SCS Senior Thesis Prospectus

Jacob Imola
Project Advisor: Jean Yang

May 11, 2017

1 Abstract

Releasing some private data from a database can easily to leaking more data than intended. A canonical example of this is the Netflix movie ratings dataset, where Netflix released the anonymized ratings of their users, but attackers were able to use similarities with another database to infer many of the names. Therefore, it is important to understand how much information a program leaks. We seek to design a programming language that will give us insight into this quantity. Our language will give us a guarantee through its type system about how much declassification a particular program has. The types will decidable, much like liquid types in Liquid Haskell, and will be relational, meaning they involve two programs, rather than one. This will give our language practical importance because it could be implemented and utililzed by real applications.

2 Problem

In modern databases, it's often useful to aggregate data, like counting the average salary of a group of individuals. In order to compute such values, it is necessary to release, or declassify, some information about the database. Of course, the more data we release, the more we compromise the privacy of the dataset; for instance, an attacker could combine the average salary in an unexpected way with another dataset to learn more that what was intended. Therefore, a programming language that could tell us in advance of the potential information leaks of a program would be very useful. We could use this programming language to help us write code that is guaranteed to adhere to a certain level of declassification.

3 Research Plan

I plan to read about refinement types and aquaint myself with liquid types, which are a small, decidable subset of refinement types. These types are used in the language Liquid Haskell, which will be a good starting place [VSJ14]. They are also used in [KF07]. It will also be important to thoroughly understand what progress has been made already, and where these languages fall short. I have already read about the languages PINQ [McS10] and Fuzz [RP10], and I will read about a few more, such as the language in [BFG$^+$16]. Fuzz will be particularly useful because it uses a relational typesystem; its only shortcoming is that it's undecidable. Relational types are also used in [BFG$^+$14] to control the runtime of a program, but its ideas could be applied to privacy.

My contribution will be making a language with a relational, decidable type system, which will be a framework on which we can form a cost semantics. This cost semantics could then be applied to privacy. However, I will only do the cost semantics if there is time; my primany goal is typesystem. My hope is that the language will be a mixture of Fuzz and liquid types in that it uses strong, but decidable, types. I would also like to implement some or all of this language to demonstrate its pracitcal use. I will adhere to the following schedule:

- Quarter 1: Read and understand the referenced papers. Think about how to combine their ideas. Write some applications in Liquid Haskell.
- Quarters 2-3: Decide on a typesystem for the language. Write the rules governing the statics, amending them as necessary. Attempt to make the statics as expressive as possible while still being decidable. Think about use cases of the languages for inspiration for what to include. Write the dynamics of the language, which will be easier. Ensure the language has useful properties, such as typesafety, preservation, and progress.
- Quarter 4: Implement the language. If it isn't feasible to do the whole language, then implement just a subset of it that will demonstrate its features. Get rough ideas for what a cost semantics in my language will look like.

Student	Date
Supervisor	Date

References

[BFG+14] Gilles Barthe, Cédric Fournet, Benjamin Grégoire, Pierre-Yves Strub, Nikhil Swamy, and Santiago Zanella-Béguelin. Probabilistic relational verification for cryptographic implementations. *SIGPLAN Not.*, 49(1):193–205, January 2014.

[BFG+16] Gilles Barthe, Gian Pietro Farina, Marco Gaboardi, Emilio Jesús Gallego Arias, Andy Gordon, Justin Hsu, and Pierre-Yves Strub. Differentially private bayesian programming. *CoRR*, abs/1605.00283, 2016.

[KF07] Kenneth Knowles and Cormac Flanagan. Type reconstruction for general refinement types. In *Proceedings of the 16th European Symposium on Programming*, ESOP'07, pages 505–519, Berlin, Heidelberg, 2007. Springer-Verlag.

[McS10] Frank McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, September 2010.

[RP10] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. *SIGPLAN Not.*, 45(9):157–168, September 2010.

[VSJ14] Niki Vazou, Eric L. Seidel, and Ranjit Jhala. Liquidhaskell: Experience with refinement types in the real world. *SIGPLAN Not.*, 49(12):39–51, September 2014.