi_1, \ldots, \mathrm{ql}_n . \varphi_n))$.

The calculus $\mathcal{K}$ results from $\mathcal{K}_{sk}$ by replacing the rules $\exists'$ and *Skolem* by *SQEl* . $\mathcal{K}$ is complete with respect to $\mathcal{K}_{sk}$, i.e. for every sequent s which is $\mathcal{K}_{sk}$-derivable there is a $\mathcal{K}$-derivation, since $\exists'$ and *Skolem* can be simulated by *SQEl*. However, *SQEl* has advantages compared to these rules because one does not need to bother about the order of certain reductions.

Example 2. We reduce our example sequent by *SQEl*.

$$\frac{\longrightarrow \varphi(X_1, f_1(X_1), Z_1), \neg\varphi(Z_2, X_2, f_2(X_2))}{\longrightarrow \exists x_1 . \forall y_1 . \exists z_1 . \varphi(x_1, y_1, z_1), \exists x_2 . \forall y_2 . \exists z_2 . \neg\varphi(z_2, x_2, y_2)} \; \mathit{SQEl}$$

A comparison to example 1 shows the advantages of simultaneous quantifier elimination. First, one does not need to worry about the order of quantifier eliminations. Second, the skolem term in the second formula depends only on X_2 and not on both X_1 and X_2 as in example 1. This shows that the quantifier elimination of different formulas in a sequent do not depend on each other.

Remark 1. In the Isabelle system [11] for example a dual technique to skolemization is employed. According to this technique a universally quantified formula $\forall x . \varphi(x)$ is reduced to $\bigwedge x . \varphi(x)$ and an existentially quantified formula $\exists x . \varphi(x)$ to $\varphi(?x)$ where $\bigwedge$ is a meta-logic quantifier and $?x$ is a (higher-order) meta-variable. Due to lifting over quantifiers meta-variables receive arguments which essentially determine which constants may be used in instantiations. This causes close interdependencies between formulas in a sequent which are not present when skolemization is applied. Before a constant may be instantiated, i.e. appear as an argument of a meta-variable, the corresponding quantifier must have been reduced already. Therefore, it appears to be quite difficult to develop an optimized handling of quantifiers equivalent to *SQEl* for this technique. For details of the technique we refer the interested reader to [11].

4 Semantical Justification

In this section we present a correctness proof for $\mathcal{K}$ using semantical arguments. For this purpose an auxiliary calculus $\mathcal{K}_{aux}$ is defined which allows to reason about sequents with substitutions. The explicitly stated substitutions are used for meta-level arguments only. We prove the soundness of $\mathcal{K}_{aux}$ and then conclude the soundness of $\mathcal{K}$ from that. In the process we introduce orderings on constants and variables, an approach which is motivated by orderings on positions in the context of matrix characterizations.[3, 14]

Sequents with Substitutions. A *sequent with substitution* s is a pair $\longrightarrow \Gamma; \sigma$ consisting of a sequent $\longrightarrow \Gamma$ and a substitution σ. We define $\mathit{free}(s) = \bigcup_{\varphi \in \Gamma} \mathit{free}(\sigma(\varphi))$. The *value* $A(\alpha)(s)$ is *true* in an algebra A under an assignment $\alpha : \mathcal{V} \to S_A$ where $\mathit{free}(s) \subseteq \mathcal{V}$ iff $A(\alpha)(\longrightarrow \sigma(\Gamma)) = \mathit{true}$.

Below, we define the *auxiliary quantifier elimination function* QE_{aux} which takes a quantifier list ql, a formula φ, a set O of constants and variables, and a binary relation $\ll$ over O as arguments. ql determines which quantified variables shall be eliminated from φ. In O the set of all constants and variables introduced during the elimination are collected while a relation over these symbols is collected in $\ll$. QE_{aux} returns a triple consisting of a formula φ', a set O' of constants and variables, and an ordering $\ll'$ over O'.

- $\mathrm{QE}_{aux}(\epsilon, \varphi, O, \ll) := \langle \varphi, O, \ll \rangle$,
- $\mathrm{QE}_{aux}(\forall x.\mathrm{ql}, \varphi, O, \ll) := \mathrm{QE}_{aux}(\mathrm{ql}, \varphi[c/x], O \cup \{c\}, \ll \cup \{(o,c) \mid \forall o \in O\})$,
  where $c := \mathit{fgen}$ is new.
- $\mathrm{QE}_{aux}(\exists x.\mathrm{ql}, \varphi, O, \ll) := \mathrm{QE}_{aux}(\mathrm{ql}, \varphi[X/x], O \cup \{X\}, \ll \cup \{(o,X) \mid \forall o \in O\})$
  where $X := \mathit{vgen}$ is new.

