Implementation of Second-Order Cut-Elimination

Mikheil Rukhaia

Vienna University of Technology
e0827684@student.tuwien.ac.at

Abstract. *This paper is about implementation of second-order cut-elimination. We define LKII, the second-order calculus, extending LK with the second-order quantifier introduction rules. The goal was to extend CERES system for LKII-proofs, but method CERES cannot be extended to the second-order calculus in the straightforward way. We decided to extend Gentzen's method to LKII. We extended reduction rules for the second-order quantifiers. We implemented reduction rules and a specific algorithm to apply reduction rules to an LKII-proof and integrated it into the CERES system. So, system was extended to the second-order calculus.*

1 Introduction

Gentzen's cut-elimination theorem is one of the most important theorems of logic. Removing cuts from formal proofs corresponds to the elimination of intermediary lemmas from mathematical proofs. In cut-free proofs all statements are subformulas of the end-sequent and they allow extraction of Herbrand disjunctions and interpolants. So, cut-free proofs are analytic.

Gentzen's original proof of his theorem is constructive [1]. He gave a reductive first-order cut-elimination method that is nondeterministic. It is proved (for example by Orevkov in [7]) that cut-elimination in general is nonelementary, i.e. there is no elementary bound on the size of cut-free proof in terms of the original proof.

CERES is a first-order cut-elimination method by resolution. It is introduced by Matthias Baaz and Alexander Leitsch in [8]. CERES is based on the construction of a resolution refutation of the unsatisfiable characteristic clause set. The characteristic clause set is constructed from the derivation of cut-formula and a resolution refutation of this clause set is used as a skeleton of **LK**-proof with atomic cuts only. The complexity of the CERES method is directly related to the resolution refutation complexity of the characteristic clause set.

The CERES method is implemented in the system `CERES` [4]. The programs `CERES`, `HLK` [5] and `ProofTool` [6] are three computer programs that form a system for the computer-aided analysis of mathematical proofs. `HLK` is used to formalize mathematical proofs and generate input for `CERES`. Then `CERES` is used to transform formal proofs (into cut-free, Atomic Cut Normal Form (ACNF), etc.) and extract relevant information (characteristic clause set, herbrand sequent, etc.). `ProofTool` is used to visualize these formal proofs.

The goal was to extend the system CERES to second-order logic, but the CERES method can not be extended in the straightforward way to second-order logic because of various obstacles, such as a major difference between first- and second-order resolution calculus and skolemization. So, extension of Gentzen's method for the second-order calculus was the first solution.

Implementation of Gentzen style cut-elimination is mainly based on the algorithm described in the Gentzen's original proof of his cut-elimination theorem [1]. The basic idea is to use the cut reduction rules to move the cut rule upwards within the **LKII**-proof. We extended reduction rules for the second-order quantifier rules. The reduction rules are divided into two parts: the grade reduction and the rank reduction rules. The grade reduction rules reduce the complexity of the cut-formula and the rank reduction rules reduce the number of occurrences of the cut-formula in the proof. These rules are described in details in the next section. Third section is detailed description of the **SecondOrder** class that has been added to the system CERES. It also contains description of major changes in the existing CERES source code. The last section is conclusion.

2 Theoretical Background

In this section we will briefly describe the calculus **LKII**. Then we will present the reduction rules, that are implemented as methods of the SecondOrder class. At the end of this section we will present algorithm, how this reduction rules are applied to the **LKII**-proof in order to get a cut-free **LKII**-proof.

2.1 Calculus LKII

Calculus **LK** in our case is defined as it is in [3] with some minor changes. We do not have the *equality* and *definition* rules. The calculus **LKII** is defined as an extension of the calculus **LK** with the following second-order quantifier introduction rules:

$$\frac{A(X/\lambda\bar{x}.F), \Gamma \vdash \Delta}{(\forall^2 X)A, \Gamma \vdash \Delta}\,\forall^2 : l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(X/Y)}{\Gamma \vdash \Delta, (\forall^2 X)A}\,\forall^2 : r$$

$$\frac{A(X/Y), \Gamma \vdash \Delta}{(\exists^2 X)A, \Gamma \vdash \Delta}\,\exists^2 : l \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta, A(X/\lambda\bar{x}.F)}{\Gamma \vdash \Delta, (\exists^2 X)A}\,\exists^2 : r$$

Where X is a second-order variable, F is a second-order formula with free variables not bound in A and bound variables of F not in A; and Y is a second-order eigenvariable of the same type as X.

Notion of **LKII**-derivation and **LKII**-proof is defined in an analogous way as it is for **LK**-derivation and **LK**-proof.

