1

# Final Project Submission Guidelines

CNF Clause Trimming with Gröbner Bases via Term Ordering

S. Clegg1, M. Huff2, D. Janibagian3

ECE Dept., Univ. of Utah

spencermclegg@gmail.com1,ocarinahuff@msn.com2, drew.janibagian@gmail.com3

Abstract

This project proposes a CNF SAT transformation algorythm employing a unique partitioning technique followed by efficient monomial ordering, polynomial to CNF conversion, and finally, run through a SAT solver engine. Gr¨obner bases as a means to analyze the problem structure. The excercise will follow similar methods of C. Condrat and P. Kalla, “A Grobner Basis Approach to CNF formulae Preprocessing,".

1. The Project Report

Each group will submit a project report. I am not going to enforce any page limits, but chances are that if the report is longer than 15 pages (single column), it is probably too long; a 5 page report is probably too short. Make your own judgment about the length of the report. Here’s a guideline, I repeat, just a guideline, on how to write your report.

* First of all, give your project a descriptive title. “ECE 6745: Final Project Report” should NOT be the title, it may be the sub-title. Authorship and your email addresses should be included in the title page.
* Provide a 1-paragraph abstract or summary of the project.
* Organize the report into sections. For example, you may include sections for:
  + Introduction: What problem are you solving? Why is it important, what applications are you targeting?
  + Previous work (published papers or books that you studied)
  + Preliminaries and Notation used
  + Theory that you studied or experimented with
  + Algorithm implementation and the tools that you developed. You may provide a methodology or design flow if you integrated a varied set of tools (e.g., Read RTL → parse into SMT-Lib → invoke solver → read solutions from solver into MILP → . . .
  + Experimental results: How did you setup the experiments, what parameters did you use for variable

orders, term orders, bound on SAT solver restarts, etc. Present your experiments in a Tabular (or a graphical) format that compares various execution statistics such as solving time, number of iterations, number of conflicts/restarts (if using SAT), and so on.

2

* + Conclusions: Draw your own conclusions from your experiments, and conclude the paper.
  + Your report should have a bibliography.
* You may include an appendix if you want me to go through any supplementary material.
* Please run a spell-checker before your submit your report.

There is no need to describe in detail the basics. For example, there is no need to define Ideals, Varieties, Gröbner bases, Buchberger’s algorithm, or the entire conflict analysis procedure of the SAT solver – all from scratch. Just specify the notation properly: For example, you may just say “We denote by Z2k the finite ring of integers (mod 2k), and R = Z2k [x1, . . . , xn] the polynomial ring. Ideal J = (f1, . . . , fs) ⊆ R, and let V(J) be the variety. A Gröbner basis GB(J) can be computed using Möller’s algorithm over integers . . . ”, and so

on.

If you have implemented a specific algorithm, please describe it in pseudo-code nicely. For example, if you are modifying Buchberger’s algorithm to do something specific for your project, then you may describe it as follows:

Require: F = {f1, . . . , fs} ⊂ F[x1, . . . , xn], fi /= 0

Ensure: H = {h1, . . . , hl}, polynomials with special properties

1: Initialize: G := F; G := {{fi, fj} | fi /= fj ∈ G}

2: while G /= ∅ do

3: Pick a pair {f, g} ∈ G

4: G := G − {{f, g}}

5: Spoly(f, g) −→G + h

6: if h /= 0 then

7: G := G ∪ {{u, h} | ∀u ∈ G}

8: G := G ∪ {h}

9: end if

10: if h has some nice property then

11: H = H ∪ {h}

12: end if

13: end while

Algorithm 1 Kalla’s algorithm for groovy analysis

Describe your modification to the basic algorithm: “In the algorithm, lines 10-12 check if h is awesome. If it is, we make a note of it...” Please feel free to make use of examples and figures. Be creative!

1. The rest of the submission

Along with your report, you should include all the tools and scripts that you have built. No need to include existing/public domain tools such as Singular, Gurobi, PicoSAT, ABC, etc. Include only the scripts that you developed, such as your Perl scripts, or Singular Code. If you worked on ZBDDs, and your code spans 2-3 C/C++ files, include those files along with instructions on how to compile them with CUDD-version. Include a README that provides instructions on how to execute and re-create your experiments.

Include the benchmarks/data that you experimented with.

3

*A. Submission*

Create a directory “your-name-project”. This directory should contain subdirectories: i) source-code; ii) project report; iii) benchmarks; iv) anything else you want me to have. You should tar/zip it and either

email it to me or share it via some cloud service. If your report is written in LATEX, feel free to send me the source Q. You don’t have to, but it is your choice.

Please submit everything before close of business Friday Dec 19. Any other issues, please just ask.