Example 3. With the appropriate symbols generated by *vgen* and *fgen* the value of $(\forall x_1 \exists y_1 \forall z_1, \varphi(x_1, y_1, z_1), \emptyset, \emptyset)$ under QE_{aux} is

$$\langle \varphi(X_1, c_1, Z_1), \{X_1, c_1, Z_1\}, \{(X_1, c_1), (X_1, Z_1), (c_1, Z_1)\} \rangle.$$

Orderings on Constants and Variables. QE_{aux} eliminates quantifiers in the order in which they occur in ql. This order is represented by the relation $\ll$ on the variables and constants introduced during elimination which is returned by QE_{aux}. Clearly, $\ll$ is an ordering, the *quantifier list ordering.*

For a set of variables $\mathcal{V}$, a set of constants $\mathcal{C}$, and a substitution σ we define two relations $\sim \subseteq \mathcal{V} \times \mathcal{V}$ and $\sqsubset \subseteq (\mathcal{V} \cup \mathcal{C}) \times \mathcal{V}$ as the minimal relations such that:

- for any $u, v \in \mathcal{V}$ if $\sigma(u) = v$ then $u \sim v$,
- for any $u \in \mathcal{V}$ and $v \in \mathcal{C} \cup \mathcal{V}$ if v occurs in $\sigma(u)$ and $\sigma(u) \neq v$ then $v \sqsubset u$,
- and for any $u, v \in \mathcal{V}$ if $v \sqsubset u$ and $u \sim u'$ then $v \sqsubset u'$.

We combine given orderings $\sqsubset$ and $\ll$ to a relation $\lhd \subseteq (\mathcal{C} \cup \mathcal{V})^2$, i.e. $\lhd = (\sqsubset \cup \ll)^+$ where $^+$ denotes the transitive closure. Indicating that $\lhd$ is intended to be usually irreflexive we call it a *reduction ordering.* If $\lhd$ is a reduction ordering over some set O of variables and constants and $O' \subseteq O$, then the *restriction* of $\lhd$ to O' is defined by $\lhd_{O'} := \{(o, o') \mid (o, o') \in \lhd \text{ and } o, o' \in O'\}$.

Calculus for Sequents with Substitutions. The auxiliary calculus $\mathcal{K}_{aux}$ is depicted below. Substitutions are explicitly stated and do not change in a $\mathcal{K}_{aux}$-proof. This is not problematic since we use the calculus only to reason about its soundness but not for proof search. Note, that in contrast to $\mathcal{K}$ in $\mathcal{K}_{aux}$ no skolem-terms are introduced during quantifier elimination.

$$\frac{}{\longrightarrow \Gamma, \varphi, \neg\varphi; \sigma}\,ax \qquad \frac{\longrightarrow \Gamma, \varphi_1; \sigma \quad \longrightarrow \Gamma, \varphi_2; \sigma}{\longrightarrow \Gamma, \varphi_1 \wedge \varphi_2; \sigma}\,\wedge \qquad \frac{\longrightarrow \Gamma, \psi_1, \ldots, \psi_n; \sigma}{\longrightarrow \Gamma, \mathrm{ql}_1 . \varphi_1, \ldots, \mathrm{ql}_n . \varphi_n; \sigma}\,\mathit{SQEl}^*_{aux} \qquad \frac{\longrightarrow \Gamma\sigma'; \sigma}{\longrightarrow \Gamma; \sigma}\,\mathit{Subst}^{**}$$

* For each $i \in \{1, \ldots, n\}$ holds $\langle \psi_i, O_i \ll_i \rangle = \mathrm{QE}_{aux}(ql_i, \varphi_i, \mathit{free}(\longrightarrow \Gamma, \psi_1, \ldots, \psi_n), \emptyset)$, $\sqsubset$ the ordering from σ, $\ll \; = \; (\bigcup_{i=1}^{n} \ll_i)$, $\lhd = (\sqsubset \cup \ll)^+$, $O = \bigcup_{i=1}^{n} O_i$ the set of (free) variables and constants occurring in the premise and $\lhd_O$ is an irreflexive ordering.
** Where $\sigma \circ \sigma' = \sigma$ holds.