2.2 Reduction Rules

Reduction rules are defined as they are in [2], slightly modified. They are divided into two parts: grade reduction and rank reduction. The grade reduction rules reduce the complexity of the cut-formula and the rank reduction rules reduce the number of occurrences of the cut-formula in the proof. Now we will list all these rules, named as the corresponding methods of the SecondOrder class.

Grade Reduction Rules Let us list all grade reduction rules in this subsection.

caseAndRAndL1. Cut rule premises are lower sequents of the $\wedge\colon r$ and $\wedge\colon l1$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_2}{\Gamma_2 \vdash \Delta_2, B}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\wedge\colon r \quad \dfrac{\dfrac{\phi_r}{A, \Gamma \vdash \Delta}}{A \wedge B, \Gamma \vdash \Delta}\wedge\colon l1}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_r}{A, \Gamma \vdash \Delta}}{\Gamma_1, \Gamma \vdash \Delta_1, \Delta}\, cut}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta}\, w, \pi\colon l*}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, w, \pi\colon r*$$

If Γ_2 and/or Δ_2 is empty, then the corresponding weakening and permutation rules are omitted.

caseAndRAndL2. Cut rule premises are lower sequents of the $\wedge\colon r$ and $\wedge\colon l2$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_2}{\Gamma_2 \vdash \Delta_2, B}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\wedge\colon r \quad \dfrac{\dfrac{\phi_r}{B, \Gamma \vdash \Delta}}{A \wedge B, \Gamma \vdash \Delta}\wedge\colon l2}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_2}{\Gamma_2 \vdash \Delta_2, B} \quad \dfrac{\phi_r}{B, \Gamma \vdash \Delta}}{\Gamma_2, \Gamma \vdash \Delta_2, \Delta}\, cut}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_2, \Delta}\, w, \pi\colon l*}{\Gamma_1, \Gamma_2, \Gamma \vdash \Delta_1, \Delta_2, \Delta}\, w, \pi\colon r*$$

If Γ_1 and/or Δ_1 is empty, then the corresponding weakening and permutation rules are omitted.

caseOrR1OrL. Cut rule premises are lower sequents of the $\vee\colon r1$ and $\vee\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, A}}{\Gamma \vdash \Delta, A \vee B} \vee\colon r1 \qquad \dfrac{\dfrac{\phi_1}{A, \Gamma_1 \vdash \Delta_1} \qquad \dfrac{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A \vee B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \vee\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, A} \qquad \dfrac{\phi_1}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma, \Gamma_1 \vdash \Delta, \Delta_1}\, cut}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1}\, w, \pi\colon l*}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, w, \pi\colon r*$$

If Γ_2 and/or Δ_2 is empty, then the corresponding weakening and permutation rules are omitted.

caseOrR2OrL. Cut rule premises are lower sequents of the $\vee\colon r2$ and $\vee\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, B}}{\Gamma \vdash \Delta, A \vee B} \vee\colon r2 \qquad \dfrac{\dfrac{\phi_1}{A, \Gamma_1 \vdash \Delta_1} \qquad \dfrac{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A \vee B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \vee\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma \vdash \Delta, B} \qquad \dfrac{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{\Gamma, \Gamma_2 \vdash \Delta, \Delta_2}\, cut}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_2}\, w, \pi\colon l*}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, w, \pi\colon r*$$

If Γ_1 and/or Δ_1 is empty, then the corresponding weakening and permutation rules are omitted.

caseImplRImplL. Cut rule premises are lower sequents of the $\supset\colon r$ and $\supset\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{A, \Gamma \vdash \Delta, B}}{\Gamma \vdash \Delta, A \supset B} \supset\colon r \qquad \dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A \supset B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \supset\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_l}{A, \Gamma \vdash \Delta, B} \qquad \dfrac{\phi_2}{B, \Gamma_2 \vdash \Delta_2}}{A, \Gamma, \Gamma_2 \vdash \Delta, \Delta_2}\, cut}{\Gamma_1, \Gamma, \Gamma_2 \vdash \Delta_1, \Delta, \Delta_2}\, cut}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta, \Delta_2}\, \pi\colon l}{\Gamma, \Gamma_1, \Gamma_2 \vdash \Delta, \Delta_1, \Delta_2}\, \pi\colon r$$

If Γ or Γ_1 is empty or $\Gamma = \Gamma_1$ and/or Δ or Δ_1 is empty or $\Delta = \Delta_1$, then the corresponding permutation rules are omitted.

caseNotRNotL. Cut rule premises are lower sequents of the $\neg\colon r$ and $\neg\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma_1 \vdash \Delta_1, \neg A}\,\neg\colon r \qquad \dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2, A}}{\neg A, \Gamma_2 \vdash \Delta_2}\,\neg\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2, A} \qquad \dfrac{\phi_l}{A, \Gamma_1 \vdash \Delta_1}}{\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1}\,cut}{\Gamma_1, \Gamma_2 \vdash \Delta_2, \Delta_1}\,\pi\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,\pi\colon r$$

