FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Space & Congruence Compression of Proofs

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, Dr.

Wien, 4. Mai 2014

(Unterschrift Andreas Fellner, BSc)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Space & Congruence Compression of Proofs

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, Dr.

Vienna, 4. Mai 2014

(Signature of Author) (Signature of Advisor)

Abstract

This work is about compression of formal proofs. Formal proofs are of great importance to modern computer scienece. They can be used to combine deductive systems. For example SAT- Solvers [1] are heavily used for all kinds of computations, because of their efficiency. A formal proof is a certificate of the correctness of the output of a SAT- Solver. Furthermore, from formal proofs information can be extracted about some underlying problem. For example, Interpolants [4] can be extracted from proofs, as done in [2].

Typically problems tackled by automated systems are huge. Therefore the produced proofs are huge. For example, [3] reports about a 13 GB proof of one case of the Erdős Discrepancy Conjecture. With such proof sizes, computer system reach their boundaries and that is why it is necessary to compress proofs. Our work presents two methods for proof compression.

The first method removes redundancies in the congruence part of SMT- proofs. Congruence reasoning deduces equations from a set of given equations, using the four axioms *reflexivity*, *symmetry*, *transitivity*, and *congruence*. We found that SMT- Solver often use an unnecessarily big set of input equations to deduce one particular equality. We want to find smaller sets of equations, that suffice to proof the same result and therefore replace subproofs with shorter ones. Furthermore, we will proof the NP - Completeness of the problem of finding the shortest explanation of one equation within a set of input equations.

The second method investigates the memory requirements of proofs. While processing a proof, not all parts of the proof have to be kept in memory at all times. Subproof can be loaded into memory when needed and can be removed from memory again when they are not. In which traversal order subproofs are visited is essential to the maximum memory consumption during proof processing. We want to construct traversal orders with low memory requirements using heuristics.

Kurzfassung

Diese Arbeit befasst sich mit der Komprimierung von formalen Beweisen. Formale Beweise sind von großer Bedeutung in der modernen Informatik. Sie können verwendet werden um deduktive Systeme miteinander zu kombinieren. Ein Beispiel sind SAT- Solver [1], welche ob ihrer Effektivität gerne für diverse Berechnungen verwendet werden. Ein formaler Beweis kann als Zertifikat für die Korrektheit des Ergebnisses eines SAT- Solvers dienen. Des Weiteren können aus ihnen Informationen, wie etwa Interpolants [4], extrahiert werden, welche zur Lösung eines Problems beitragen [2].

Formale Beweise sind typischerweise sehr groß, siehe etwa [3] 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 Vorraussetzung 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.

CHAPTER 1

Introduction

1.1 General Information

This document is intended as a template and guideline and should support the author in the course of doing the master's thesis. Assessment criteria comprise the quality of the theoretical and/or practical work as well as structure, content and wording of the written master's thesis. Careful attention should be given to the basics of scientific work (e.g., correct citation).

1.2 Organizational Issues

A master's thesis at the Faculty of Informatics has to be finished within six months. During this period regular meetings between the advisor(s) and the author have to take place. In addition, the following milestones have to be fulfilled:

1. Within one month after having fixed the topic of the thesis the master's thesis proposal has to be prepared and must be accepted by the advisor(s). The master's thesis proposal must follow the respective template of the dean of academic affairs. Thereafter the proposal has to be applied for at the deanery. The necessary forms may be found on the web site of the Faculty of Informatics. `http://www.informatik.tuwien.ac.at/dekanat/formulare.html`

2. Accompanied with the master's thesis proposal, the structure of the thesis in terms of a table of contents has to be provided.

3. Then, the first talk has to be given at the so-called "Seminar for Master Students". The slides have to be discussed with the advisor(s) one week in advance. Attendance of the "Seminar for Master Students" is compulsory and offers the opportunity to discuss arising problems among other master students.

4. At the latest five months after the beginning, a provisional final version of the thesis has to be handed over to the advisor(s).

5. As soon as the provisional final version exists, a first poster draft has to be made. The making of a poster is a compulsory part of the "Seminar for Master Students" for all master studies at the Faculty of Informatics. Drafts and design guidelines can be found at `http://www.informatik.tuwien.ac.at/studium/richtlinien`.

6. After having consulted the advisor(s) the second talk has to be held at the "Seminar for Master Students".

7. At the latest six months after the beginning, the corrected version of the master's thesis and the poster have to be handed over to the advisor(s).

8. After completion the master's thesis has to be presented at the "epilog". For detailed information on the epilog see:
`http://www.informatik.tuwien.ac.at/studium/epilog`

1.3 Structure of the Master's Thesis

If the curriculum regulates the language of the master's thesis to be English (like for "Business Informatics"), the thesis has to be written in English. Otherwise, the master's thesis may be written in English or in German. The structure of the thesis is predetermined. The table of contents is followed by the introduction and the main part, which can vary according to the content. The master's thesis ends with the bibliography (compulsory) and the appendix (optional).

- Cover page
- Acknowledgements
- Abstract of the thesis in English and German
- Table of contents
- Introduction
  - motivation
  - problem statement (which problem should be solved?)
  - aim of the work
  - methodological approach
  - structure of the work
- State of the art / analysis of existing approaches
  - literature studies

  - analysis
  - comparison and summary of existing approaches
- Methodology
  - used concepts
  - methods and/or models
  - languages
  - design methods
  - data models
  - analysis methods
  - formalisms
- Suggested solution/implementation
- Critical reflection
  - comparison with related work
  - discussion of open issues
- Summary and future work
- Appendix: source code, data models, ...
- Bibliography

CHAPTER 2

Resolution

Resolution

Resolution

CHAPTER 3

Proof Compression

Resolution

Resolution

CHAPTER 4

Pebbling

Pebbling

Pebbling

CHAPTER 5

Congruence

Congruence

Congruence

CHAPTER 6

Conclusion

Conclusions

Conclusions

Bibliography

[1] 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.

[2] 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.

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

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