Example 4. The orderings $\ll$, $\sqsubset$, and $\lhd_O$ for the following rule application are depicted in the diagram to the right. $\ll$ is symbolized by solid arrows and $\sqsubset$ by dashed arrows.

$$\frac{\longrightarrow \varphi(X_1, c_1, Z_1), \neg\varphi(X_2, c_2, Z_2); \{Z_2/X_1, c_1/X_2, c_2/Z_1\}}{\longrightarrow \exists x_1 . \forall y_1 . \exists z_1 . \varphi(x_1, y_1, z_1), \exists x_2 . \forall y_2 . \exists z_2 . \neg\varphi(x_2, y_2, z_2); \{Z_2/X_1, c_1/X_2, c_2/Z_1\}}\,\mathit{SQEl}$$

$$\begin{tikzcd} Z_1 & Z_2 \\ c_1 \arrow[u] \arrow[dr, dashed] & c_2 \arrow[u] \arrow[ul, dashed] \\ X_1 \arrow[u] & X_2 \arrow[u] \end{tikzcd}$$

Theorem 2 (Soundness). *If there exists a $\mathcal{K}_{\mathrm{aux}}$-proof $\mathcal{P}$ of a sequent with substitution s then s is valid.*

Proof. The proof is by induction on the structure of $\mathcal{P}$. The base case where $\mathcal{P}$ consists only of an application of *ax* is trivial. In the induction step a case distinction depending on the last rule application in $\mathcal{P}$ is made. We concentrate on the interesting cases where $\wedge$, *Subst*, or $SQEl_{aux}$ is applied. In each case we assume that all premises of the rule are valid and infer the validity of the conclusion. For a more detailed proof we refer the interested reader to [1].

- Let $\wedge$ be the last rule applied in $\mathcal{P}$. We assume that for every algebra A and every assignment α holds $A(\alpha)(\longrightarrow \Gamma, \varphi_1; \sigma) = \mathit{true} = A(\alpha)(\longrightarrow \Gamma, \varphi_2; \sigma)$. Let A be an arbitrary algebra and α be an arbitrary assignment. If there is a $F \in \Gamma$ such that $A(\alpha)(\sigma(F)) = \mathit{true}$, then $A(\alpha)(\longrightarrow \Gamma, \varphi_1 \wedge \varphi_2; \sigma) = \mathit{true}$ holds trivially. Otherwise, $A(\alpha)(\sigma(\varphi_1)) = \mathit{true} = A(\alpha)(\sigma(\varphi_2))$ must hold which implies $A(\alpha)(\longrightarrow \Gamma, \varphi_1 \wedge \varphi_2; \sigma) = \mathit{true}$.
- Let *Subst* be the last rule applied in $\mathcal{P}$. We assume that for every algebra A and every assignment α holds $A(\alpha)(\longrightarrow \sigma'(\Gamma); \sigma) = \mathit{true}$. According to the side condition of the rule holds $\sigma \circ \sigma' = \sigma$. Thus, $\longrightarrow \sigma(\Gamma) =\longrightarrow \sigma(\sigma'(\Gamma))$ and the validity of the conclusion follows.
- Let $SQEl_{aux}$ be the last rule applied in $\mathcal{P}$. The proof is done by induction over the number m of quantifiers eliminated, i.e. the sum of the lengths of the ql_i. The base case where $m = 0$ is trivial, since the premise and the conclusion are the same sequent. We first prove the case $m = 1$ with $s =\longrightarrow \Gamma, Qx.\varphi; \sigma$.
  - If $Q = \exists$, we assume that for every algebra A and every assignment α holds $A(\alpha)(\longrightarrow \Gamma, \varphi[X/x]; \sigma) = \mathit{true}$. Let A be an arbitrary algebra and α be an arbitrary assignment. The interesting case is where $A(\alpha)(\sigma(F)) = \mathit{false}$ for all $F \in \Gamma$ and $A(\alpha)(\sigma(\varphi[X/x])) = \mathit{true}$. Let σ' be the restriction of σ to $\mathit{free}(\Gamma \cup \{\exists x.\varphi\})$. Then
    $$\begin{aligned}\mathit{true} &= A(\alpha)(\sigma(\varphi[X/x])) = A(\alpha)(\sigma'(\varphi[X/x])) = A(\alpha)((\sigma'(\varphi))[\sigma'(X)/x])\\ &= A(\alpha[A(\alpha)(\sigma'(X))/x])(\sigma'(\varphi)) \text{ by substitution theorem}\\ &= A(\alpha)(\exists x.\sigma'(\varphi)) \text{ by definition of the semantics of } \exists\\ &= A(\alpha)(\sigma'(\exists x.\varphi)) = A(\alpha)(\sigma(\exists x.\varphi)) = A(\alpha)(\longrightarrow \Gamma, \exists x.\varphi; \sigma)\end{aligned}$$
  - If $Q = \forall$, we assume that for every algebra A and every assignment α holds $A(\alpha)(\longrightarrow \Gamma, \varphi[c/x]; \sigma) = \mathit{true}$. Let A be an arbitrary algebra and α be an arbitrary assignment. The interesting case is where $A(\alpha)(\sigma(F)) = \mathit{false}$ for all $F \in \Gamma$ and $A(\alpha)(\sigma(\varphi[c/x])) = \mathit{true}$. We consider all *variants* A_a of A, i.e. all algebras which differ from A only in the interpretation of c such that $A_a(c) = a$. The side-condition of $SQEl_{aux}$ and the definition of QE_{aux} ensure that $X \ll c$ holds for every $X \in \mathit{free}(\longrightarrow \Gamma, \forall x.\varphi; \sigma)$. Because $\lhd$ is required to be irreflexive for any $F \in \Gamma$, c does not occur in $\sigma(F)$ and thus, $A_a(\alpha)(\sigma(F)) = A(\alpha)(\sigma(F)) = \mathit{false}$. Let σ' be the restriction of σ to $\mathit{free}(\Gamma \cup \{\forall x.\varphi)\}$. Then for all A_a holds
    $$\begin{aligned}\mathit{true} &= A_a(\alpha)(\sigma'(\varphi[c/x])) = A_a(\alpha)((\sigma'(\varphi))[c/x]) \text{ since } \sigma' \text{ is admissible}\\ &= A_a(\alpha[A_a(c)/x])(\sigma'(\varphi)) \text{ by substitution theorem}\\ &= A_a(\alpha[a/x])(\sigma'(\varphi))\ .\end{aligned}$$
    For every $a \in S_A$ there is an A_a, thus $A(\alpha)(\longrightarrow \Gamma, \forall x.\varphi; \sigma) = \mathit{true}$.