If Γ_1 or Γ_2 is empty or $\Gamma_1 = \Gamma_2$ and/or Δ_1 or Δ_2 is empty or $\Delta_1 = \Delta_2$, then the corresponding permutation rules are omitted.

caseForallRForallL. Cut rule premises are lower sequents of the $\forall\colon r$ and $\forall\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/\alpha)}}{\Gamma_1 \vdash \Delta_1, (\forall x)A}\,\forall\colon r \qquad \dfrac{\dfrac{\phi_r}{A(x/t), \Gamma_2 \vdash \Delta_2}}{(\forall x)A, \Gamma_2 \vdash \Delta_2}\,\forall\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\dfrac{\phi_l(\alpha/t)}{\Gamma_1 \vdash \Delta_1, A(x/t)} \qquad \dfrac{\phi_r}{A(x/t), \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

caseExistsRExistsL. Cut rule premises are lower sequents of the $\exists\colon r$ and $\exists\colon l$ rules:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/t)}}{\Gamma_1 \vdash \Delta_1, (\exists x)A}\,\exists\colon r \qquad \dfrac{\dfrac{\phi_r}{A(x/\alpha), \Gamma_2 \vdash \Delta_2}}{(\exists x)A, \Gamma_2 \vdash \Delta_2}\,\exists\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A(x/t)} \qquad \dfrac{\phi_r(\alpha/t)}{A(x/t), \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

caseForallR2ForallL2. Cut rule premises are lower sequents of the $\forall^2\colon r$ and $\forall^2\colon l$ rules:

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A(X/Y)\end{matrix}}{\Gamma_1 \vdash \Delta_1, (\forall^2 X)A}\,\forall^2\colon r \qquad \dfrac{\begin{matrix}\phi_r\\ A(X/\lambda\bar{x}.F), \Gamma_2 \vdash \Delta_2\end{matrix}}{(\forall^2 X)A, \Gamma_2 \vdash \Delta_2}\,\forall^2\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\begin{matrix}\phi_l(Y/\lambda\bar{x}.F)\\ \Gamma_1 \vdash \Delta_1, A(X/\lambda\bar{x}.F)\end{matrix} \qquad \begin{matrix}\phi_r\\ A(X/\lambda\bar{x}.F), \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

caseExistsR2ExistsL2. Cut rule premises are lower sequents of the $\exists^2\colon r$ and $\exists^2\colon l$ rules:

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A(X/\lambda\bar{x}.F)\end{matrix}}{\Gamma_1 \vdash \Delta_1, (\exists^2 X)A}\,\exists^2\colon r \qquad \dfrac{\begin{matrix}\phi_r\\ A(X/Y), \Gamma_2 \vdash \Delta_2\end{matrix}}{(\exists^2 X)A, \Gamma_2 \vdash \Delta_2}\,\exists^2\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A(X/\lambda\bar{x}.F)\end{matrix} \qquad \begin{matrix}\phi_r(Y/\lambda\bar{x}.F)\\ A(X/\lambda\bar{x}.F), \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

Rank Reduction Rules Let us list all rank reduction rules in this subsection.

caseLeftAxiom. Cut rule left premise is an axiom:

$$\dfrac{A \vdash A \qquad \begin{matrix}\phi_r\\ A, \Gamma \vdash \Delta\end{matrix}}{A, \Gamma \vdash \Delta}\,cut \qquad \text{transforms to} \qquad \begin{matrix}\phi_r\\ A, \Gamma \vdash \Delta\end{matrix}$$

caseRightAxiom. Cut rule right premise is an axiom:

$$\dfrac{\begin{matrix}\phi_l\\ \Gamma \vdash \Delta, A\end{matrix} \qquad A \vdash A}{\Gamma \vdash \Delta, A}\,cut \qquad \text{transforms to} \qquad \begin{matrix}\phi_l\\ \Gamma \vdash \Delta, A\end{matrix}$$

caseLeftWeakR. Cut rule left premise is the lower sequent of the $w\colon r$ rule:

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1\end{matrix}}{\Gamma_1 \vdash \Delta_1, A}\,w\colon r \qquad \begin{matrix}\phi_r\\ A, \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1}\,w, \pi\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,w, \pi\colon r*$$

If Γ_2 is empty and/or Δ_2 is empty, then the corresponding weakening and permutation rules are omitted.

caseRightWeakL. Cut rule right premise is the lower sequent of the $w\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2}}{A, \Gamma_2 \vdash \Delta_2}\, w\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\phi_r}{\Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_2}\, w, \pi\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, w, \pi\colon r*$$

