Theoretical Computer Science 549 (2014) 1–16

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Algorithmic introduction of quantified cuts ☆

Stefan Hetzl [a], Alexander Leitsch [b], Giselle Reis [b], Daniel Weller [a,*]

[a] Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Vienna, Austria
[b] Institute of Computer Languages, Vienna University of Technology, Vienna, Austria

ARTICLE INFO

Article history:
Received 23 January 2013
Received in revised form 15 May 2014
Accepted 27 May 2014
Available online 2 June 2014
Communicated by A. Avron

Keywords:
Proof compression
Cut-introduction
Classical logic
First-order logic

ABSTRACT

We describe a method for inverting Gentzen's cut-elimination in classical first-order logic. Our algorithm is based on first computing a compressed representation of the terms present in the cut-free proof and then cut-formulas that realize such a compression. Finally, a proof using these cut-formulas is constructed. Concerning asymptotic complexity, this method allows an exponential compression of quantifier complexity (the number of quantifier-inferences) of proofs.

1. Introduction

Cut-elimination introduced by Gentzen [15] is the most prominent form of proof transformation in logic and plays an important role in automating the analysis of mathematical proofs. The removal of cuts corresponds to the elimination of intermediate statements (lemmas), resulting in a proof which is analytic in the sense that all statements in the proof are subformulas of the result. Thus a proof of a combinatorial statement is converted into a purely combinatorial proof. Cut-elimination is therefore an essential tool for the analysis of proofs, especially to make implicit parameters explicit.

In this paper we present a method for inverting Gentzen's cut-elimination by computing a proof with cut from a given cut-free proof as input. As cut-elimination is the backbone of proof theory, there is considerable proof-theoretic interest and challenge in understanding this transformation sufficiently well to be able to invert it. But our interest in cut-introduction is not only of a purely theoretical nature. Proofs with cuts have properties that are essential for applications: on the one hand, cuts are indispensable for formalizing proofs in a human-readable way. One the other hand cuts have a very strong compression power in terms of proof length.

Computer-generated proofs are typically analytic, i.e. they only contain logical material that also appears in the theorem shown. This is due to the fact that analytic proof systems have a considerably smaller search space which makes proof-search practically feasible. In the case of the sequent calculus, proof-search procedures typically work on the cut-free fragment. But also resolution is essentially analytic as resolution proofs satisfy the subformula property of first-order logic. An important property of non-analytic proofs is their considerably smaller length. The exact difference depends on the logic (or theory) under consideration, but it is typically enormous. In (classical and intuitionistic) first-order logic there are proofs with cut of length n whose theorems have only cut-free proofs of length 2_n (where $2_0 = 1$ and $2_{n+1} = 2^{2_n}$) (see [36]

☆ This work was supported by the projects P22028-N13, I603-N18 and P25160-N25 of the Austrian Science Fund (FWF), by the ERC Advanced Grant ProofCert and the WWTF Vienna Research Group 12-04.

* Corresponding author.
E-mail addresses: stefan.hetzl@tuwien.ac.at (S. Hetzl), leitsch@logic.at (A. Leitsch), giselle@logic.at (G. Reis), weller@logic.at (D. Weller).

http://dx.doi.org/10.1016/j.tcs.2014.05.018
0304-3975/

and [31]). The length of a proof plays an important role in many situations such as human readability, space requirements and time requirements for proof checking. For most of these situations general-purpose data compression methods cannot be used as the compressed representation would need to be uncompressed for checking proof theoretical properties. It is therefore of high practical interest to develop proof-search methods which produce non-analytic and hence potentially much shorter proofs. In the method presented in this paper we start with a cut-free proof and abbreviate it by computing useful cuts based on a structural analysis of the cut-free proof.

There is another, more theoretical, motivation for introducing cuts which derives from the foundations of mathematics: most of the central mathematical notions have developed from the observation that many proofs share common structures and steps of reasoning. Encapsulating those leads to a new abstract notion, like that of a group or a vector space. Such a notion then builds the base for a whole new theory whose importance stems from the pervasiveness of its basic notions in mathematics. From a logical point of view this corresponds to the introduction of cuts into an existing proof database. While we cannot claim to contribute much to the understanding of such complex historical processes by the current technical state of the art, this second motivation is still worthwhile to keep in mind, if only to remind us that we are dealing with a difficult problem here.