  In the induction step we assume the soundness of $SQEl_{aux}$ for the elimination of less than m quantifiers ($m > 1$) and show the soundness for m quantifiers. The irreflexivity of $\lhd$ ensures that there is a maximal element

$o \in O$ with regard to $\lhd$. According to the definition of $\sqsubset$ o is not instantiated for any variable in O. Let o be introduced by the elimination of $Q_j x.\psi'_j$. We split the application of $SQEl_{aux}$ as follows into two applications of the rule where each of the applications reduces less than m quantifiers. Due to the choice of $Q_j x.\psi_j$ the side conditions for both rule applications are fulfilled.

$$\dfrac{\dfrac{\longrightarrow \Gamma, \psi_1, \ldots, \psi_j, \ldots, \psi_n ; \sigma}{\longrightarrow \Gamma, \psi_1, \ldots, Qx.\psi'_j, \ldots, \psi_n ; \sigma}\ SQEl_{aux}}{\longrightarrow \Gamma, \mathrm{ql}_1.\varphi_1, \ldots, \mathrm{ql}_j.Qx.\varphi_j, \ldots, \mathrm{ql}_n.\varphi_n ; \sigma}\ SQEl_{aux} \qquad \begin{array}{l} 1 \text{ quantifier elimination} \\ (m-1) \text{ quantifier eliminations} \end{array}$$

Remark 2. In the induction step of the above proof we show that it is always possible to focus on a single formula in a sequent. In the case where the rule $SQEl_{aux}$ is the last rule applied this is non-trivial because multiple formulas are reduced in a single rule application. Free variables in a sequent cause dependencies between formulas. Only the side condition of the rule $SQEl_{aux}$ allow us to single out a specific formula according to the reduction ordering and ensure in the case $\forall$ that this formula is valid in all variants of a specific algebra.