If Γ_1 is empty and/or Δ_1 is empty, then the corresponding weakening and permutation rules are omitted.

caseLeftContrR. Cut rule left premise is the lower sequent of the $c\colon r$ rule:

$$\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A^n}\, c\colon r}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A^n} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, A^{n-1}, \Delta_2}\, cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A^{n-1}}\, \pi\colon r} \qquad \dfrac{\phi_r^{n-1}}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2, \Gamma_2 \vdash \Delta_1, \Delta_2, A^{n-2}, \Delta_2}\, cut}{\Gamma_1, \Gamma_2, \Gamma_2 \vdash \Delta_1, \Delta_2, \Delta_2, A^{n-2}}\, \pi\colon r$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma_1, \Gamma_2, \ldots, \Gamma_2 \vdash \Delta_1, \Delta_2, \ldots, \Delta_2, A} \qquad \dfrac{\phi_r^1}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2, \ldots, \Gamma_2 \vdash \Delta_1, \Delta_2, \ldots, \Delta_2, \Delta_2}\, cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, \ldots, \Delta_2, \Delta_2}\, \pi, c\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, \pi, c\colon r*$$

Where ϕ_r^i is a variant of ϕ_r for all $i = 1, \ldots, n-1$, such that the (first- and second-order) eigenvariables are renamed. If Γ_2 and/or Δ_2 is empty, then the corresponding contraction and permutation rules are omitted.

caseRightContrL. Cut rule right premise is the lower sequent of the $c\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_r}{A^n, \Gamma_2 \vdash \Delta_2}}{A, \Gamma_2 \vdash \Delta_2}\, c\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l^{n-1}}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_r}{A^n, \Gamma_2 \vdash \Delta_2}\, cut}{\Gamma_1, A^{n-1}, \Gamma_2 \vdash \Delta_1, \Delta_2}\, \pi\colon l}{A^{n-1}, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut}{A^{n-1}, \Gamma_1, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_1, \Delta_2}\, \pi\colon l$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_l^1}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\vdots}{A, \Gamma_1, \ldots, \Gamma_1, \Gamma_2 \vdash \Delta_1, \ldots, \Delta_1, \Delta_2}\, cut}{\Gamma_1, \ldots, \Gamma_1, \Gamma_2 \vdash \Delta_1, \ldots, \Delta_1, \Delta_2}\, \pi, c\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \ldots, \Delta_1, \Delta_2}\, \pi, c\colon r*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

Where ϕ_l^i is a variant of ϕ_l for all $i = 1, \ldots, n-1$, such that the (first- and second-order) eigenvariables are renamed. If Γ_1 and/or Δ_1 is empty, then the corresponding contraction and permutation rules are omitted.

caseLeftUnary. Cut rule left premise is the lower sequent of an arbitrary unary rule, except the $\pi\colon r$ rule:

$$\dfrac{\dfrac{\phi_l}{\dfrac{\Gamma \vdash \Delta}{\Gamma_1 \vdash \Delta_1, A}\, unary} \quad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta', A}\, \pi\colon r} \quad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\dfrac{\Gamma, \Gamma_2 \vdash \Delta', \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, unary, \pi\colon r*}\, cut$$

If $\Delta = \Delta_2$ then the corresponding permutation rules are omitted.

caseRightUnary. Cut rule right premise is the lower sequent of an arbitrary unary rule, except the $\pi\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_r}{\dfrac{\Gamma \vdash \Delta}{A, \Gamma_2 \vdash \Delta_2}\, unary}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

transforms to

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \quad \dfrac{\phi_r}{\dfrac{\Gamma \vdash \Delta}{A, \Gamma' \vdash \Delta}\, \pi\colon l}}{\dfrac{\Gamma_1, \Gamma' \vdash \Delta_1, \Delta}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, unary, \pi\colon l*}\, cut$$

If $\Gamma_1 = \Gamma$ then the corresponding permutation rules are omitted.

caseLeftBinary. Cut rule left premise is the lower sequent of an arbitrary binary rule, except the *cut* rule:

$$\dfrac{\dfrac{\begin{matrix}\phi_1 & \phi_2\\ \Gamma_1 \vdash \Delta_1, A & \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma \vdash \Delta, A}\, binary \qquad \phi_r}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut \qquad A, \Pi \vdash \Lambda$$

transforms to

$$\dfrac{\dfrac{\begin{matrix}\phi_1 & \phi_r\\ \Gamma_1 \vdash \Delta_1, A & A, \Pi \vdash \Lambda\end{matrix}}{\Gamma_1, \Pi \vdash \Delta_1, \Lambda}\, cut \qquad \phi_2}{\Gamma, \Pi \vdash \Delta, \Lambda}\, binary \qquad \Gamma_2 \vdash \Delta_2$$