Gentzen's method of cut-elimination is based on reductions of cut-derivations (subproofs ending in a cut), transforming them into simpler ones; basically the cut is replaced by one or more cuts with lower logical complexity. A naive reversal of this procedure is infeasible as it would lead to a search tree which is exponentially branching on some nodes and infinitely branching on others. Therefore we base our procedure on a deeper proof-theoretic analysis: in the construction of a Herbrand sequent S' corresponding to a cut-free proof φ' (see e.g. [3]) obtained by cut-elimination on a proof φ of a sequent S with cuts, only the *substitutions* generated by cut-elimination on quantified cuts are relevant. In fact, it is shown in [18] that, for proofs with Σ_1 and Π_1-cuts only, S' can be obtained just by computing the substitutions defined by cut-elimination without applying Gentzen's procedure as a whole. Via the cuts in the proof φ one can define a tree grammar generating a language consisting exactly of the terms (to be instantiated for quantified variables in S) for obtaining the Herbrand sequent S' [18]. Hence, generating a tree grammar G from a set of Herbrand terms T (generating T) corresponds to an inversion of the quantifier part of Gentzen's procedure. The computation of such an inversion forms the basis of the method of *cut-introduction* presented in this paper. Such an inversion of the quantifier part of cut-elimination determines which instances of the cut-formulas are used but it does not determine the cut-formulas. In fact, a priori it is not clear that every such grammar can be realized by actual cut-formulas. However, we could show that, for any such tree grammar representing the quantifier part of potential cut-formulas, actual cut-formulas can be constructed. Finally, a proof containing these cut-formulas can be constructed.

Work on cut-introduction can be found at a number of different places in the literature. Closest to our work are other approaches which aim to abbreviate or structure *a given input proof*: [41] is an algorithm for the introduction of atomic cuts that is capable of exponential proof compression. The method [13] for propositional logic is shown to never increase the size of proofs more than polynomially. Another approach to the compression of first-order proofs by introduction of definitions for abbreviating terms is [40].

Viewed from a broader perspective, this paper should be considered part of a large body of work on the generation of non-analytic formulas that has been carried out by numerous researchers in various communities. Methods for lemma generation are of crucial importance in inductive theorem proving which frequently requires generalization [7], see e.g. [25] for a method in the context of rippling [8] which is based on failed proof attempts. In automated theory formation [9,10], an eager approach to lemma generation is adopted. This work has, for example, led to automated classification results of isomorphism classes [34] and isotopy classes [35] in finite algebra. See also [28] for an approach to inductive theory formation. In pure proof theory, an important related topic is Kreisel's conjecture (see footnote 3 on page 400 of [38]) on the generalization of proofs. Based on methods developed in this tradition, [4] describes an approach to cut-introduction by filling a proof skeleton, i.e. an abstract proof structure, obtained by an inversion of Gentzen's procedure with formulas in order to obtain a proof with cuts. The use of cuts for structuring and abbreviating proofs is also of relevance in logic programming: [30] shows how to use focusing in order to avoid proving atomic subgoals twice, resulting in a proof with atomic cuts.

This paper is organized as follows:

In Section 3 we define Herbrand sequents and *extended* Herbrand sequents which represent proofs with cut. The concept of rigid acyclic regular tree grammars is applied to establish a relation between an extended Herbrand sequent S^* and a (corresponding) Herbrand sequent S': the language defined by this grammar is just the set of terms T to be instantiated for quantifiers in the original sequent S to obtain S'. Given such a grammar G generating T there exists a so-called *schematic extended Herbrand sequent* $\hat{S}$ in which the (unknown) cut-formulas are represented by monadic second-order variables. It is proved that $\hat{S}$ always has a solution, the *canonical solution*. From this solution, which gives an extended Herbrand sequent and the cut-formulas for a proof, the actual proof with these cuts is constructed.

To make the underlying methods more transparent, Section 3 deals only with end-sequents of the form $\forall x\, F \rightarrow$. In Section 4 the method is generalized to sequents of the form

$$\forall \bar{x}_1 F_1, \ldots, \forall \bar{x}_n F_n \rightarrow \exists \bar{y}_1 G_1, \ldots, \exists \bar{y}_m G_m$$