CIS400/401 Project Proposal Specification [Verification of System FC in Coq]

Dept. of CIS - Senior Design 2014-2015*

Tiernan Garsys
tgarsys@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Tayler Mandel
tmandel@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Lucas Peña
lpena@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Noam Zilberstein
noamz@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

We plan to verify a formalized version of System FC, the core language of the Glascow Haskell Compiler (GHC), using the Coq proof assistant. We will then prove a translation from our formal language to the actual implementation of System FC that is used in GHC. The goals of verification are to prove that the evaluation semantics of System FC are type safe.

There are two main benefits to this project. First, the verification would provide assurance regarding the safety and accuracy of GHC. Second, and perhaps more importantly, it will provide foundation to verify other properties of GHC such as compiler optimization.

1. INTRODUCTION

Haskell has one of the strongest type systems of any mainstream programming language, with features such as Type Families, Typeclasses, and Generalized Algebraic Datatypes. When writing Haskell, there are a lot of guarantees of correctness encoded in the type system. We wish to ensure that this type safety of features like these is preserved in System FC, the GHC core language. We will do this by proving the progress and preservation theorems using our formalized definition of System FC.

Progress and preservation are the most basic indications of safety for any type system. More specifically, progress states that a well-typed term is either a value or can take a step of evaluation. Preservation indicates that if a well-typed term takes a step, the resulting term will still be well-typed (TODO: Cite TAPL). When formalizing System FC, we will prove both progress and preservation for each feature added.

System FC is built on top of the simpler language System F. System F, also known as the polymorphic lambda calculus, is an extension of the simply-typed lambda calculus to include the abstraction and application of type terms. This feature essentially allows for functions to take types as parameters, granting the ability to define functions whose actual types vary based on these input types. We will first formally verify System F in Coq and then we will add the additional features needed to transform System F into a full formalization of System FC. These features include type coercion, type families, and type constructors. Once we have added these features to System F, we will have a formalized language that is equivalent to System FC. We will then be able to prove a translation to System FC which will show that we have indeed verified the core language of GHC.

The formally verified version of System FC will pave the way for future work in formal verification. Once the core language is verified, it will be possible to verify that translations of the language preserve types and semantic meaning. Such translations include compiler optimizations performed by GHC. Compiler optimizations can introduce compiler bugs, so the ability to verify optimizations could make several important in verifying the entirety on the compiler.

2. RELATED WORK

Perhaps the most important section of your proposal is *related work*. Here you demonstrate that you have read and understand what others in the field have done. This ensures you (1) know the state-of-the-art, (2) are not re-doing others work, and (3) you know the performance levels you must achieve to make a contribution. As you discuss each related work, make note of how each has advanced the field. Keep in mind that this section should not read like a regular research paper you write for other classes. In other words, you should not just discuss related work for the sake of having a related work section; rather, tell a story about the state-of-the-art of the field and where your work fits in.

This section should have in-line citations to your bibliography (really all sections should have citations, but we expect them to be most dense in this section). We are going to require that your proposal has at least 6 references. Fortunately, LaTeX makes citations easy. Your TA has had no difficulty, as the work of Ivanov *et al.* [2] demonstrates. Need help with LaTeX? Be sure to check out [3] and [1], two helpful on-line resources.

What defines a good resource? Wikipedia is **NOT** a good resource. We would like to see references from academic journals/conferences (ACM, IEEE, etc.). We realize not everyone is doing pure research and for students with 'imple-

*Advisors: Stephanie Weirich (sweirich@cis.upenn.edu), Richard Eisenberg (eir@cis.upenn.edu).

mentation' projects such sources may be rare. No matter the case, your sources need to be reputable.

Let us return to your factorization proposal. You should put out the earliest related work; naïve methods like trial divison and the Sieve of Eratosthenes, but state they are of no modern relevance. Then discuss modern methods like the Quadratic Sieve and General Number Field Sieve. Note the humongous time and memory bounds of these algorithms. But wait! You are going to propose a better way ...