Or if cut-formula comes from the right premise of the binary rule:

$$\dfrac{\dfrac{\begin{matrix}\phi_1 & \phi_2\\ \Gamma_1 \vdash \Delta_1 & \Gamma_2 \vdash \Delta_2, A\end{matrix}}{\Gamma \vdash \Delta, A}\, binary \qquad \phi_r}{\Gamma, \Pi \vdash \Delta, \Lambda}\, cut \qquad A, \Pi \vdash \Lambda$$

transforms to

$$\dfrac{\begin{matrix}\phi_1 & \dfrac{\begin{matrix}\phi_2 & \phi_r\\ \Gamma_2 \vdash \Delta_2, A & A, \Pi \vdash \Lambda\end{matrix}}{\Gamma_2, \Pi \vdash \Delta_2, \Lambda}\, cut\\ \Gamma_1 \vdash \Delta_1 & \end{matrix}}{\Gamma, \Pi \vdash \Delta, \Lambda}\, binary$$

caseRightBinary. Cut rule right premise is the lower sequent of an arbitrary binary rule, except the *cut* rule:

$$\dfrac{\phi_l \qquad \dfrac{\begin{matrix}\phi_1 & \phi_2\\ A, \Gamma_1 \vdash \Delta_1 & \Gamma_2 \vdash \Delta_2\end{matrix}}{A, \Gamma \vdash \Delta}\, binary}{\Pi, \Gamma \vdash \Lambda, \Delta}\, cut \qquad \Pi \vdash \Lambda, A$$

transforms to

$$\dfrac{\dfrac{\begin{matrix}\phi_l & \phi_1\\ \Pi \vdash \Lambda, A & A, \Gamma_1 \vdash \Delta_1\end{matrix}}{\Pi, \Gamma_1 \vdash \Lambda, \Delta_1}\, cut \qquad \phi_2}{\Pi, \Gamma \vdash \Lambda, \Delta}\, binary \qquad \Gamma_2 \vdash \Delta_2$$

Or if cut-formula comes from the right premise of the binary rule:

$$\dfrac{\phi_l \qquad \dfrac{\begin{matrix}\phi_1 & \phi_2\\ \Gamma_1 \vdash \Delta_1 & A, \Gamma_2 \vdash \Delta_2\end{matrix}}{A, \Gamma \vdash \Delta}\, binary}{\Pi, \Gamma \vdash \Lambda, \Delta}\, cut \qquad \Pi \vdash \Lambda, A$$

transforms to

$$\dfrac{\dfrac{\phi_1}{\Gamma_1 \vdash \Delta_1} \qquad \dfrac{\dfrac{\phi_l}{\Pi \vdash \Lambda, A} \qquad \dfrac{\phi_2}{A, \Gamma_2 \vdash \Delta_2}\, cut}{\Pi, \Gamma_2 \vdash \Lambda, \Delta_2}\, binary}{\Pi, \Gamma \vdash \Lambda, \Delta}$$

caseLeftPermR. Cut rule left premise is the lower sequent of the $\pi\colon r$ rule:

$$\dfrac{\dfrac{\dfrac{\Phi}{\Gamma_1 \vdash \Delta_1, A}\, \pi\colon r \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut}$$

We distinguish cases according to Φ.

$$\Phi \equiv \dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta}\, w\colon r}{\Gamma_1 \vdash \Delta_1', A, \Delta_1''} \qquad \text{transforms to} \qquad \dfrac{\dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta}}{\Gamma_1, \Gamma_2 \vdash \Delta}\, w, \pi\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, w, \pi\colon r*$$

Where $\Delta_1 = \Delta_1', \Delta_1''$. If Γ_2 is empty and/or $\Delta_1 = \Delta$ and Δ_2 is empty, then the corresponding weakening and permutation rules are omitted.

$$\Phi \equiv \dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta}\, \pi\colon r}{\Gamma_1 \vdash \Delta_1', A, \Delta_1''} \text{ transforms to}$$

$$\dfrac{\dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta}\, \pi\colon r}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

$$\Phi \equiv \dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta_1', A, A, \Delta_1''}\, c\colon r}{\Gamma_1 \vdash \Delta_1', A, \Delta_1''}$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi}{\Gamma_1 \vdash \Delta_1', A, A, \Delta_1''}\, \pi\colon r}{\Gamma_1 \vdash \Delta_1, A, A}\, c\colon r \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut$$

$$\Phi \equiv \dfrac{\dfrac{\phi}{\Gamma_1' \vdash \Delta_1', A, \Delta_1''}\, unary}{\Gamma_1 \vdash \Delta_1', A, \Delta_1'''}$$

transforms to

$$\dfrac{\dfrac{\dfrac{\phi}{\Gamma_1' \vdash \Delta_1', A, \Delta_1''}\,\pi\colon r}{\Gamma_1' \vdash \Delta_1', \Delta_1'', A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}\,cut}{\dfrac{\Gamma_1', \Gamma_2 \vdash \Delta_1', \Delta_1'', \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,unary, \pi\colon r*}$$