Theorem 3 (Soundness). *If there exists a $\mathcal{K}$-proof for a sequent then it is valid.*

Proof. There are three differences between $\mathcal{K}_{aux}$ and $\mathcal{K}$. In $\mathcal{K}_{aux}$ a substitution is explicitly stated in sequents, ***Eigenvariables*** have no arguments (i.e. are no skolem-terms), and the *Subst*-rule is a sequent rule while the *Inst*-rule of $\mathcal{K}$ is a rewrite rule on proof trees. Proof search in $\mathcal{K}_{aux}$ would require that an appropriate substitution is guessed before any rule may be applied. None of the rules in $\mathcal{K}_{aux}$ is capable to modify this substitution. This appears to be impractical, however, $\mathcal{K}_{aux}$ is only an auxiliary calculus. We argue that the skolemization based technique applied in $\mathcal{K}$ is a realization of the constraints imposed by $\mathcal{K}_{aux}$.

During proof search in $\mathcal{K}$ substitutions can only be applied globally to the proof tree using the *Inst*-rule. From all applications of *Inst* in a $\mathcal{K}$-proof $\mathcal{P}$ of a sequent s a substitution σ can be constructed such that a $\mathcal{K}_{aux}$-proof $\mathcal{P}_{aux}$ for $s; \sigma$ exists. $\mathcal{P}_{aux}$ can be constructed inductively from $\mathcal{P}$. An application of *Inst* in $\mathcal{P}$ results in an application of *Subst* on all open leaves of $\mathcal{P}_{aux}$. An application *SQEl* results in an application of $SQEl_{aux}$. Skolemization ensures that the side-condition of $SQEl_{aux}$ holds, i.e. $\lhd_O$ is irreflexive. If $\lhd_O$ would not be irreflexive, then by construction of $\lhd_O$ the substitution were not idempotent – a contradiction. Any other $\mathcal{K}$-rule is mapped to the respective $\mathcal{K}_{aux}$-rule.

5 Syntactical Justification

Since the early seventies systems have emerged that use interactive proof strategies based on complex user defined proof rules. The soundness of these systems is guaranteed by explicitly performing a proof in the basic calculus for each (application of a) derived rule (or tactic). This *proof theoretic* approach is simple in the sense that no additional *formal* concepts are required. All we need is a system architecture that keeps up a consistent state of the overall proof generation process and forces us to expand all derived steps.

We present a generalization of this approach where in a formalized meta-language we prove that a proof exists for all instances of a derived rule. As compared to tactical theorem proving here additional meta-logical concepts are required. However, the approach still is uniform in the sense that it relies on a fixed collection of formal concepts that are *syntactic* in nature. A main motivation of this paper is to compare this approach to a *semantic* justification which uses various concepts from model theory but which of course also could be formalized in a meta-level formalism.

The syntactic (or proof theoretic) approach seems to be particular useful for proof generation mechanisms that themselves use meta-level notions. For example, it provides a simple and clear semantics for meta-*variables* as placeholders for syntactic objects, like (object-level) variables, terms, and formulas. The approach is not limited to cases where the objects involved are first-class citizens with respect to (object-level) quantification nor is it restricted to meta-variables. In the context of the quantifier elimination rule also Skolem-functions are meta-level symbols.

The meta-level justification of quantifier elimination is *local* in the sense that we are able to guarantee a proof that replaces the derived step without any further proof transformation.

Basic Notions. We assume that the *abstract syntax* of the underlying object language is given as an abstract data type. Basic (static) types for this data structure include V for (object-level) variables, *TERM* for (object-level) terms, and *FOR* for (object-level) formulas. The axiomatization uses *constructor symbols*, like *mk-and* : $[FOR, FOR \to FOR]$ and *mk-ex* : $[V, FOR \to FOR]$ for building up conjunctive and existentially quantified formulas, respectively. *Selectors* are used to decompose structured objects. As usual *predicates*, like *Ex* and *All* serve to detect the kind of formula we are dealing with.

In addition to the abstract syntax of the object language we rely on *auxiliary* data structures, like natural numbers (NAT), lists ($LIST(\ldots)$), and trees ($TREE(\ldots)$). For lists we use the following notation:

- *emptyl* : $LIST(\ldots)$ for the empty list
- *cons* : $[\ldots, LIST(\ldots) \to LIST(\ldots)]$ for adding an element in front of a list
- $\cdot$: $[LIST(\ldots), LIST(\ldots) \to LIST(\ldots)]$ for concatenating lists
- $.$: $[LIST(\ldots), NAT \to \ldots]$ for selecting elements
- $\downarrow$: $[LIST(\ldots), NAT \to LIST(\ldots)$ for computing the initial segment
- $|\,|$: $[LIST(\ldots) \to NAT]$ for the length of a list
- $-$: $[LIST(\ldots), LIST(\ldots) \to LIST(\ldots)]$ for removing all occurrences of elements given by the second list from the first list.

