Space & Congruence Compression of Proofs

ANTRAG

Diplomarbeit

im Rahmen des Studiums

European Master in Computational Logic

eingereicht von

Andreas Fellner, BSc
Matrikelnummer 0825918

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ. Prof. Dr.phil. Alexander Leitsch
Mitwirkung: Dr. techn. Bruno Woltzenlogel Paleo

Wien, 18. August 2014

(Unterschrift Andreas Fellner, BSc) (Unterschrift Betreuung)

Space & Congruence Compression of Proofs

PROPOSAL

Master thesis

in

European Master in Computational Logic

by

Andreas Fellner, BSc
Registration Number 0825918

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ. Prof. Dr.phil. Alexander Leitsch
Assistance: Dr. techn. Bruno Woltzenlogel Paleo

Vienna, 18. August 2014

(Signature of Author) (Signature of Advisor)

Abstract

Problem definition

Proofs are the backbone of mathematics. They allow scientists to build theorems on top of another and thus discover new knowledge. Proofs not only serve as stepping stones, they can also provide insight on the nature of the underlying problem.

Both statements are true for formal proofs as well. Formal proofs allow system to trust the output of other systems and therefore they can safely be built on top of another. For example SAT-Solvers are used extensively in modern deductive systems [**?**]. However, solvers may contain bugs. Therefore their output can not be trusted blindly. A formal proof can assure the correctness of the output. Formal proofs not only help in combining systems, they can also be used to obtain information about the underlying problem. For example interpolants, which have important applications in Verification and Synthesis of programs [**?**], can be extracted from formal proofs [**?**]. From hereon, we mean formal proofs when speaking of proofs.

Typically problems that are tackled by automated systems are large. As a consequence proofs produced during the process are large. So large that even for proof processing algorithms with low complexity, it is highly desirable to reduce the hardness of the input, while maintaining the quality of its conclusion. Proof processing algorithms could be correctness checking, information extraction or proof manipulating techniques. That is the problem that our work tackles. We present methods to compress proofs, produced by SMT- or SAT- Solvers, w.r.t. two different measures.

The first measure is *length*. The length of a proof is the number of inferences. For example the length of the resolution proof is the number of resolution applications. The congruence reasoning part of SMT-proofs has often been found to be redundant. Congruence reasoning derives equations of terms that are implied by a given set of input equations, using the four axioms *reflexivity*, *symmetry*, *transitivity* and *congruence*. It can be redundant in the sense that subsets of the input may suffice to derive certain equations.

The second measure is *space*. Typically proofs are represented as directed acyclic graphs. The space of a proof p and a traversal order $\prec$ is the maximal number of nodes of that graph that have to be kept in memory at once, while processing p following $\prec$. The goal is to construct traversal orders for proofs with small space measures.

State of the art

The research field of proof complexity studies lower bounds of various measures of proofs in different proof calculi [?, ?, ?]. A typical questions of this fields intuitively is of the following kind. Given a proof calculus $\mathcal{C}$, find a sequence of theorems $(t_1, \ldots, t_n, \ldots)$ such that $\mathrm{length}(p(t_m)) \in \Omega(2^m)$, where $p(t)$ is a shortest proof of t in $\mathcal{C}$. One classical result is the worst case exponential length of resolution refutations in the propositional resolution calculus [?]. Besides the classical length measure, space requirements of proofs have been studied [?, ?, ?, ?, ?] using pebbling games. The problem of finding optimal strategies in the variant of the pebbling game that is most relevant to us is NP-complete [?].

Our approach differs from this field of study, as we do not mean to prove new lower bounds for a class of problems or calculi, but to provide concrete algorithms to reduce measures of given proofs.

For Propositional Logic many length compression algorithms have been proposed [?, ?, ?, ?, ?]. On the other hand, for First Order Logic Cut-Introduction has been studied [?], which is a form of proof compression. However, none of the approaches deal with the congruence reasoning of SMT-proofs. Congruence reasoning has been long studied and classical congruence closure algorithms are those of Nelson and Oppen [?], Downey, Sethi and Tarjan [?] and Shostak [?]. More recently abstract congruence closure has been proposed [?], which replaces subterms with fresh constants to simplify the algorithms and increase performance. Congruence closure algorithms producing explanations have been proposed by Pascal Fontaine [?] and Nieuwenhuis-Oliveras [?, ?]. The problem of deciding whether from a given set of equations, there is an explanation for one particular equation of size k is believed to be NP-complete [?, ?]. However, this result has not been proven yet.

Space compression of proofs is a young branch of proof compression and has not been studied extensively yet. To the best of our knowledge neither an algorithm to compress proofs in space nor one to construct strategies in pebbling games has been proposed so far. The DRUP proof format [?] is an extension of the well known RUP format, with the addition of deletion information. Deletion information indicates when nodes can be dropped from memory. The format has been introduced to help cope with huge proofs produced by solvers. For example [?] reports of a 13GB proof in the DRUP format for one case of the Erdős Discrepancy Conjecture.