Where $\Delta_1 = \Delta_1', \Delta_1'''$ and *unary* is an arbitrary unary rule.

$$\Phi \equiv \dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l, A, \Delta_l''} \qquad \dfrac{\phi_2}{\Gamma_r \vdash \Delta_r}}{\Gamma_1 \vdash \Delta_l, A, \Delta_l', \Delta_r'}\,binary$$

transforms to

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l, A, \Delta_l''}\,\pi\colon r}{\Gamma_l \vdash \Delta_l, \Delta_l'', A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}\,cut}{\dfrac{\Gamma_l, \Gamma_2 \vdash \Delta_l, \Delta_l'', \Delta_2}{\Gamma_l, \Gamma_2 \vdash \Delta_l, \Delta_2, \Delta_l''}\,\pi\colon r} \qquad \dfrac{\phi_2}{\Gamma_r \vdash \Delta_r}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,binary$$

Where $\Delta_1 = \Delta_l, \Delta_l', \Delta_r'$ and *binary* is an arbitrary binary rule.

$$\Phi \equiv \dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l} \qquad \dfrac{\phi_2}{\Gamma_r \vdash \Delta_r, A, \Delta_r''}}{\Gamma_1 \vdash \Delta_l', \Delta_r, A, \Delta_r'}\,binary$$

transforms to

$$\dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\phi_2}{\Gamma_r \vdash \Delta_r, A, \Delta_r''}\,\pi\colon r}{\Gamma_r \vdash \Delta_r, \Delta_r'', A} \qquad \dfrac{\phi_r}{A, \Gamma_2 \vdash \Delta_2}\,cut}{\dfrac{\Gamma_r, \Gamma_2 \vdash \Delta_r, \Delta_r'', \Delta_2}{\Gamma_r, \Gamma_2 \vdash \Delta_r, \Delta_2, \Delta_r''}\,\pi\colon r}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,binary$$

Where $\Delta_1 = \Delta_l', \Delta_r, \Delta_r'$ and *binary* is an arbitrary binary rule.

caseRightPermL. Cut rule left premise is the lower sequent of the $\pi\colon l$ rule:

$$\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\Phi}{A, \Gamma_2 \vdash \Delta_2}\,\pi\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

We distinguish cases according to Φ.

$$\Phi \equiv \dfrac{\dfrac{\phi}{\Gamma \vdash \Delta_2}}{\Gamma_2', A, \Gamma_2'' \vdash \Delta_2}\,w\colon l \qquad \text{transforms to} \qquad \dfrac{\dfrac{\dfrac{\phi}{\Gamma \vdash \Delta_2}}{\Gamma_1, \Gamma_2 \vdash \Delta_2}\,w, \pi\colon l*}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,w, \pi\colon r*$$

Where $\Gamma_2 = \Gamma_2', \Gamma_2''$. If $\Gamma_2 = \Gamma$ and Γ_1 is empty and/or Δ_1 is empty, then the corresponding weakening and permutation rules are omitted.

$$\Phi \equiv \dfrac{\begin{matrix}\phi\\ \Gamma \vdash \Delta_2\end{matrix}}{\Gamma_2', A, \Gamma_2'' \vdash \Delta_2}\,\pi\colon l \text{ transforms to } \dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \dfrac{\begin{matrix}\phi\\ \Gamma \vdash \Delta_2\end{matrix}}{A, \Gamma_2 \vdash \Delta_2}\,\pi\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

$$\Phi \equiv \dfrac{\begin{matrix}\phi\\ \Gamma_2', A, A, \Gamma_2'' \vdash \Delta_2\end{matrix}}{\Gamma_2', A, \Gamma_2'' \vdash \Delta_2}\,c\colon l$$

transforms to

$$\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \dfrac{\dfrac{\begin{matrix}\phi\\ \Gamma_2', A, A, \Gamma_2'' \vdash \Delta_2\end{matrix}}{A, A, \Gamma_2 \vdash \Delta_2}\,c\colon l}{A, \Gamma_2 \vdash \Delta_2}\,\pi\colon l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,cut$$

$$\Phi \equiv \dfrac{\begin{matrix}\phi\\ \Gamma_2'', A, \Gamma_2' \vdash \Delta_2'\end{matrix}}{\Gamma_2''', A, \Gamma_2' \vdash \Delta_2}\,unary$$

transforms to

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \dfrac{\begin{matrix}\phi\\ \Gamma_2'', A, \Gamma_2' \vdash \Delta_2'\end{matrix}}{A, \Gamma_2'', \Gamma_2' \vdash \Delta_2'}\,\pi\colon l}{\Gamma_1, \Gamma_2'', \Gamma_2' \vdash \Delta_1, \Delta_2'}\,cut}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,unary, \pi\colon l*$$