Above the level of formula we have *sequents* and (proof) *trees* built up by *mk-seq* : $[LIST(FOR), LIST(FOR) \to SEQ]$ and *mk-tree* : $[SEQ, LIST(SEQ) \to TREE(SEQ)]$.

The function given by $fv_f : [FOR \to LIST(V)]$ computes a list containing all variables that have (free) occurrences in a formula. For terms we use fv_t. fv_f^* and fv_t^* are extensions to lists of formulas and terms, respectively. Substitutions are computed by *subst* : $[FOR, V, TERM \to FOR]$.

As our meta-language we use a higher-order language where all function symbols are interpreted as total functions.

Metalevel Representation of Simultaneous Quantifier Elimination. In a first step we define a set of trees given by a predicate Qe. Not all instances of this scheme represent valid proof steps. Those trees for which a proof in the basic calculus exists are filtered out by additional constraints later on.

$$\forall t : \mathit{TREE}(\mathit{SEQ}).\forall fl : \mathit{List}(\mathit{FOR}).\forall l : \mathit{LIST}(\mathit{LIST}(\mathit{TERM})).$$
$$\forall m : \mathit{LIST}(\mathit{LIST}([\mathit{LIST}(V) \to V])).\forall q : \mathit{LIST}(\mathit{NAT}).$$
$$Qe(t, fl, l, m, q) \leftrightarrow (Fits(fl, l, m, q) \wedge$$
$$t \equiv \mathit{mk\text{-}tree}(\mathit{mk\text{-}seq}(\mathit{emptyl}, fl), \mathit{cons}(\mathit{mk\text{-}seq}(\mathit{emptyl}, \mathit{elim}^*(fl, l, m, q)), \mathit{emptyl})))$$

The succedent of the conclusion of the rule is given by fl the antecedent of both the conclusion and the premise being empty. The terms that are substituted for existential quantifiers are given by l. For each formula in fl there has to be a list in l of appropriate length. In the same way m contains lists of so-called Skolem-functions used to replace universally quantified variables. In the current stage of the development the functions in the lists of m are not constrained at all. All we know is their type: $[\mathit{LIST}(V) \to V]$. From this we already see that Skolem-functions compute (object-level) variables. Finally q determines the number of quantifiers that we want to remove from the quantifier prefix of each formula in fl. If there is no quantifier prefix for a member of fl then the corresponding number in q has to be zero. This as well as other (more or less obvious) *syntactic* constraints for the arguments are formalized by $Fits$.

We continue with the definition of $\mathit{elim}^*(fl, l, m, q)$ which computes the succedent of the premise of the rule:

$$\mathit{elim}^*(fl, l, m, q) \equiv el^*(fl, l, m, q, 0) \qquad n \equiv |q| \to el^*(fl, l, m, q, n) \equiv \mathit{emptyl}$$
$$(n < |q| \to el^*(fl, l, m, q, n) \equiv$$
$$\mathit{cons}(\mathit{elim}(fl, fl.n, l.n, m.n, q.n, 0, 0), el^*(fl, l, m, q, n + 1))$$

The function *elim* successively removes one quantifier after the other. Existentially quantified variables are replaced by terms given by the third argument. These terms correspond to the meta-variables used by the object-level proof procedure. The Skolem-functions given by the fourth argument are used to replace universally quantified variables. Skolem-functions are applied to a list of variables containing the free variables of fl and those variables occurring in the terms that have been substituted for existentially quantified variables before.

$$(\mathit{Ex}(f) \wedge i \not\equiv 0) \to \mathit{elim}(fl, f, tl, sl, i, u, v) \equiv$$
$$\mathit{elim}(fl, \mathit{subst}(\mathit{body}(f), \mathit{bvar}(f), tl.u), tl, sl, i - 1, u + 1, v)$$
$$(\mathit{All}(f) \wedge i \not\equiv 0) \to \mathit{elim}(fl, f, tl, sl, i) \equiv$$
$$\mathit{elim}(fl, \mathit{subst}(\mathit{body}(f), \mathit{bvar}(f), sl.v(fv_f^*(fl) \cdot fv_t^*(tl \downarrow u))), tl, sl, i - 1, u, v + 1))$$
$$(\neg \mathit{Ex}(f) \vee \neg \mathit{All}(f) \vee i \not\equiv 0) \to \mathit{elim}(fl, f, tl, sl, i) \equiv f$$

