

MIT Doctoral Dissertation
PUBLISH ABSTRACT ONLY
INFORMATION_____
Abstract No.***DO NOT WRITE IN THIS SPACE***

Vol/Issue _____

School Code _____

Advisor _____

PLEASE TYPE OR PRINT

PERSONAL DATA

1. Full name (as it appears on dissertation title page)

(last) (first) (middle)

2. Year of birth (optional) _____

3. Present mailing address _____
-
- _____

Future mailing address _____

Effective date of future mailing address _____

Home telephone _____ Business telephone _____

DOCTORAL DEGREE DATA

4. Full name of university conferring degree
- Massachusetts Institute of Technology

5. Degree awarded (check one) Ph.D. Sc.D.

6. Year degree awarded _____

7. IMPORTANT: Attach a copy of your dissertation title page and abstract to this form. Please be certain that the name of your dissertation supervisor is included on both.

8. Subject categories for your dissertation. Enter 4-digit code from 'Subject Categories' list found on the opposite side of this form, and write in the category selected. You may enter two additional codes and categories on the lines provided.

Code _____ Category _____

Code _____ Category _____

Code _____ Category _____

(Optional) List up to five additional words from your dissertation not already found in ***either*** your title ***or*** abstract which would be useful for database access.

a. _____ b. _____ c. _____

d. _____ e. _____

[**TODO:** Run ‘make update-thesis’ before submission to update the date on the cover page]

[**TODO:** change `\finalfalse` to `\finaltrue`]

[**TODO:** Make sure to exclude no files (not rewriting)]

Performance Engineering of Proof-Based Software Systems at Scale

by

Jason S. Gross

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2021

© Jason S. Gross, MMXXI. All rights reserved.

The author hereby grants to MIT permission to reproduce and to
distribute publicly paper and electronic copies of this thesis document
in whole or in part in any medium now known or hereafter created.

Author
Department of Electrical Engineering and Computer Science
August 19, 2015

Certified by
Adam Chlipala
Associate Professor of Computer Science
Thesis Supervisor

Accepted by
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Performance Engineering of Proof-Based Software Systems at Scale

by
Jason S. Gross

Submitted to the Department of Electrical Engineering and Computer Science
on August 19, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

[**TODO: More formal wording**] Formally verified proofs are important. Unfortunately, large-scale proofs, especially automated ones, in Coq, can be quite slow.

This thesis aims to be a partial guide to resolving the issue of slowness. We present a survey of the landscape of slowness in Coq, with a number of micro- and macro-benchmarks. We describe various metrics that allow prediction of slowness, such as term size, goal size, and number of binders, and note the occasional surprise-lack-of-bottleneck for some factors, such as total proof term size.

We identify three main categories of workarounds and partial solutions to slowness: design of APIs of Gallina libraries; changes to Coq’s type theory, implementation, or tooling; and automation design patterns, including proof by reflection. We present lessons drawn from the case-studies of a category-theory library, a proof-producing parser-generator, and a verified compiler and code generator for low-level cryptographic primitives.

[**TODO: Fix run-on sentence**] The central new contribution presented by this thesis, beyond hopefully providing a roadmap to avoid slowness in large Coq developments, is a reflective framework for partial evaluation and rewriting which, in addition to being used to compile a code-generator for field arithmetic cryptographic primitives which generates the code currently used in Google Chrome, can serve as a template for a possibly replacement for tactics such as `rewrite`, `rewrite_strat`, `autorewrite`, `simpl`, and `cbn` which achieves much better performance by running in Coq’s VM while still allowing the flexibility of equational reasoning.

[**TODO: Maybe instead use the alternative from the thesis-proposal?**] The proposed research is a study of performance issues that come up in engineering large-scale proof-based systems in Coq. The thesis presents lessons learned about achieving acceptable performance in Coq in the course of case-studies on formalizing category theory, developing a parser synthesizer, and constructing a verified compiler for synthesizing efficient low-level cryptographic primitives. We also present a novel method of simple and fast reification, and a prototype tool for faster rewriting and customizable

reduction which does not require extending Coq's trusted code base.

Thesis Supervisor: Adam Chlipala

Title: Associate Professor of Computer Science