Where $\Gamma_2 = \Gamma_2''', \Gamma_2'$ and *unary* is an arbitrary unary rule.

$$\Phi \equiv \dfrac{\begin{matrix}\phi_1\\ \Gamma_l'', A, \Gamma_l \vdash \Delta_l\end{matrix} \quad \begin{matrix}\phi_2\\ \Gamma_r \vdash \Delta_r\end{matrix}}{\Gamma_l', A, \Gamma_l, \Gamma_r \vdash \Delta_2}\,binary$$

transforms to

$$\dfrac{\dfrac{\begin{matrix}\phi_l\\ \Gamma_1 \vdash \Delta_1, A\end{matrix} \quad \dfrac{\begin{matrix}\phi_1\\ \Gamma_l'', A, \Gamma_l \vdash \Delta_l\end{matrix}}{A, \Gamma_l'', \Gamma_l \vdash \Delta_l}\,\pi\colon l}{\Gamma_1, \Gamma_l'', \Gamma_l \vdash \Delta_1, \Delta_l}\,cut \quad \begin{matrix}\phi_2\\ \Gamma_r \vdash \Delta_r\end{matrix}}{\Gamma_l'', \Gamma_1, \Gamma_l \vdash \Delta_1, \Delta_l}\,\pi\colon l \qquad \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,binary$$

Where $\Gamma_2 = \Gamma_l', \Gamma_l, \Gamma_r$ and *binary* is an arbitrary binary rule.

$$\Phi \equiv \dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l} \qquad \dfrac{\phi_2}{\Gamma''_r, A, \Gamma_r \vdash \Delta_r}}{\Gamma'_l, \Gamma'_r, A, \Gamma_r \vdash \Delta_2}\, binary$$

transforms to

$$\dfrac{\dfrac{\phi_1}{\Gamma_l \vdash \Delta_l} \qquad \dfrac{\dfrac{\dfrac{\phi_l}{\Gamma_1 \vdash \Delta_1, A} \qquad \dfrac{\dfrac{\phi_2}{\Gamma''_r, A, \Gamma_r \vdash \Delta_r}}{A, \Gamma''_r, \Gamma_r \vdash \Delta_r}\,\pi : l}{\Gamma_1, \Gamma''_r, \Gamma_r \vdash \Delta_1, \Delta_r}\,cut}{\Gamma''_r, \Gamma_1, \Gamma_r \vdash \Delta_1, \Delta_r}\,\pi : l}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\,binary$$

Where $\Gamma_2 = \Gamma'_l, \Gamma'_r, \Gamma_r$ and *binary* is an arbitrary binary rule.

2.3 Cut-elimination Algorithm

We decided to implement the following algorithm: the program eliminates the leftmost upmost cut rule occurring into the proof. It goes through the proof, starting from bottom and when it finds the upmost leftmost cut rule, first it tries to reduce the grade if possible, then it tries to reduce the rank. For the grade reduction, program first tries to remove from the cut-formula second-order quantifiers, then first-order quantifiers, then implication, then $\wedge$, then $\vee$, and then $\neg$. If all these cases fail, then the program begins to apply the rank reduction rules. First it tries to apply weakening rule cases, then axiom rule cases, then contraction rule cases, then arbitrary unary and binary rule cases and if none of them is applicable, then the program tries permutation rule cases. For each rule case, the program first tries to reduce rank on the left and then on the right cut-derivation. This procedure is applied repeatedly, while all cuts are eliminated from the proof.

3 Implementation

The system `CERES` is written on C++. It is a quite complex system with his own data-structures. Our cut-elimination algorithm for the second-order calculus is implemented into two files: **SecondOrder.h** and **SecondOrder.cpp**. The class **SecondOrder** is publicly derived from the class **ProofOperation**. A constructor of the SecondOrder class receives the *CeresInput* object as input and extracts from this object *LKProof*, *ProofDatabase* and *AxiomSet*. Also assigns the value *false* to the global boolean variable **error** and creates *LKProofVector*. The last one is used to hold intermediary proofs.