We are left with the problem of filtering out the valid instances of Qe. This is done by introducing an additional list of natural numbers e which defines a possible order of *sequential* quantifier elimination steps. If we have $e.n \equiv i$ this means that in the n^{th} step the leading quantifier of the i^{th} formula is removed. Of course e has to be constrained with respect to fl and q. For example the number of occurrences of i in e, denoted by $\sharp(e, i)$, has to be equal to the $q.i$. These additional constraints are formalized by $\mathit{Adm}(e, fl, q)$.

The main problem that has to be solved is to provide a suitable meaning for the Skolem-functions given by the lists in m. Note that $(m.i).j$ is the Skolem-function used to eliminate the j^{th} universal quantifier in the i^{th} formula. We define:

$$\begin{aligned}&\mathit{Skol}(fl, l, m, q, e) \leftrightarrow \forall 0 \leq i < |m|.\forall 0 \leq j < |m.i|.\ \ (m.i).j \equiv \\ &\quad \lambda tl : \mathit{LIST}(\mathit{TERM}).\ s((fv^*(\mathit{elim}^*(fl, l, m, \mathit{red}(q, fl, e, i, j))) - fv^*(fl)) \cdot tl),\end{aligned}$$

where the reduced list $\mathit{red}(q, fl, e, i, j)$ is defined by

$$\forall 0 \leq k < |m|.\ \mathit{red}(q, fl, e, i, j).k \equiv \sharp(e \downarrow \mathit{step}(fl, e, i, j), k)\ .$$

s is a fixed function (the basic Skolem-function) that computes a variable which is not contained in the list of variables given as an argument and $\mathit{step}(fl, e, i, j)$ is the step in e in which the j^{th} universal quantifier of $fl.i$ is removed.

Using this predicate we can clarify the nature of our syntactic Skolem-functions by proving the following basic theorem:

$$\begin{aligned}&\forall t : \mathit{TREE}.\forall fl : \mathit{LIST}(\mathit{FOR}).\forall l : \mathit{LIST}(\mathit{LIST}(\mathit{TERM})).\\ &\forall m : \mathit{LIST}(\mathit{LIST}([\mathit{LIST}(V) \to V])).\forall q : \mathit{LIST}(\mathit{NAT}).\forall e : \mathit{LIST}(\mathit{NAT})\\ &\quad ((\mathit{Adm}(e, fl, q) \wedge \mathit{Qe}(t, fl, l, m, q) \wedge \mathit{Skol}(fl, l, m, q, e)) \to \mathit{Prov}(t))\end{aligned}$$

Unfortunately this result is not yet satisfactory since during proof construction we allow ***substitutions*** involving Skolem-functions. To treat substitutions (for metavariables denoting terms) we replace fl and l by functions $\widetilde{fl} : [\mathit{LIST}(\mathit{TERM}) \to \mathit{LIST}(\mathit{FOR})]$ and $\tilde{l} : [\mathit{LIST}(\mathit{TERM}) \to \mathit{LIST}(\mathit{LIST}(\mathit{TERM}))]$. The function $gl : [\mathit{LIST}(\mathit{LIST}([\mathit{LIST}(V) \to V])) \to \mathit{LIST}(\mathit{TERM})]$ provides the right hand sides of the substitutions.

Making the substituted terms depending on a structure m (of functions) allows to formalize substitutions that involve Skolem-functions. Every substitution occurring in the actual proof process can be represented as a list gl.

Again, $\widetilde{fl}, \tilde{l}, gl$, and q have to fit together which is expressed by the predicate $Fits_s$. Using these additional notations we are able to prove:

$$\begin{aligned}&\forall gl.\forall \widetilde{fl}.\forall \tilde{l}.\forall q.(\mathit{Fits}_s(\widetilde{fl}, \tilde{l}, gl, q) \to \exists m.\exists e.(\mathit{Fits}(\widetilde{fl}(gl(m)), \tilde{l}(gl(m)), m, q) \wedge \\ &\qquad\qquad \mathit{Adm}(e, \widetilde{fl}(gl(m)), q) \wedge\ \mathit{Skol}(\widetilde{fl}(gl(m)), \tilde{l}(gl(m)), m, q, e)))\end{aligned}$$

Since the definition of gl allows the *nesting* of functions the choice of e (and therefore also of m) really depends on gl. In this way we have modelled the fact that the fullsemantics of the Skolem-functions can only determined a posteriori, that is after a substitution has been applied.

