FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

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: Bruno Woltzenlogel Paleo, MSc

Wien, 10. April 2014

(Unterschrift Andreas Fellner, BSc)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

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: Bruno Woltzenlogel Paleo, MSc

Vienna, 10. April 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 other theorems 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 also true for formal proofs, which are particularly interesting to Computer Science. Formal proofs allow system to trust the output of another and therefore systems can safely be built on top of another. For example SAT-Solvers are used extensively in modern deductive systems [6]. However solvers may contain bugs, therefore it may be dangerous to blindly trust their output. 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 [22], can be extracted from formal proofs [19]. From hereon, we mean formal proofs when talking about 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 been found to be redundant often. 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 [1, 4, 9]. 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 $\text{length}(p(t_m)) \in \Omega(2^m)$, where $p(t)$ is the shortest proof of t in $\mathcal{C}$. One classical result is the worst case exponential length of resolution refutations in the propositional resolution calculus [1]. Besides the classical length measure, space requirements of proofs have been studied [5, 14, 20, 26, 28] 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 [28].

Our approach differs from this field of study, as we do not mean to prove new lower bounds for a classes 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 [3, 7, 8, 11, 16]. On the other hand, for First Order Logic Cut-Introduction has been studied [17], 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 [23], Downey, Sethi and Tarjan [13] and Shostak [29]. More recently abstract congruence closure has been proposed [2], which replaces subterms with fresh constants to simplify the algorithms and increase performance. Congruence closure algorithms producing explanations have been proposed by Pascal Fontaine [15] and Nieuwenhuis-Oliveras [24, 25], which uses the ideas of abstract congruence closure. 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 [24, 25]. However this result has not been proven yet.

Space compression of proofs is a rather new 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 [18] 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 [21] 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 into the proof compression software Skeptik [27] 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:

The first is a congruence closure algorithm that is able to produce explanations, such as the algorithms cited above. 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 explanations, 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 [2] that abstract congruence closure can have when looking at single instances. To obtain short explanations, we think that shortest path algorithms like the one of Dijkstra [10, 12] will be useful. The algorithm returns a proof corresponding to the shortest explanation it finds.

The second is an algorithm manipulates the proof itself. It applies the congruence closure algorithm to the conclusions of subproofs reasoning about equality in order to find redundancies. There can be a trade-off between the size of the proof and the size of the explanation. A shorter explanation does not necessarily correspond to a shorter proof, if the proof corresponding to the longer explanation reuses some nodes many times. However shorter explanations lead to stronger conclusions and to less inference steps further down the proof. Intuitively shorter explanations should always be better, but the relationship between proof- and explanation size has to be investigated. The algorithm also fixes proof nodes with premises that have been changed by it earlier, similar to how it is described in [8].

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 heuristics is the NP-completeness of finding optimal strategies in a variant of the pebble game [28] 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 less equations in polynomial time. Second we need to reduce another NP-complete problem to the shortest explanation problem. The Palest Path Problem [30] is one option for reduction, that has been foreseen to possibly be fruitful by the source of the claim in [24, 25].

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 with 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. Andererseits können aus ihnen Informationen, wie etwa Interpolants [22], extrahiert werden, welche zur Lösung eines Problems beitragen [19]. Formale Beweise sind typischerweise sehr groß, siehe etwa [21] 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-completeness 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.

Bibliography

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[2] Leo Bachmair and Ashish Tiwari. Abstract congruence closure and specializations. In *CADE*, pages 64–78, 2000.

[3] Omer Bar-Ilan, Oded Fuhrmann, Shlomo Hoory, Ohad Shacham, and Ofer Strichman. Linear-time reductions of resolution proofs. In *Haifa Verification Conference*, pages 114–128, 2008.

[4] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present and future. *Bulletin of the EATCS*, 65:66–89, 1998.

[5] Eli Ben-Sasson. Size space tradeoffs for resolution. In *STOC*, pages 457–464, 2002.

[6] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[7] Roderick Bloem, Sharad Malik, Matthias Schlaipfer, and Georg Weissenbacher. Reduction of resolution refutations and interpolants via subsumption.

[8] Joseph Boudou and Bruno Woltzenlogel Paleo. Compression of propositional resolution proofs by lowering subproofs. In *TABLEAUX*, pages 59–73, 2013.

[9] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[10] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989.

[11] Scott Cotton. Two techniques for minimizing resolution proofs. In *SAT*, pages 306–312, 2010.

[12] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[13] Peter J. Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. *J. ACM*, 27(4):758–771, 1980.

[14] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Inf. Comput.*, 171(1):84–97, 2001.

[15] Pascal Fontaine. *Techniques for verification of concurrent systems with invariants*. PhD thesis, PhD thesis, Institut Montefiore, Université de Liege, Belgium, 2004.

[16] Pascal Fontaine, Stephan Merz, and Bruno Woltzenlogel Paleo. Compression of propositional resolution proofs via partial regularization. In *CADE*, pages 237–251, 2011.

[17] Stefan Hetzl, Alexander Leitsch, and Daniel Weller. Towards algorithmic cut-introduction. In *LPAR*, pages 228–242, 2012.

[18] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *FMCAD*, pages 181–188, 2013.

[19] Georg Hofferek, Ashutosh Gupta, Bettina Könighofer, Jie-Hong Roland Jiang, and Roderick Bloem. Synthesizing multiple boolean functions using interpolation on a single proof. *CoRR*, abs/1308.4767, 2013.

[20] John E. Hopcroft, Wolfgang J. Paul, and Leslie G. Valiant. On time versus space. *J. ACM*, 24(2):332–337, 1977.

[21] Boris Konev and Alexei Lisitsa. A sat attack on the erdos discrepancy conjecture. *CoRR*, abs/1402.2184, 2014.

[22] Kenneth L. McMillan. Applications of craig interpolants in model checking. In *TACAS*, pages 1–12, 2005.

[23] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, 1980.

[24] Robert Nieuwenhuis and Albert Oliveras. Proof-producing congruence closure. In *RTA*, pages 453–468, 2005.

[25] Robert Nieuwenhuis and Albert Oliveras. Fast congruence closure and extensions. *Inf. Comput.*, 205(4):557–580, 2007.

[26] Jakob Nordström. Pebble games, proof complexity, and time-space trade-offs. *Logical Methods in Computer Science*, 9(3), 2013.

[27] Bruno Woltzenlogel Paleo, Joseph Boudou, and Andreas Fellner. Skeptik system description.

[28] Ravi Sethi. Complete register allocation problems. *SIAM J. Comput.*, 4(3):226–248, 1975.

[29] Robert E. Shostak. An algorithm for reasoning about equality. *Commun. ACM*, 21(7):583–585, 1978.

[30] Ashish Tiwari. Analyzing selected network optimization problems under the color cost metric.