3. PROJECT PROPOSAL

Now is the time to introduce your proposed project in all of its glory. Admittedly, this is not the easiest since you probably have not done much actual research yet. Even so, setting and realizing realistic research goals is an important skill. Begin by summarizing what you are going to do and the expected benefit it will bring.

3.1 Anticipated Approach

To begin, we plan to create a Coq formalization of the semantics of System F as defined in Types and Programming Languages. We will then prove that progress and preservation hold for this formalization. Once we have properly proven these properties, we plan to extend our formalization to include coercions without datatypes. Given this extension, we will then adjust our verification to account for the addition of these coercions. This will be a subset of System FC missing features such as datatypes and type families.

3.2 Technical Challenges

In this subsection note where you anticipate having novel difficulty. Maybe you have never setup a MySQL database or even used SQL before at all – yes, that is a challenge – but not one readers care about. More novel would be the fact that many buildings on Penn's campus look similar and your classifier may be inaccurate in such instances. The purpose of this section is two-fold: 1) you will think about which parts of your project would require the most time and effort and 2) you will convince the reader that this is a project worth undertaking.

3.3 Evaluation Criteria

This formalization for System FC has never been done. Upon completion, we will provide assurance of the correctness of the type system of Haskell, a widely used programming language. We would also provide a building block for verifications of other features of System FC.

4. RESEARCH TIMELINE

- ALREADY COMPLETED: Understand the formal definition of System F.
- PRIOR-TO THANKSGIVING :
- PRIOR-TO CHRISTMAS :
- COMPLETION TASKS : Formalize System FC, verify formal language, prove translation between formal language and System FC as implemented in GHC.
- IF THERE'S TIME : Verify other GHC features, such as optimizations and extensions

5. REFERENCES

[1] The Comprehensive TeX Archive Network (CTAN). A (not so) short introduction to LaTeX2e. `http://www.ctan.org/tex-archive/info/lshort/english/`.

[2] Radoslav Ivanov, Miroslav Pajic, and Insup Lee. Attack-resilient sensor fusion. In *DATE'14: Design, Automation and Test in Europe*, 2014.

[3] Wikibooks. LaTeX. `http://en.wikibooks.org/wiki/LaTeX`. Note: Students should not cite Wikis!

APPENDIX

A. OTHER SPECIFICS

Your proposal need not have appendices like this section and the next but we still have info to share:

1. PROPOSAL LENGTH: We require that your proposal be 4–5 pages in length, bibliography included. Be careful, LaTeX and our style-file in particular are *extremely* space efficient. An 9-page MS-Word document could easily become a 5-page LaTeX one.
2. PLAGARISM: **DO NOT** plagarize. If you are caught, you will fail the class (*i.e.*, not graduate), or worse.

B. LaTeX EXAMPLES

At this point, the proposal specification is complete. From here on out, we are just going to show off some commonly used LaTeX technique. Be sure to look at the 'code behind' and see Tab. 1, Eqn. 1 and Fig. 1 for the output! Keep in mind that the appendix is usually not a good place for your figures. Place them where you need them and remember to refer to them in the body of your text; otherwise, the reader will keep reading and will miss them!

$$M(p) = \int_0^\infty (1+\alpha x)^{-\gamma} x^{p-1} dx \tag{1}$$

User Type	Cleanup%	Honesty%
Good	90-100%	100%
Purely Malicous	0-10%	0%
Malicious Provider	0-10%	100%
Feedback Malicous	90-100%	0%
Disguised Malicous	50-100%	50-100%
Sybil Attacker	0-10%	Irrelevant

Table 1: Example Table

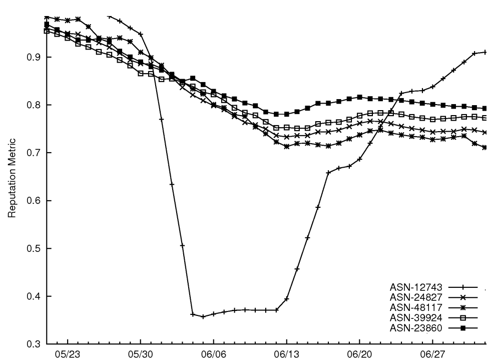

Figure 1: Example Figure/Graph