The meta-level formalization we have presented exactly models the usage of the quantifier elimination rule by separating the visible part, given by the argument list of the elements in m, from the hidden part, given by the additional arguments supplied to s. It is this hidden part that can only be determined after a substitution has been made.

6 Conclusion

We defined the sequent calculus $\mathcal{K}$ which incorporates a rule for simultaneous quantifier elimination. $\mathcal{K}$ is sound and complete. The more difficult proof of the soundness theorem has been carried out by semantical as well as by syntactical arguments. While the semantical perspective clarifies the interdependencies of formulas in a sequent the syntactical approach allows to translate $\mathcal{K}$-proofs into usual sequent proofs which do not contain any meta-level constructs. This translation preserves the structure of a proof. Subsequently, simultaneous quantifier elimination can be used in sequent calculi for non-classical logics as well. The simultaneous quantifier elimination rule presented in this article has been implemented in the VSE II system for formal software development which is currently under development at the DFKI as a successor of the VSE system [7].

Other systems handle quantifiers with different degrees of sophistication. Lazy handling of instantiations has been used in classical as well as in non-classical logics and in first-order as well as in higher-order formalisms. Except in sequent calculus and natural deduction calculi, Skolemization has been studied in the context of resolution, connection method and tableau calculi as well.

For instance in PVS [5] Gentzen-like quantifier elimination rules are used where instantiations must be guessed. Ketonen and Weyhrauch [8] present an approach where sequents are annotated by a substitution, like in the semantical part of this article, and used a technique similar to our quantifier list ordering. However, they have only classical quantifier rules with the corresponding non-determinism in proof search. In the Isabelle system [11] a technique dual to classical skolemization is used. In remark 1 we have pointed out that this technique causes close interdependencies between formulas in a sequent which make it quite difficult to develop an optimized handling of quantifiers equivalent to our simultaneous quantifier elimination rule.

References

1. S. Autexier and H. Mantel. *Semantical Investigation of Simultaneous Skolemization for First-Order Sequent Calculus*, Seki Report SR-98-05, 1998.
2. A. Avron. Simple Consequence Relations, In *Information and Computation 92*, p. 105–139, 1991.
3. W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, 2nd edition, 1987.
4. K. A. Bowen. Programming with full first-order logic, In Hayes, Michie, Pao, Eds., *Machine Intelligence 10*, 1982.
5. J. Crow, S. Owre, J. Rushby, N. Shankar and M. Srivas. *A Tutorial Introduction to PVS*, Presented at WIFT'95, 1995.
6. G. Gentzen. *Untersuchungen über das logische Schließen*, Mathematische Zeitschrift. 39:179-210 and 405-431, 1935.
7. D. Hutter, B. Langenstein, C. Sengler, J. Siekmann, W. Stephan and A. Wolpers. Verification Support Environment (VSE), *High Integrity Systems*, p. 523-530, 1996.
8. J. Ketonen and R. Weyhrauch. A Decidable Fragment of Predicate Calculus, In *TCS, Volume 32-3*, 1984.
9. P. Lincoln and N. Shankar. Proof Search in First-Order Linear Logic and other Cut-Free Sequent Calculi, In *Proceedings of 9th LICS*, p. 282-291, 1994.
10. J. Loeckx, H.-D. Ehrich and M. Wolf. *Specification of Abstract Data Types*, Wiley-Teubner, 1996.
11. L.C. Paulson. *Isabelle, A Generic Theorem Prover*, Springer Verlag, 1994.
12. N. Shankar. Proof Search in the Intuitionistic Sequent Calculus, In D . Kapur, Ed., *Proceedings of CADE-11*, p. 522–536, 1992.
13. T. Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze, In *Skrifter utgit av Videnskapselskapet i Kristiania*, p. 4-36, 1920.
14. L. Wallen. *Automated Deduction in Non-Classical Logic.* MIT Press, 1990.