Chapter 7 (draft): Conclusion

Mark Lemay

December 30, 2021

This thesis has attempted to articulate and address a common hesitation around dependent types. Programmers do not want do be interrupted if their is only a chance of an error. Addressing this legitimate concern has lead to a new way of treating warnings in a dependent type system. By creating a parallel system where checks are made and given runtime behavior, programmers still get all of the benefits, but fewer drawbacks of dependent type systems.

This turned out to be surpassingly more subtle then expected. As we saw in Chapter 3, the programmer's intent needs to be inferred, so that a reasonable check can be localized. This is possible through an extension to bidirectional type checking. Runtime errors complicate the semantics, this issue was sidestepped by applying a new relation that extracts blame. Checks need their own runtime behavior, which is strait forward in the pure functional setting.

Further, user defined data turned out to be far more complicated then anticipated. Extending pattern matching to track equalities seems like a clever idea, however the formalism in Chapter 5 is not as simple as we might like. It is unclear if a simpler approach is possible.

Finally, in Chapter 6, there are several ways to improve the current system and build towards future work. more

The approach to warnings presented in this thesis may be more generally applicable. Type systems can still be designed to harshly avoid errors, but by creating a parallel system where checks are made and given runtime behavior, the type system will be less imposing to new users. For instance, many interesting linear type systems are currently being explored, allowing warnings may make these systems more usable to programmers who are not used to those restrictions.

Dependent types have seemed on the verge of mainstream use for decades. While Dependent types are not mainstream yet, they have the unique potential to bridge the gap between those who program and those who prove. Each community has built invaluable expertise that could benefit the other. Once that connection is made solid, more robust software is the least we can expect.

While this thesis has not single handedly made this connection, I think it is a necessary piece of the puzzle.

Part I
TODO

Todo list

1 notes

2 unused