Methodology and approach

We will propose algorithms for compressing proofs and show their complexity. The algorithms will be implemented in the proof compression software Skeptik [?] in the programming language Scala. We will evaluate the algorithms on a broad range of proofs produced by SAT- and SMT-Solvers to show the actual compression obtained. The experiments will be carried out on the Vienna Scientific Cluster. To remove redundancies in the congruence reasoning part of SMT-proofs, we will need two algorithms.

A congruence closure algorithm that is able to produce explanations in the form of proofs. We will not use the ideas of abstract congruence closure, because we want to reduce the number of literals and therefore do not want to introduce new constants. Even though the new constants can be removed from proofs, in our setting the algorithm will be applied many times to small instances and not the other way around. Therefore we think that the overhead of dealing with the extra constants is not worth the possible performance edge [?] that abstract congruence closure can have when looking at single instances. To obtain short proofs with short conclusions, we think that shortest path algorithms like the one of Dijkstra [?, ?] will be useful.

The second algorithm manipulates the proof itself. It applies the congruence closure algorithm to the conclusions of subproofs reasoning about equality in order to find and remove redundancies. The algorithm also fixes proof nodes with premises that have been changed by it earlier, similar to how it is described in [?]. There can be a trade-off between the size of the proof and the size of its conclusion. A shorter conclusion does not necessarily correspond to a shorter proof, if the original proof reuses some nodes many times. However, shorter conclusions cause fewer inference steps further down the proof. Intuitively shorter conclusions should always be better, but the relationship between proof- and conclusion size has to be investigated.

The algorithms to compress proofs in the space measure construct traversal orders using heuristics. Traversal orders correspond to strategies in pebbling games. The reason for the use of heuristics is the NP-completeness of finding optimal strategies in a variant of the pebble game [?] and the implied infeasibility of constructing the optimal strategy. There will be two algorithms to construct traversal orders. One operates on proofs in a Top-Down fashion, i.e. from the axioms towards the root, which corresponds to playing the pebbling game. The other one operates on proofs in a Bottom-Up fashion, i.e. from the root towards the axioms, constructing orders by recursively queuing up premises. The heuristics use characteristics of proof nodes, such as the number of children.

Theorems stating the complexity of our algorithms will have to be proven from scratch or adapted from earlier publications.

We want to show the NP-completeness of the shortest explanation problem. To this end two things have to be done. First we need to find an algorithm that, given an Oracle to make correct decisions, finds an explanation of k or fewer equations in polynomial time. Second we need to reduce another NP-complete problem to the shortest explanation problem. The Palest Path Problem [?] is one option for reduction, that has been foreseen to possibly be fruitful by the source of the claim in [?, ?].

Expected results

We will enrich the field of proof compression by new algorithms. It is hard to predict the compression achieved by the congruence reasoning algorithm. Good propositional proof compression techniques usually achieve between 10% and 20% compression. Such numbers would be nice to see for our method. The space measure of a proof is always relative to a traversal order, therefore orders have to be compared to each other. We hope to show that one method or heuristic is particularly better than the others. The proof of NP-completeness of the shortest explanation problem is a gap in science, which we aim to fill.

Kurzfassung

Diese Arbeit befasst sich mit der Komprimierung von formalen Beweisen. Formale Beweise sind von großer Bedeutung in der modernen Informatik. Sie können für den sicheren Austausch von deduktiven System verwendet werden. Des Weiteren können aus ihnen Informationen, wie etwa Interpolanten [?], extrahiert werden, welche zur Lösung eines Problems beitragen [?].

Formale Beweise sind typischerweise sehr groß, siehe etwa [?] für einen 13GB Beweis eines Falles der Erdős Discrepancy Conjecture. Bei solchen Beweisgrößen stoßen Computersysteme an ihre Grenzen und deswegen ist es erforderlich Beweise zu komprimieren. Unsere Arbeit präsentiert zwei Methoden zur Beweiskomprimierung.

Die erste Methode entfernt Redundanzen im Kongruenzteil von SMT-Beweisen. Kongruenzbeweise schließen von einer Menge an Gleichungen auf neue Gleichungen mit der Voraussetzung der vier Axiome: *Reflexivität*, *Symmetrie*, *Transitivität* und *Kongruenz*. Beweise, die von SMT-Solvern erzeugt werden, schließen oft auf neue Gleichungen aus einer unnötig großen Menge. Wir wollen kleinere Mengen finden, die für den Beweis der selben Aussage ausreichen und somit redundante Beweise durch kürzere ersetzen. Außerdem werden wir die NP-Vollständigkeit des Problems der kürzesten Erklärung einer Gleichung beweisen.

Die zweite Methode untersucht die Speicherplatzanforderungen von Beweisen. Beim Bearbeiten von Beweisen muss nicht der gesamte Beweis zu jeder Zeit im Speicher gelagert werden. Teilbeweise werden erst in den Speicher geladen, wenn sie benötigt werden und werden wieder aus diesem entfernt, sobald sie nicht mehr benötigt werden. In welcher Ordnung die Teilbeweise geladen werden, ist essentiell für die maximale Speicherplatzanforderung. Wir wollen Ordnungen mit niedrigen Speicherplatzanforderungen mit Hilfe von Heuristiken konstruieren.