All reduction rules are implemented as methods of the *SecondOrder* class. Each of them takes *RuleCut* as input and returns either transformed proof as a *ProofNode* object, or the value *NULL* (if it was not a proper case, or there was some error during the transformation of the proof).

The algorithm, that was described in Section 2.3 is implemented as the method **executeCut**. In this method after applying the reduction rules, we test if none of reduction rules were applied correctly (i.e. all of them returned the value *NULL*), then error messages are printed, *error* is set to be *true* and the program terminates.

In the **main.cpp** file we added boolean variables **arg_second_order** and **arg_write_interm_proofs**, which are *false* by default and they are set to be *true* if user typed in command line the parameters **-so** and **-w** respectively. The first one is a parameter to tell program that the input proof is a second-order proof and the second one is to tell if we want to output intermediary proofs or not. We test whether *arg_second_order* is set to be *true*, and if so, then the constructor of SecondOrder object is called and the operation SecondOrder is applied to the proof repeatedly, while proof is not a cut-free and there is no an error. In the cycle we test if *arg_write_interm_proofs* is true, then *lkproof* is added to *LKProofVector*. Here is the corresponding piece of code of the main function:

```
SecondOrder *t;
if (arg_second_order) {
    t = new SecondOrder(*input);
    LKProof *lkproof = t->getProof();
    std::string s = lkproof->getName();
    int i = 0;
    while (!t->isCutFree(lkproof->getRoot())&&!t->hasError()) {
      lkproof->execute(t);
      if(arg_write_interm_proofs) {
        i++;
        std::stringstream s_1;
        s_1 << i;
        t->setProofinProofVector(lkproof->clone(s +"_" + s_1.str()));
      }
    }
}
```

To export resulted proof(s) into the XML file, we have modified the **exportceres.h** and **exportceres.cpp** files. We have overloaded the method **write** and operator $\ll$ to output the SecondOrder object as an XML file. In the output file the program writes resulted cut-free proof, original proof, intermediary proofs that were saved in *LKProofVector* and the axiom set, if there was any.

There are some auxiliary methods implemented in the SecondOrder class. Most interesting one is **renameEigenvariables**, that takes a *ProofNode* object as input and returns again a *ProofNode* object, where first- and second- order eigenvariables are renamed to preserve the proof regularity. This method is necessary when the reduction rules for contraction cases are executed. The method calls the **renameFirstOrderEigenvariables** and **renameSecondOrderEigenvariables** methods. Later ones are implemented by the following algorithm for first- and second-order eigenvariables respectively: first create *blacklist*, a list of eigenvariables of the whole proof. Then create *eigvlist*, a list of eigenvari-

ables of input *ProofNode* (in which eigenvariables should be renamed). Then go through *eigvlist* and generate for each variable a fresh one, which is not in *blacklist*. Then construct a substitution and apply it to the input *ProofNode*. Then return *ProofNode*.

To create a list of first- and second-order eigenvariables, the methods **buildEigvarList** and **buildSecondOrderEigvarVector** are implemented. They take *ProofNode* as input and output *VariableList* and *SecondOrderVariableVector* respectively. Algorithm is very simple: go throw the proof and find strong quantifier rules. Get from this rules eigenvariables and add them into the list. Then return the list.

The code, which was written in the *SecondOrder.h* and *SecondOrder.cpp* files together is around 2000 lines.

4 Conclusion

The algorithm according to the Gentzen's original method was implemented and integrated into the existing `CERES` system. So, the `CERES` system was extended for second-order proofs. Using the `CERES` system, it is possible to compare two methods of cut-elimination (the CERES method and the Gentzen's method) and their performance. To call the program for second-order proofs user should use the **-so** parameter from a command line and the optional parameter **-w** if he/she wants to write all intermediary proofs in the output file.

The system was tested with some inputs and bugs that were found are corrected.

References

1. G. Takeuti. *Proof Theory.* The foundations of mathematics, volume 81, Illinois, U.S.A., 1975.
2. C. Richter. *Proof Transformations by Resolution.* PhD thesis, Vienna University of Technology, 2006.
3. *The Calculus LK.* Available at http://www.logic.at/ceres/downloads/calculus_LK.pdf
4. *The Cut-Elimination System `CERES` Homepage.* Available at http://www.logic.at/ceres/
5. *Handy LK Homepage.* Available at http://www.logic.at/hlk/
6. *Proof Tool Homepage.* Available at http://www.logic.at/prooftool/
7. V.P. Orevkov, *Lower Bounds for Increasing Complexity of Derivations After Cut-Elimination.* Journal of Soviet Mathematics, 1982.
8. Matthias Baaz, Alexander Leitsch, *Cut-elimination and Redundancy-elimination by Resolution.* Journal of Symbolic Computation, 2000.